Stochastic Graph Exploration *

Aris Anagnostopoulos
Sapienza University of Rome
aris@diag.uniroma1.it

Ilan R. Cohen
CWI, Amsterdam
ilanrcohen@gmail.com

Stefano Leonardi [1]
Sapienza University of Rome
leonardi@diag.uniroma1.it

Jakub Łącki
Google Research, New York
jlacki@google.com

Abstract

Exploring large-scale networks is a time consuming and expensive task which is usually operated in a complex and uncertain environment. A crucial aspect of network exploration is the development of suitable strategies that decide which nodes and edges to probe at each stage of the process.

To model this process, we introduce the *stochastic graph exploration problem*. The input is an undirected graph $G = (V, E)$ with a source vertex s, stochastic edge costs drawn from a distribution $\pi_e, e \in E$, and rewards on vertices of maximum value R. The goal is to find a set F of edges of total cost at most B such that the subgraph of G induced by F is connected, contains s, and maximizes the total reward. This problem generalizes the stochastic knapsack problem and other stochastic probing problems recently studied.

Our focus is on the development of efficient nonadaptive strategies that are competitive against the optimal adaptive strategy. A major challenge is the fact that the problem has an $\Omega(n)$ adaptivity gap even on a tree of n vertices. This is in sharp contrast with $O(1)$ adaptivity gap of the stochastic knapsack problem, which is a special case of our problem. We circumvent this negative result by showing that $O(\log nR)$ resource augmentation suffices to obtain $O(1)$ approximation on trees and $O(\log nR)$ approximation on general graphs. To achieve this result, we reduce stochastic graph exploration to a memoryless process—the *minesweeper* problem—which assigns to every edge a probability that the process terminates when the edge is probed. For this problem, interesting in its own, we present an optimal polynomial time algorithm on trees and an $O(\log nR)$ approximation for general graphs.

We study also the problem in which the maximum cost of an edge is a logarithmic fraction of the budget. We show that under this condition, there exist polynomial-time oblivious strategies that use $1 + \epsilon$ budget, whose adaptivity gaps on trees and general graphs are $1 + \epsilon$ and $8 + \epsilon$, respectively. Finally, we provide additional results on the structure and the complexity of nonadaptive and adaptive strategies.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Stochastic approximation

Keywords and phrases stochastic optimization, graph exploration, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.131

---

* Partially done while the authors were visiting the Algorithms and Uncertainty program at the Simons Institute for the Theory of Computing, Berkeley. The authors would like to thank Yossi Azar, Anupam Gupta, Peter Auer and C. Seshadhri for fruitful discussions.

[1] Partially supported by ERC Advanced Grant 788893 AMDROMA "Algorithmic and Mechanism Design Research in Online Markets".

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 131; pp. 131:1–131:29
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Exploring networked data is a time consuming and expensive task which is also subject to several limitations. For example, social networks can be explored only through the use of specific APIs made available by the provider which restrict the number of nodes that can be probed and limit the number of neighbors of each node that can be discovered with one probe. The cost and the difficulty of exploring large-scale networks can be an obstacle to collecting suitable snapshots for the purpose of testing new network analysis tools. The testing is more often executed on static networks made available in public repositories [20, 21] collected in the past for other purposes. It is therefore of crucial importance the development of effective and efficient methods to explore large-scale networks.

The core of a network exploration method is the definition of a probing strategy that decides which nodes or edges to probe at each stage of the process. Both the edge-probe and the node-probe models are useful in this setting. In the case of the exploration of social networks, a node-probing strategy allows to gain knowledge on a subset of the neighbors of the probed node. In the case of the exploration of the Twitter graph, an edge-probing strategy allows to gain information on those tweets of a user that are retweeted from his followers.

One main difficulty in the definition of an effective probing strategy is the intrinsic uncertain nature in terms of cost and probability of success of the process of discovering links in a network, especially if these links represent complex relationships between nodes. In order to confirm the existence of a link between two nodes, it may be required to execute several experiments whose outcome cannot be predicted in advance. Examples are the in-vitro reactions between proteins needed to discover protein-to-protein interaction networks [8, 26] or the influence between humans in social networks.

The second main difficulty stems from the adaptive nature of the optimal probing strategy that needs to be updated from time to time while new parts of the network are discovered. Adaptive strategies are computationally expensive, given that they must be continuously updated. In the case of large network exploration, the communication cost of adaptive strategies is also high since many machines are usually working in parallel at the exploration process, and the updated strategy must be communicated to the machines participating in the process. We are therefore interested in devising *nonadaptive probing strategies* that are simple and that define the sequence of probes in advance before the process is started. The obvious drawback is that nonadaptive probing strategies may be suboptimal.

Several recent works [19, 24, 25] have focused on the task of exploring real-world networks when a limited budget is available. However, these papers do not provide a comprehensive theoretical study of these problems. In this work we initiated the study of exploring an undirected network from a root node. The graph has costs on the edges and rewards on the vertices. A budget limits the total cost of the of the graph edges that are probed.

More formally, the input of the *stochastic graph exploration problem* is an undirected graph $G = (V, E)$ with a source vertex $s \in V$, *stochastic* edge costs $C : E \to \mathbb{R}_{\geq 0}$ distributed according to π_e, $e \in E$, and deterministic rewards of vertices $w : V \to \mathbb{R}_{\geq 0}$. (The model can be easily extended to rewards distributed according to independent random variables.) During the graph-exploration process we construct a set of edges $F \subseteq E$ that we probe and we traverse. All vertices of the subgraph of G spanned by F must be connected to s via the edges of F. We probe edges one by one and we add them to F. The actual cost of an edge e, drawn from the distribution π_e, is revealed only when the edge is traversed. The goal is to maximize the 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 we probe an edge such that the total cost exceeds B the process terminates.

In the stochastic graph exploration problem, we aim to design simple polynomial-time computable *nonadaptive strategies* with a reward as close as possible to the reward obtained by the optimal *adaptive strategy*, which decides on the next edge to be traversed after the cost of all previously traversed edges is revealed (see Section 2 for precise definitions). This is customary in a class of stochastic optimization problems [6], for which it is common to bound the *adaptivity gap* of the nonadaptive strategy.

The stochastic graph exploration problem generalizes some important stochastic optimization problems. If the graph G is a star graph, our problem models *exactly* the stochastic knapsack problem [6, 10]. Stochastic knapsack admits an $O(1)$ adaptivity gap, that is, there exists an optimal nonadaptive strategy, which approximates the optimal adaptive strategy up to a constant factor. The nonadaptive strategy is devised by exploiting a suitable LP relaxation for the problem because the standard formulation has an unbounded integrality gap defined as the worst-case ratio between the optimal integral cost and optimal fractional cost of the LP. In the LP version of the problem that is used, the costs of the edges are reduced to their truncated (by the maximum budget) expected costs and the rewards are also reduced by the probability that the cost of the item is below the maximum budget.

If the network we need to explore is a tree, the stochastic graph problem is a stochastic knapsack problem with precedence constraints: only a subset of items are available in the beginning and adding each item to the knapsack will make some new items—the direct descendants of the explored node—available. Unfortunately, as opposed to the knapsack problem, the adaptivity gap of the stochastic graph exploration problem that we consider is unbounded even on a tree network and therefore the LP-based approach of stochastic knapsack cannot directly be extended.

The stochastic graph exploration problem also models stochastic graph probing problems. Probing problems in graphs have been introduced [9,16] because of their applications to kidney exchange and online dating. Consider a probing probability for each edge $p : E \to [0, 1]$, that is, edge e will materialize with probability $p(e)$ each time is probed, independently of the other edges and of the previous probes. The goal is to maximize the number of vertices that are connected to a source vertex s by the set F of edges that have been successfully probed when the total number of probes is limited by B. Nonadaptive strategies probe a list of edges in a sequence till success or the total budget B is reached. The stochastic graph exploration problem we study models the stochastic graph probing problem by setting the costs of the edges distributed according to $\mathbf{Pr}(C_e = i) = (1 - p(e))^{i-1}p(e)$, with i being the number of probes needed to discover edge e.

1.1 Summary of Our Results

Our main contribution is the definition of the stochastic graph exploration problem and the study of the adaptivity gap of nonadaptive probing strategies. Here is a summary of our results:

Our first result is an $\Omega(n)$ adaptivity gap for the stochastic graph exploration problem even on a spider graph, which is a tree containing a single node of degree more than two. (Observe that the problem for a simple path is easy because the optimal strategy will traverse sequentially the edges of the path starting from the root.)

One first direction we pursue to circumvent the impossibility result is to allow a limited amount of resource augmentation: instead of using budget B, we allow the algorithm to use a budget of $\beta \cdot B$, for some value of β. 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). We present polynomial time computable nonadaptive strategies in a graph of n vertices that are $(O(1), O(\log nR))$-approximate for trees and $(O(\log nR), O(\log nR))$-approximate for general graphs, with R being the maximum reward of a vertex.

The idea is to transform the stochastic exploration problem into a memoryless stochastic process, which we call the *minesweeper problem*, and which may be of independent interest. In the minesweeper problem, the budget and the edge costs are replaced by probabilities $p(e)$, which are specified for every edge e. When an edge e is probed, the process stops with probability $1 - p(e)$. Hence, the final reward of a vertex is discounted by the probability that the strategy does not stop before the vertex is acquired. The minesweeper problem is, in fact, a special case of stochastic graph exploration, where the support of each π_e (distribution of cost of edge e) is $\{0, B+1\}$ and the budget is B.

We prove that an α-approximate strategy for the minesweeper problem implies an $(O(\alpha), O(\log nR))$-approximate nonadaptive strategy for the stochastic graph exploration problem. The idea of the reduction is as follows. We construct a minesweeper problem instance, where $p(e) = \mathbf{Pr}(\pi_e < X_B)$, where X_B is random variable that follows an exponential distribution with parameter B. We first show that, for any subset of edges F, the probability that their total cost in the stochastic graph exploration is at most B is at most a constant factor of the probability that minesweeper would stop on this set. On the other hand, the expected additional reward that can be achieved from minesweeper after the total cost becomes larger than $O(B \log nR)$ is negligible.

We then show how to compute in polynomial time an optimal strategy for the minesweeper problem on trees and an $O(\log nR)$-approximate strategy on general graphs. These results imply imply an $(O(1), O(\log nR))$-approximate strategy for trees and an $(O(\log nR), O(\log nR))$-approximate strategy for general graphs. To show the optimal result on trees we prove two facts. First, the order of traversal of the edges in each subtree can be determined independently. Second, we show a simple optimality condition which helps us determine how many edges from each subtree should be probed before switching to a different subtree. We remark that our approach is in a spirit similar to the greedy optimal strategy defined by the Gittins index [11, 12] for multi-armed bandit problems. However, differently from the standard setting of the Gittins index, in the minesweeper problem, a whole new set of arms is made available for each node of the tree reached by the exploration process. Moreover, in the minesweeper problem, the discount factor is not constant because it depends on the probability assigned to the traversed edge. This approach is not viable for general graphs, and we provide an approximate solution instead, by showing a reduction of minesweeper to max-prize problem [7].

We also pursue a second direction to circumvent the lower bound on the adaptivity gap for trees: we restrict the distributions by considering the case when the edge costs are bounded by $\frac{\epsilon^2 B}{c \log n}$ for a suitable constant c. We show, under this condition, the existence of a polynomial time computable $(1+\epsilon, 1+\epsilon)$-approximate nonadaptive strategy for trees and $(1+\epsilon, 8+\epsilon)$-approximate nonadaptive strategy for any graph G. We note that this approach can be extended to prove a result with resource augmentation similar to the one we obtained through reduction to the minesweeper problem. Yet, we believe that both the minesweeper problem and the reduction technique can be of independent interest.

Our final result is related to the problem of finding a nonadaptive probing strategy that is $(o(n), O(1))$-approximate. We leave open this challenging problem even for trees. However, we establish an interesting result for the characterization of nonadaptive strategies. We prove that any nonadaptive strategy that probes edges in order until it succeeds or until the budget is exceeded can be $(O(1), O(1))$-approximated by a set strategy, which probes all edges at once and obtains a reward only if all edges of a set are successfully probed within budget. We specifically prove that the adaptivity gap of a nonadaptive strategy can be approximated up to a factor of 6 by a set strategy that uses budget $9B$. We use this result to give an algorithm for finding a strategy for trees, which is $(O(1), O(1))$-approximate, compared to the best *nonadaptive* strategy. Surprisingly, the resulting strategy is adaptive.

1.2 Related Work

The adaptivity gap of stochastic problems has been studied for the knapsack problem [6, 10] which is a special case of the problem we study. The adaptivity gap has also been studied for budgeted multi-armed bandits [13, 14, 22] by resorting to suitable linear programming relaxation. Differently from 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, 4, 9] motivated from kidney exchange and for more general classes of matroid optimization problems [16, 17].

The stochastic graph exploration problem we introduce is also related to the *stochastic orienteering* problem [5, 15]. 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.

1.3 Organization of the Paper

In Section 2 we formally define our problems. In Section 3 we show the lower bounds on the adaptivity gap for stochastic graph exploration. In Section 4 we show our reduction to the minesweeper problem and our results for stochastic graph exploration with resource augmentation. In Section 5 we present a near-optimal set strategy for trees. In Section 6 we present our results for the case of edges of small costs and, finally, in Section 7 we study the power of resource augmentation for relating the cost of nonadaptive strategies to the cost of optimal set strategies.

2 Problem Definition

We start by an auxiliary definition. Let $G = (V, E)$, with $|V| = n$, be an undirected graph and $s \in V$. We say that a set $F \subseteq E$ is *connected to s* if F induces a connected subgraph of G and s is the endpoint of at least one $e \in F$.

Let us now define the StochasticExploration problem (in the following sometimes abbreviated by SGE). This problem instance is given by a tuple (G, s, C, w), where G is an undirected graph $G = (V, E)$, $s \in V$ is a source vertex, C is a function that assigns *stochastic* edge costs to each edge, and $w : V \to \mathbb{R}_{\geq 0}$ is a function that assigns (deterministic) reward to each vertex.[2] And we denote R as the maximum reward of a vertex i.e. $R = \max_{v \in V} w(v)$. Formally, for each $e \in E$, $C(e)$ is a random variable distributed according to π_e that takes

[2] The results hold also if the rewards are random variables that are independent of each other and the edge costs. It suffices to replace each reward with its expected value.

values in $\mathbb{R}_{\geq 0}$, all random variables $C(e)$ being jointly independent. For an edge (u, v) we will often denote $C(u, v) = C((u, v))$.

Consider the following single-player game. The player has an initial budget of B ($B = 1$ if not specified) and maintains an initially empty subset F of E, which we call the set of *acquired edges*. In each step the player can choose an edge $e \in E \setminus F$ and *probe* it (if $F = E$, the game finishes). Probing an edge e is only allowed when $F \cup \{e\}$ is connected to s. When e is probed, the actual cost $C(e)$ of e, drawn from the distribution π_e, is revealed. If the cost e is not greater than the remaining budget, e is *acquired* (added to F) and $C(e)$ is subtracted from the budget. If $C(e)$ exceeds the remaining budget, the game finishes. The goal of the player is to maximize the final *payoff* of F, which is the total reward of all vertices in the subgraph of G induced by F.

Let us now define the MineSweeper problem, which we often abbreviate to MS. This problem is defined by a tuple (G, s, p, w), where G is an undirected graph, $s \in V$ is a start vertex, $p : E \to [0, 1]$ is a function that assigns to each edge e the probability that e materializes and $w : V \to \mathbb{R}_{\geq 0}$ is a function that assigns (deterministic) reward to each vertex. The only difference between MS and SGE is in how edges are probed. There are no edge costs or budget. Instead, whenever an edge e is probed, it materializes (independently of the other edges) with probability $p(e)$ and is acquired immediately. If the edge does not materialize, the process ends immediately. Note that as in SGE, probing an edge e is only allowed when $F \cup \{e\}$ is connected to s. Note that the MineSweeper problem is a special case of the StochasticExploration problem, by letting, for each edge e, π_e be the distribution in which with probability $p(e)$ we obtain the value 0 and with probability $1 - p(e)$ the value $B + 1$.

We consider the following types of strategies for both problems:

- An *adaptive* strategy is a mapping from the set of already acquired edges (and the remaining budget, in the case of SGE) to the next edge to be probed.
- A *nonadaptive* strategy, also called a *list* strategy, is described by a sequence $e_1, \ldots, e_k$ consisting of distinct elements of E, such that for each $1 \leq i \leq k$, the set $\{e_1, \ldots, e_i\}$ is connected to s. In this strategy, the edges are simply probed according to their order in the sequence.
- A *set* strategy is a nonadaptive strategy with the additional restriction that it does not obtain any payoff if it does not acquire all edges from the list.[3]

For a strategy S for SGE, we denote by $r(\mathcal{I}_{\text{SGE}}, S, B)$ the expected payoff of strategy S for the SGE problem instance $\mathcal{I}_{\text{SGE}} = (G, s, C, w)$ with initial budget of B, which is the expected reward of the set of nodes in the returned solution. When $B = 1$ we sometimes omit the third argument of $r(\cdot)$. Similarly, we denote by $r_{\text{MS}}(\mathcal{I}_{\text{MS}}, S)$ the expected payoff of strategy S for the MS problem instance $\mathcal{I}_{\text{MS}}$. We call a strategy S *optimal* for $\mathcal{I}$ with budget B, if for all strategies S', $r(\mathcal{I}, S, B) \geq r(\mathcal{I}, S', B)$. Let OPT_{ad} be the optimal adaptive strategy for the SGE problem and OPT_{na} be the optimal nonadaptive strategy. We call a strategy S α-approximate, if for each instance $\mathcal{I}$, $r(\mathcal{I}, S) \geq 1/\alpha \cdot r(\mathcal{I}, \text{OPT}_{\text{ad}})$. Finally, an algorithm ALG is (α, β)-approximate if for any instance $\mathcal{I}$ it computes a α-approximate strategy by using a β factor resource augmentation, i.e. $r(\mathcal{I}, \text{ALG}(\mathcal{I}), \beta \cdot B) \geq 1/\alpha \cdot r(\mathcal{I}, \text{OPT}_{\text{ad}}, B)$.

[3] Note that we abuse earlier definitions slightly for the sake of simplicity.

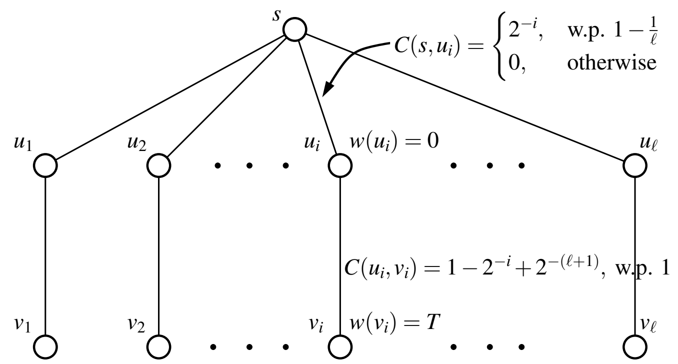

Figure 1 An instance in which the optimal adaptive strategy obtains a payoff which is $\Omega(n)$ larger than the payoff of the optimal nonadaptive strategy.

3 Lower Bounds

In this section we prove a lower bound on the *adaptivity gap* of StochasticExploration. Namely, we show an instance $\mathcal{I}_{LB} = (G, s, C, w)$ such that $r(\mathcal{I}_{LB}, \mathrm{OPT}_{\mathrm{ad}})/r(\mathcal{I}_{LB}, \mathrm{OPT}_{\mathrm{na}}) = \Omega(n)$, where $\mathrm{OPT}_{\mathrm{ad}}$ and $\mathrm{OPT}_{\mathrm{na}}$ denote the optimal adaptive and nonadaptive strategies.

The instance $\mathcal{I}_{LB}$ is shown in Figure 1. The graph G contains the set of nodes $\{s, u_1, u_2, \ldots, u_\ell, v_1, \ldots, v_\ell\}$, and the set of edges (s, u_i) and (u_i, v_i) for each $i \in [\ell]$. For each $i \in [\ell]$ we set $w(u_i) = 0$, $w(v_i) = T$, $C(s, u_i) = 2^{-i}$ with probability $1 - 1/l$ and 0 otherwise, and $C(u_i, v_i) = 1 - 2^{-i} + 2^{-(\ell+1)}$ with probability 1.

▶ **Lemma 1.** *Let OPT_{ad} and OPT_{na} denote the optimal adaptive and nonadaptive strategies for instance $\mathcal{I}_{LB}$. Then, $r(\mathcal{I}_{LB}, OPT_{ad})/r(\mathcal{I}_{LB}, OPT_{na}) = \Omega(n)$.*

One natural approach for StochasticExploration instance is to replace the stochastic edge costs with the truncated expected costs, that is, set the cost of an edge e to $\mathbb{E}[\min\{1, C(e)\}]$. However as this following example illustrates this approach does not lead to a good solution, even if constant budget augmentation is allowed.

▶ **Lemma 2.** *Let OPT_{ad} denote the optimal adaptive strategy for an instance $\mathcal{I}$ and let n be the number of vertices in the instance. Let OPT_{na} be the optimal nonadaptive strategy computed on instance $\mathcal{I}_{TR}$ obtained from $\mathcal{I}$ by setting edge costs $\mathbb{E}[\min\{1, C(e)\}]$, $e \in E$. Assume the nonadaptive algorithm is allowed to use a budget of $1 < c < n/10$. Then, there exists an instance $\mathcal{I}$ such that $r(\mathcal{I}, OPT_{ad})/r(\mathcal{I}_{TR}, OPT_{na}) = \Omega(n/2^{2c})$.*

4 The General Case and the Minesweeper Problem

In this section we describe algorithms for solving StochasticExploration, which use logarithmic budget augmentation. We first show how to reduce an instance of SGE to MineSweeper and then present solutions for MineSweeper on trees and general graphs. During the description of the reduction we also introduce the logarithmic budget augmentation. First, we observe that in the MineSweeper problem we do not have budget so there is no history that an algorithm may have to remember, except for the edges that it has probed (and succeeded). This implies the following:

▶ **Observation 3.** *There exists an optimal strategy for the* MineSweeper *problem that is nonadaptive.*

4.1 Reduction from StochasticExploration to MineSweeper

In this section we show how, given an instance $\mathcal{I}_{\mathrm{SGE}} = (G, s, C, w)$ of StochasticExploration, we transform it to an instance $\mathcal{I}_{\mathrm{MS}} = (G, s, p, w)$ of MineSweeper. The graph and the rewards remain the same; the challenge is to define the correct edge probability function $p(\cdot)$ for MS and relate it to the cost function $C(\cdot)$ of SGE. For each edge e' we transform the cost distribution $C(e')$ to the probability that the edge materializes, $p(e')$ (a scalar). Let $X_{e'}$ be a random variable distributed according to the exponential distribution with parameter 1, let $c_{e'}$ be the cost, which is distributed according to $C(e')$, and we set $p(e') = \mathbf{Pr}(X_{e'} > c_{e'})$. Next we show how this choice couples the two problems.

First, we show that for any subset of edges F the probability that their total cost in SGE is at most 1 is at most a factor e times of the probability that all the edges in F materialize, and therefore MS does not stop on this set. Let $\mathcal{E}_F$ be the event that all the edges in F materialize and $\mathcal{G}_F$ the event that $\sum_{e' \in F} c_{e'} \leq 1$. The following lemma makes use of properties of the exponential distribution.

▶ **Lemma 4.** *For any $F \subseteq E$ we have that $\mathbf{Pr}(\mathcal{G}_F) \leq e \cdot \mathbf{Pr}(\mathcal{E}_F)$.*

This lemma allows us to prove the following lemma, which gives a strategy for MS that is competitive with the optimal adaptive strategy for SGE. The idea behind the proof is to define a strategy for MS in such a way that we can couple the execution of the two strategies in the corresponding problems.

▶ **Lemma 5.** *Consider an* SGE *instance $\mathcal{I}_{\mathrm{SGE}} = (G, s, C, w)$ and let $\mathcal{I}_{\mathrm{MS}} = (G, s, p, w)$ be an instance for* MS *as defined previously. Let OPT_{ad} denote the optimal adaptive strategy for* SGE *and OPT_{MS} the optimal strategy for* MS*. We have that*

$$r((G, s, C, w), OPT_{ad}, 1) \leq e \cdot r_{MS}((G, s, C, w), OPT_{MS}).$$

Recall from Observation 3 that the optimal strategy for the MineSweeper problem is nonadaptive, therefore it can be specified by a list of edges that are selected sequentially until for one of them there is a failure. Let $\mathrm{OPT}_{\mathrm{MS}}$ be such an optimal sequence of edges. Next we show that the sequence of edges $\mathrm{OPT}_{\mathrm{MS}}$ can provide an approximate result to the StochasticExploration problem if we allow for some budget augmentation.

▶ **Lemma 6.** *Consider an* SGE *instance $\mathcal{I}_{\mathrm{SGE}} = (G, s, C, w)$ and let $\mathcal{I}_{\mathrm{MS}} = (G, s, p, w)$ be an instance for* MS *as defined previously. Let OPT_{MS} be the optimal sequence of edges for the* MineSweeper *instance, and let S be the (nonadaptive) strategy for* StochasticExploration *that probes the same edges, in the same order. Then we have that*

$$r((G, s, C, w), S, 2\ln(nR)) \geq r_{MS}((G, s, C, w), OPT_{MS}) - o(1),$$

where $R = \max_{v \in V} w(v)$.

Collecting the results of Lemmas 5 and 6 we obtain the following theorem.

▶ **Theorem 7.** *Consider an* SGE *instance $\mathcal{I}_{\mathrm{SGE}} = (G, s, C, w)$ and let $\mathcal{I}_{\mathrm{MS}} = (G, s, p, w)$ be an instance for* MS *as defined previously. Then*

$$r((G, s, C, w), OPT_{na}, 2\ln(nR)) + o(1) \geq r_{MS}((G, s, C, w), OPT_{MS}) \geq \frac{r((G, s, C, w), 1, OPT_{ad})}{e}.$$

4.2 MineSweeper on Trees

We show that the minesweeper problem on trees can be solved optimally in near-linear time.

▶ **Theorem 8.** *Consider the instance $\mathcal{I} = (T, s, p, w)$ of the minesweeper problem, where T is a tree. The optimal strategy, OPT_{MS}, for* MineSweeper *on T can be computed in $O(n \log n)$ time, where n is the number of vertices of T.*

The algorithm is surprisingly simple and based on a greedy approach. We define the utility of an edge to be the expected payoff from probing it, divided by the probability that the edge does not materialize. The algorithm is based on two observations. First, we observe that if there is a node x in the graph with a single child y and the utility of the edge xy is larger than the utility of the edge connecting x and its parent, then without loss of optimality we can assume that the edge xy is probed right after the edge connecting x and its parent, so we can merge these two edges into a single one. Second, if there is a node x, such that one can probe all edges in the subtree of x in the order of decreasing utilities (and not violate the constraint that an edge can be probed only after one of its endpoints has been acquired) then one can replace the entire subtree of x with a line, which is a subtree imposing the concrete order of probing edges. It turns out that by using both these rules one can find the optimal order of probing edges efficiently.

We obtain the algorithm by generalizing some existing results from the area of scheduling. At the same time our analysis is arguably simpler. We give the proof of Theorem 8 in the full version of the paper.

4.3 MineSweeper on general graphs

In this section we present an algorithmic solution to Minesweeper for general graphs, which provides a bicriteria approximation for our problem. We prove the following theorem.

▶ **Theorem 9.** *Consider the instance $\mathcal{I} = (G, s, p, w)$ of the minesweeper problem, where $G = (V, E)$ is an undirected graph. An $O(\log nR)$-approximate strategy can be computed in polynomial time.*

In the following we provide a sketch of the proof. Assume that the optimal solution is the sequence of edges $S^* = (e_1, \ldots, e_k)$. We first observe that the edges in S^* must form a tree. Define $\mathcal{M}(E')$ to be the event that all the edges in the set E' materialize. Also let $w(e_1, \ldots, e_i) = \sum_{j=1}^{i} w(e_i)$. Then S^* is a sequence that maximizes

$$O^* = \sum_{i=1}^{k} \mathbf{Pr}(\mathcal{M}(\{e_1, \ldots, e_i\}), \neg\mathcal{M}(\{e_{i+1}\})) \cdot w(e_1, \ldots, e_i)\,.$$

For $\ell = 0, 1, \ldots, \ln nR$, define $I(\ell)$ to be all values j such that $w(e_1, \ldots, e_j) \in [2^\ell, 2^{\ell+1} - 1]$, and $\iota(\ell)$ to be the smallest such j.

We can write after some manipulations:

$$O^* \leq \sum_{\ell=0}^{\ln nR} 2w(e_1, \ldots, e_{\iota(\ell)}) \cdot \mathbf{Pr}\big(\mathcal{M}(\{e_1, \ldots, e_{\iota(\ell)}\})\big) \leq 2\ln(nR) \cdot w(\tilde{E}) \cdot \mathbf{Pr}\big(\mathcal{M}(\tilde{E})\big)\,,$$

with $\tilde{E} \subset E$ being the set of edges that defines a tree that contains s and maximizes $w(\tilde{E}) \cdot \mathbf{Pr}\big(\mathcal{M}(\tilde{E})\big)$. Therefore, our goal becomes that of finding that set of edges $\tilde{E}$ that forms a tree and maximizes $w(\tilde{E}) \cdot \mathbf{Pr}\big(\mathcal{M}(\tilde{E})\big)$.

For this purpose, we use the problem of *max-prize tree*. In the max-prize tree [7] we are given an undirected graph $G = (V, E)$ with a source vertex $s \in V$, (deterministic) edge costs $c : E \to \mathbb{R}_{\geq 0}$, deterministic rewards on the vertices $w : V \to \mathbb{R}_{\geq 0}$, and a budget $B \in \mathbb{R}$. The objective is to build a subgraph $G' = (V', E')$ of G such that (1) G' is a tree, (2) $s \in V'$, and (3) $\sum_{e \in E'} c(e) \leq B$, that maximizes $\sum_{v \in V'} w(v)$.

We use for our approximation the 8-approximation algorithm for the max-prize–tree problem given by Blum et al. [7].

5 Approximating Set Strategy on Trees

In this section we show an algorithm for computing a strategy for trees, which is $(1, 1+\epsilon)$-approximate compared to the optimal set strategy. The strategy itself is adaptive.

▶ **Lemma 10.** *Let $\mathcal{I} = (G, s, C, w)$ be a* SGE *instance, where G is a tree. Let OPT_{set} be the optimal set strategy for $\mathcal{I}$. Then, in $O(n^4/\epsilon^2)$ time we can compute an adaptive strategy S, such that $r(\mathcal{I}, S, 1+\epsilon) \geq r(\mathcal{I}, OPT_{set}, 1)$. Moreover, if edge costs are not stochastic, that is, the support of each distribution π_e has size 1, the algorithm runs in $O(n^3/\epsilon)$ time and the resulting strategy is not adaptive.*

We briefly describe the ideas behind the algorithm. Consider the instance $\mathcal{I} = (T, s, C, w)$, where T is a tree. We root the tree at s and assume an order on the children of each node. Consider the sequence $P = \langle e_1, \ldots, e_n \rangle$ of the tree edges built with the following recursive algorithm. Given a node of T, iterate through its descendant edges (according to their order) and for each such edge output it and recur on the other endpoint. This traverses the tree in a preorder fashion. We define $\prec$ to be the linear order on the edges of T induced by this traversal. In the following, we assume that the edges are ordered according to $\prec$, for example, by a maximal element of a set of edges, we mean the edges that is largest according to $\prec$.

We say that a subset A of edges of T is *feasible*, if each edge $e \in A$ is either incident to the root of T, or the parent edge of e also belongs to A. Observe that given sufficient budget, a strategy can acquire any feasible set of edges of T. This follows from the fact that for each edge e of T, its parent comes before it in P. Our algorithm will probe some feasible set of edges according to the order $\prec$, that is, after probing an edge e it will not probe any edge f such that $f \prec e$.

The algorithm for computing our strategy is based on dynamic programming. A simple and inefficient approach is to use an exponential number of states. Namely, each state can be characterized by the set of edges acquired so far, denoted by A, and the remaining budget, which we discretize to a multiple of ϵ/n. Knowing the set A allows us to find all such edges e that $A \cup \{e\}$ is a feasible set and e comes after the maximal element of A in the order $\prec$. The key idea is that we can improve the number of states to polynomial, by taking advantage of the following property of the ordering $\prec$.

▶ **Lemma 11.** *Let A be a nonempty feasible set of edges of T and let e be the maximal edge of A. Given e (and without knowing A) we can compute the set F_e of all edges f such that $e \prec f$ and $A \cup \{f\}$ is a feasible set.*

6 Bounded Edge Costs

In this section, we deal with the special case of StochasticExploration, where the cost of each edge is bounded by $O(\frac{\epsilon^2}{\ln n})$ and the ratio between the smallest and largest reward R

is polynomial in n. We prove that in this setting a $(O(1), 1+\epsilon)$ strategy for SGE can be computed in polynomial time.

▶ **Theorem 12.** *Let $\mathcal{I} = (G, s, C, w)$ be an instance of* SGE*, where $C(e) = O(\frac{\epsilon^2}{\ln n})$ (for each edge e and some $0 < \epsilon = O(1)$), $R \leq \epsilon n^{O(1)}$, and the smallest reward is 1. Then, in polynomial time, we can compute a nonadaptive $(O(1), 1+\epsilon)$-approximate strategy for $\mathcal{I}$. Additionally, if G is a tree, then in time $O(n^3/\epsilon)$ we can compute a nonadaptive $(1+\epsilon, 1+\epsilon)$-approximate strategy for $\mathcal{I}$.*

To prove the theorem, we consider the following strategy. We replace the stochastic edge costs with their expected values (i.e., the edge cost distributions in the modified instance have size 1). Then, we show that the optimal set strategy using budget augmented by a factor of $1+\epsilon$ gives a $(1+\epsilon)$-approximate solution.

For ease of notation, we scale the edge costs and the budgets by a factor of $\Theta(\epsilon^2/\ln n)$, so that the edge costs are bounded by 1 and the available budget is $B = O(\epsilon^2/\ln n)$.

First, we bound the payoff of an adaptive strategy when the expected cost of its acquired edges is more than $B \cdot (1+\epsilon)$. Let $\mu_e = \mathbf{E}[C(e)]$, and $\mu(F) = \sum_{e \in F} \mu_e$.

▶ **Lemma 13.** *Let $0 < \epsilon < 1/3$ and let $\mathcal{I} = (G, s, C, w)$ be an instance of* SGE*, in which $B \geq 5c/\epsilon^2 \cdot \ln n$. Let F be a set of edges acquired by some adaptive strategy. If $\mu(F) \geq (1+\epsilon) \cdot B$ then the probability that $C(F) \leq B$ is at most n^{-c}.*

Next, we show that if the expected cost of some set of edges is close to the budget, then this cost is highly concentrated around the expected value. This enables us to give a set strategy with small budget augmentation.

▶ **Lemma 14.** *Let $\mathcal{I} = (G, s, C, w)$ be an instance of* SGE*. For any set of edges F and any $\tilde{B} \geq 5c/\epsilon^2 \cdot \ln n$, if $\mu(F) = \tilde{B}$ then the probability that $C(F) \geq (1+\epsilon)\tilde{B}$ is at most n^{-c}.*

▶ **Lemma 15.** *Let $\mathcal{I} = (G, s, C, w)$ be an instance of* SGE*, where $B \geq 5c/\epsilon^2 \ln n$, the maximum reward R satisfies $R \leq \epsilon n^{c-1}$, and the minimum reward is 1. Let $\mathcal{I}_e$ be obtained from $\mathcal{I}$ by replacing each edge cost with its expected value. Let OPT^{ϵ}_{set} be the optimal set strategy using budget $(1+\epsilon)B$ for $\mathcal{I}_e$ and OPT_{ad} be the optimal adaptive strategy using budget B for $\mathcal{I}$. Then, $(1+\epsilon)r(\mathcal{I}, OPT^{\epsilon}_{set}, (1+\epsilon)B) \geq r(\mathcal{I}, OPT_{ad}, B)$.*

Observe that finding the optimal set strategy on $\mathcal{I}_e$ is NP-hard, as it generalizes the knapsack problem. However, it becomes tractable, if we augment the budget. In particular, for trees, we use the algorithm of Lemma 10, and for general graphs, in Section 4.3, we show how to use the solution of the max-prize problem.

7 Nonadaptive strategies

In this section we consider nonadaptive strategies for the stochastic exploration problem. The main result of this section is that, for the graph exploration problem, that there exists a *set-strategy* with a constant budget augmentation, which is a constant competitive compared to the best nonadaptive algorithm. Recall that, a *set-strategy* is to choose a set of edges (without an internal order) and to try to probe all of the edge in that set. The gain of strategy for a set of edges, is nonzero only if the *entire set* was successfully probed (i.e., if the total cost of the set is smaller than the budget), and then it collects the rewards of all the vertices connected to this set. Therefore, the expected gain of *set-strategy* given a set of edges, is the total gain of vertices spanned by these edges times the probability that the total cost of these edges would not be greater than the specified budget.

First, we are able to show how much is the increment in the probability to successfully probe a set, when using a constant budget augmentation.

7.1 Power of Budget Augmentation

Let $S = \{e_1, e_2, \dots, e_n\}$ be a set of edges and let $c_i \triangleq C(e_i)$. Define $C_k^n = \sum_{i=k}^n c_i$ the realized cost of the subset of the edges $\{e_k, \dots e_n\}$ and, for ease of notation, let $C^j = C_1^j$. For any $i \in [n]$ let $P_i(a)$ be the probability that the sum of cost of the edges $\{e_1, \dots e_i\}$ is at most a, that is, $P_i(a) = \mathbf{Pr}\big(C^i \le a\big)$.

The next lemma will allow us to take advantage of budget augmentation.

▶ **Lemma 16.** *Assume that for each edge e_i, $i \in [n]$ we have $c_i \in [0, 1]$. Then*

$$P_n(3) \ge P_n(1)\,(1 - \ln(P_n(1)))\,.$$

Interestingly, the multiplicative factor increases as the probability to succeed with the original budget decreases. We will use this fact, but to compare to a list-strategy we need stronger guarantees, we simply use the above lemma twice and deduce the following.

▶ **Corollary 17.**

$$P_n(9) \ge P_n(1)\frac{(1 - \ln(P_n(1)))^2}{2}$$

7.2 List Strategy vs. Set Strategy

Now, we are ready to prove the main claim of this section, that we are able to compare the strategies using a budget augmentation. Consider an SGE problem instance $\mathcal{I} = (G, s, C, w)$. Let $\mathrm{S}_{ls} = \langle e_1, \dots, e_n \rangle$ be a nonadaptive strategy (a feasible sequence of edges) and let v_i denote the vertex whose reward is obtained when e_i is acquired. The expected payoff of probing the list with budget $B(\ge 1)$ is by linearity of expectation:

$$r(\mathcal{I}, \mathrm{S}_{ls}, B) = \sum_{j=1}^{n} w(v_j) \cdot \mathbf{Pr}\big(C^j \le B\big)\,.$$

Given a nonadaptive strategy $\mathrm{S}_{ls} = \langle e_1, \dots, e_n \rangle$, consider n different set strategies S_k, for $k = \{1 \dots n\}$, where $S_k = \{e_1, \dots e_k\}$. Note that the expected payoff of S_k with budget $9 \cdot B$ is

$$r(\mathcal{I}, S_k, 9B) = \mathbf{Pr}\big(C^k \le 9B\big) \cdot \sum_{j=1}^{k} w(v_j).$$

Finally, we show that there exists $k \in \{1, \dots, n\}$ such that the set strategy S_k with budget $9B$ obtains a constant fraction of strategy S_{ls}.

▶ **Lemma 18.**

$$\max_k\{r(\mathcal{I}, S_k, 9B)\} \ge 0.46 \cdot r(\mathcal{I}, S_{ls}, B).$$

7.3 Algorithm for Trees

By combining Lemma 18 with the algorithm of Lemma 10, we obtain the following.

▶ **Theorem 19.** *Let $\mathcal{I} = (G, s, C, w)$ be a* SGE *instance, where G is a tree. Let OPT_{na} be the optimal nonadaptive strategy for $\mathcal{I}$. Then, in $O(n^4/\epsilon^2)$ time we can compute an adaptive strategy S, such that $r(\mathcal{I}, S, 9+\epsilon) \geq 0.46 \cdot r(\mathcal{I}, OPT_{na}, 1)$.*

8 Conclusions

In this work we have introduced the stochastic exploration problem on graphs which generalizes the stochastic knapsack problem [6, 10]. We proved that, differently from stochastic knapsack, no $o(n)$ adaptivity gap is possible unless we allow some resource augmentation on the budget. We provided algorithms with bounded adaptivity gap and logarithmic resource augmentation by reducing stochastic exploration to a related memoryless problem—the minesweeper problem. We also considered the case of edges with small costs for which it is possible to provide an algorithm with $O(1)$ adaptivity gap and $O(1)$ resource augmentation. The most challenging problem left open from our work is the one of devising an algorithm with $O(1)$ approximation factor that uses only $O(1)$ resource augmentation for general graphs. The problem is open even for trees. We provided a set of additional results on the structure of optimal adaptive strategies and on the power of resource augmentation for set strategies with respect to list strategies that can help in addressing this problem.

References

1 Marek Adamczyk. Improved analysis of the greedy algorithm for stochastic matching. *Information Processing Letters*, 111(15):731 – 737, 2011. URL: `http://www.sciencedirect.com/science/article/pii/S002001901100127X`, `doi:http://dx.doi.org/10.1016/j.ipl.2011.05.007`.

2 D Adolphson and T Ch Hu. Optimal linear ordering. *SIAM Journal on Applied Mathematics*, 25(3):403–423, 1973.

3 Donald L. Adolphson. Single machine job sequencing with precedence constraints. *SIAM J. Comput.*, 6(1):40–54, 1977. URL: `https://doi.org/10.1137/0206002`, `doi:10.1137/0206002`.

4 Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When lp is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica*, 63(4):733–762, Aug 2012.

5 Nikhil Bansal and Viswanath Nagarajan. *On the Adaptivity Gap of Stochastic Orienteering*, pages 114–125. Springer International Publishing, Cham, 2014. URL: `http://dx.doi.org/10.1007/978-3-319-07557-0_10`, `doi:10.1007/978-3-319-07557-0_10`.

6 Anand Bhalgat, Ashish Goel, and Sanjeev Khanna. *Improved Approximation Results for Stochastic Knapsack Problems*, pages 1647–1665. URL: `http://epubs.siam.org/doi/abs/10.1137/1.9781611973082.127`, `arXiv:http://epubs.siam.org/doi/pdf/10.1137/1.9781611973082.127`, `doi:10.1137/1.9781611973082.127`.

7 Avrim Blum, Shuchi Chawla, David R. Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. Approximation algorithms for orienteering and discounted-reward tsp. *SIAM Journal on Computing*, 37(2):653–670, 2007.

8 Michael Caldera, Pisanu Buphamalai, Felix Mueller, and Jorg Menche. Interactome-based approaches to human disease. *Current Opinion in Systems Biology*, 3:88 – 94, 2017.

9 Ning Chen, Nicole Immorlica, Anna R. Karlin, Mohammad Mahdian, and Atri Rudra. *Approximating Matches Made in Heaven*, pages 266–278. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. URL: `http://dx.doi.org/10.1007/978-3-642-02927-1_23`, `doi:10.1007/978-3-642-02927-1_23`.

10 Brian C. Dean, Michel X. Goemans, and Jan Vondrak. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of Operations Research*, 33(4):945–964, 2008. doi:10.1287/moor.1080.0330.

11 Esther Frostig and GIDEON WEISS. Four proofs of gittins multiarmed bandit theorem, 1999.

12 Author(s) J. C. Gittins and J. C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society, Series B*, pages 148–177, 1979.

13 Sudipto Guha and Kamesh Munagala. Approximation algorithms for budgeted learning problems. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 104–113, New York, NY, USA, 2007. ACM. URL: http://doi.acm.org/10.1145/1250790.1250807, doi:10.1145/1250790.1250807.

14 Anupam Gupta, Ravishankar Krishnaswamy, Marco Molinaro, and R. Ravi. 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, October 22-25, 2011*, pages 827–836, 2011.

15 Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, and R. Ravi. Running errands in time: Approximation algorithms for stochastic orienteering. *Mathematics of Operations Research*, 40(1):56–79, 2015.

16 Anupam Gupta and Viswanath Nagarajan. *A Stochastic Probing Problem with Applications*, pages 205–216. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

17 Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Algorithms and adaptivity gaps for stochastic probing. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 1731–1747, Philadelphia, PA, USA, 2016. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=2884435.2884555.

18 WA Horn. Single-machine job sequencing with treelike precedence ordering and linear delay penalties. *SIAM Journal on Applied Mathematics*, 23(2):189–202, 1972.

19 R. Laishram, K. Areekijseree, and S. Soundarajan. Predicted max degree sampling: Sampling in directed networks to maximize node coverage through crawling. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 940–945, May 2017. doi:10.1109/INFCOMW.2017.8116502.

20 Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

21 Jure Leskovec and Rok Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1, 2016.

22 Will Ma. *Improvements and Generalizations of Stochastic Knapsack and Multi-Armed Bandit Approximation Algorithms: Extended Abstract*, pages 1154–1163. URL: http://epubs.siam.org/doi/abs/10.1137/1.9781611973402.85, arXiv:http://epubs.siam.org/doi/pdf/10.1137/1.9781611973402.85, doi:10.1137/1.9781611973402.85.

23 Colin McDiarmid. *Concentration*, pages 195–248. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. URL: https://doi.org/10.1007/978-3-662-12788-9_6, doi:10.1007/978-3-662-12788-9_6.

24 S. Soundarajan, T. Eliassi-Rad, B. Gallagher, and A. Pinar. Maxreach: Reducing network incompleteness through node probes. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 152–157, Aug 2016. doi:10.1109/ASONAM.2016.7752227.

25 Sucheta Soundarajan, Tina Eliassi-Rad, Brian Gallagher, and Ali Pinar. *epsilon* w gxx: Adaptive edge probing for enhancing incomplete networks. In *Proceedings of the 2017 ACM on Web Science Conference*, WebSci '17, pages 161–170, New York, NY, USA, 2017. ACM. URL: http://doi.acm.org/10.1145/3091478.3091492, doi:10.1145/3091478.3091492.

26 M. Vidal, ME Cusick, and AL Barabasi. Interactome networks and human disease. *Cell.*, 144(6):986 – 98, 2011.

A Omitted Proofs

▶ **Lemma 1.** *Let OPT_{ad} and OPT_{na} denote the optimal adaptive and nonadaptive strategies for instance $\mathcal{I}_{LB}$. Then, $r(\mathcal{I}_{LB}, OPT_{ad})/r(\mathcal{I}_{LB}, OPT_{na}) = \Omega(n)$.*

Proof. Let S_{AD} be the adaptive strategy that probes the edges $(s, u_\ell), (s, u_{\ell-1}), (s, u_{\ell-2}), \ldots$ until it probes an edge (s, u_i) whose cost turns out to be 0. After that it probes the edge (u_k, v_k). Note that right before probing edge (u_k, v_k) the budget that has been used so far is

$$2^{-\ell} + 2^{-(\ell-1)} + \cdots + 2^{-(k+1)} < 2^{-k} - 2^{-(\ell+1)}.$$

We have that $C(u_k, v_k) = 1 - (2^{-k} - 2^{-(\ell+1)})$ with probability 1, therefore there is sufficient budget to acquire the edge (u_k, v_k) and obtain the reward of $w(v_k) = T$. The probability that the cost of some edge (s, u_i) is 0 is

$$\sum_{k=0}^{\ell-1} \left(1 - \frac{1}{\ell}\right)^k \cdot \frac{1}{\ell} = \frac{1 - \left(1 - \frac{1}{\ell}\right)^\ell}{1 - \left(1 - \frac{1}{\ell}\right)} \cdot \frac{1}{\ell} \geq 1 - \frac{1}{e}.$$

Therefore, the expected payoff of strategy S_{AD} is $r(\mathcal{I}_{LB}, S_{\mathrm{AD}}, 1) \geq T\left(1 - \frac{1}{e}\right)$.

Consider now any nonadaptive strategy S. If the payoff of S is nonzero, it clearly attempts to probe at least one edge (u_i, v_i). Consider the first such edge (u_i, v_i). Clearly, the strategy must probe the edge (s, u_i) before attempting to probe (u_i, v_i).

Thus with probability $1 - 1/\ell$ we have that $C(s, u_i) = 2^{-i}$, and in that case, just before probing edge (u_i, v_i), the leftover budget is at most $1 - 2^{-i}$. Then, since $C(u_i, v_i) = 1 - 2^{-i} + 2^{-(\ell+1)}$, the remaining budget is not sufficient, so the game terminates with total payoff 0 In the remaining case we have $C(s, u_i) = 0$, which happens with probability $1/\ell$. In this case the strategy acquires edge (u_i, v_i), but after that the remaining budget is not sufficient to reach another node v_j. This means that the expected payoff of strategy S is at most $r(\mathcal{I}_{LB}, S, 1) \leq \frac{T}{\ell}$. The lemma follows. ◀

▶ **Lemma 2.** *Let OPT_{ad} denote the optimal adaptive strategy for an instance $\mathcal{I}$ and let n be the number of vertices in the instance. Let OPT_{na} be the optimal nonadaptive strategy computed on instance $\mathcal{I}_{TR}$ obtained from $\mathcal{I}$ by setting edge costs $\mathbb{E}[\min\{1, C(e)\}]$, $e \in E$. Assume the nonadaptive algorithm is allowed to use a budget of $1 < c < n/10$. Then, there exists an instance $\mathcal{I}$ such that $r(\mathcal{I}, OPT_{ad})/r(\mathcal{I}_{TR}, OPT_{na}) = \Omega(n/2^{2c})$.*

Proof. Let us define an instance $\mathcal{I} = (G, s, C, w)$, where $G = (V, E)$ is a graph such that $V = \{s, u_1, \ldots, u_n, v\}$, and $E = \{(s, u_1), (u_1, u_2), \ldots, (u_{n-1}, u_n), (s, v)\}$. For each edge $i \in \{1, \ldots, 2c+1\}$, $C(u_i, u_{i+1}) = 1$ with probability 1/2 and 0 otherwise, for $i > 2c+1$. $C(u_i, u_{i+1}) = 0$ with probability 1. The reward function of u_i is 0 for $i \in \{1, \ldots, 2c+1\}$ and 1 for $i > 2c+1$. In addition, $C(s, v) = 1$ with probability 1, and $w(v) = 1$.

The expected truncated cost of the edges (u_i, u_{i+1}) for $i \in \{1, \ldots, 2c+1\}$ is 1/2. Therefore, an algorithm which uses the expected truncated costs will assume it cannot reach u_{2c+2}. Hence it will acquire just the edge (s, v) and obtain a reward of 1. At the same time, with probability 2^{-2c-2}, an algorithm can obtain the reward from all $\Omega(n)$ vertices u_i. This gives a gap of $\Omega(n/2^{2c})$. ◀

▶ **Lemma 4.** *For any $F \subseteq E$ we have that $\mathbf{Pr}(\mathcal{G}_F) \leq e \cdot \mathbf{Pr}(\mathcal{E}_F)$.*

Proof. First, note that, for three independent random variables X_1, X_2, X_3 distributed according to an exponential distribution with the same parameter λ, we have that

$$\begin{aligned}\mathbf{Pr}(X_1 > a) \cdot \mathbf{Pr}(X_2 > b) &= \mathbf{Pr}(X_3 > a) \cdot \mathbf{Pr}(X_3 > b) = \mathbf{Pr}(X_3 > a) \cdot \mathbf{Pr}(X_3 > a + b \,|\, X_3 > a) \\ &= \mathbf{Pr}(X_3 > a + b)\,,\end{aligned}$$

where the second equality follows from the memorylessness property of the exponential distribution.

Let X be a random variable drawn from the exponential distribution with parameter 1. Using the fact that we proved above, we have that

$$\mathbf{Pr}(\mathcal{E}_F) = \prod_{e \in F} \mathbf{Pr}(X_e > c_e) = \mathbf{Pr}\left(X > \sum_{e \in F} c_e\right),$$

and that

$$\mathbf{Pr}(\mathcal{E}_F \mid \mathcal{G}_F) = \mathbf{Pr}\left(X > \sum_{e \in F} c_e \,\middle|\, \sum_{e \in F} c_e \leq 1\right) \geq \mathbf{Pr}(X > 1) = e^{-1}.$$

Therefore, we finish the proof of the lemma by combining this fact with

$$\mathbf{Pr}(\mathcal{E}_F) \geq \mathbf{Pr}(\mathcal{E}_F, \mathcal{G}_F) = \mathbf{Pr}(\mathcal{E}_F \mid \mathcal{G}_F) \cdot \mathbf{Pr}(\mathcal{G}_F)\,.$$

◀

▶ **Lemma 5.** *Consider an* SGE *instance* $\mathcal{I}_{\mathrm{SGE}} = (G, s, C, w)$ *and let* $\mathcal{I}_{\mathrm{MS}} = (G, s, p, w)$ *be an instance for* MS *as defined previously. Let* OPT_{ad} *denote the optimal adaptive strategy for* SGE *and* OPT_{MS} *the optimal strategy for* MS*. We have that*

$$r((G, s, C, w), OPT_{ad}, 1) \leq e \cdot r_{MS}((G, s, C, w), OPT_{MS}).$$

Proof. Consider the optimal adaptive strategy $\mathrm{OPT_{ad}}$ and a strategy for minesweeper (call it S) that selects to probe the same edges as $\mathrm{OPT_{ad}}$. We will lower bound the probability that $\mathrm{OPT_{ad}}$ terminates (i.e., exhausts its budget) before the strategy is applied to MS. Consider a graph G, and let $g(x, G)$ be the probability that $\mathrm{OPT_{ad}}$ for SGE on graph G with budget x does not finish after strategy S. for MS. Abusing notation, define

$$g(x, i) = \inf_{G=(V,E):|E|=i} g(x, G).$$

We will prove by induction on i and x that $g(x, i) \geq e^{-x}$.

First note that for $i = 0$ there is not any edge to probe so we are done. So consider $i \geq 1$ and assume that for any $x \geq 0$ and any graph $G' = (V', E')$ with $|E'| \leq i - 1$ we have that $g(x, G') \geq e^{-x}$. For $x = 0$, $\mathrm{OPT_{ad}}$ has no budget for SGE so it cannot continue and the claim is trivially true. Thus, consider the case that $i \geq 1$ and $x > 0$. Assume that we have graph $G = (V, E)$ and for $e \in E$ denote by G_e the graph in which edge e has been contracted. Then notice that we have that the probability that $\mathrm{OPT_{ad}}$ will not terminate after strategy S of MS is equal to the probability that (1) $\mathrm{OPT_{ad}}$ finishes when probing the next edge, or (2) that it does not, and neither does S but in the next steps $\mathrm{OPT_{ad}}$ does not terminate after MS. Therefore, for any x and any graph G with i edges, we have that

$$\begin{aligned}
g(x, G) &= \sum_{c_e \geq x} \mathbf{Pr}(C(e) = c_e) + \sum_{c_e < x} \mathbf{Pr}(C(e) = c_e) \cdot g(x - c_e, G_e) \cdot \mathbf{Pr}(X > c_e) \\
&\geq \sum_{c_e \geq x} \mathbf{Pr}(C(e) = c_e) + \sum_{c_e < x} \mathbf{Pr}(C(e) = c_e) \cdot g(x - c_e, i - 1) \cdot \mathbf{Pr}(X > c_e) \\
&\geq \sum_{c_e \geq x} \mathbf{Pr}(C(e) = c_e) \cdot e^{-x} + \sum_{c_e < x} \mathbf{Pr}(C(e) = c_e) \cdot e^{c_e - x} \cdot e^{-c_e} \geq e^{-x},
\end{aligned}$$

where the equality follows from the fact that in the MS problem that we defined edge e does not die with probability $\mathbf{Pr}(X > c_e)$, the first inequality from the fact that graph G_e has $i-1$ edges, and the second inequality from the fact that $e^{-x} \leq 1$ and from the induction hypothesis.

Because this holds for all graphs G with i edges, we deduce that for any $x \geq 0$ we have that $g(x, i) \geq e^{-x}$.

Therefore, we have that $g(1, G) \geq 1/e$, which means that there exists a strategy for the MS problem that with probability at least $1/e$ does not stop before strategy $\mathrm{OPT}_{\mathrm{ad}}$, and therefore has expected payoff of at least $r((G, s, C, w), \mathrm{OPT}_{\mathrm{ad}}, 1)/e$. ◀

▶ **Lemma 6.** *Consider an* SGE *instance* $\mathcal{I}_{\mathrm{SGE}} = (G, s, C, w)$ *and let* $\mathcal{I}_{\mathrm{MS}} = (G, s, p, w)$ *be an instance for* MS *as defined previously. Let OPT_{MS} be the optimal sequence of edges for the* MineSweeper *instance, and let S be the (nonadaptive) strategy for* StochasticExploration *that probes the same edges, in the same order. Then we have that*

$$r((G, s, C, w), S, 2\ln(nR)) \geq r_{MS}((G, s, C, w), OPT_{MS}) - o(1),$$

where $R = \max_{v \in V} w(v)$.

Proof. Let $L = \langle e_1, \ldots e_{n-1} \rangle$, the (optimal) list of edges that $\mathrm{OPT}_{\mathrm{MS}}$ probes. By definition, strategy S for StochasticExploration probes the same edges. We consider an execution of MineSweeper using strategy $\mathrm{OPT}_{\mathrm{MS}}$ and an execution of StochasticExploration with strategy S. We couple the two executions, such that for each edge e the value c_e be the same in both executions.

Consider a materialization of the values c_{e_1} up to $c_{e_{n-1}}$ of the edges in L. Let r be the index such that

$$\sum_{i=1}^{r} c_{e_i} \leq 2\ln(nR)$$

and

$$\sum_{i=1}^{r+1} c_{e_i} > 2\ln(nR).$$

Then notice that the revenue of strategy S for StochasticExploration is a value W, which is the sum of the rewards of the nodes reachable from the source node s using edges $e_1, \ldots, e_r$. We now show that the expected value of $\mathrm{OPT}_{\mathrm{MS}}$ is $W + o(1)$. Recall that when $\mathrm{OPT}_{\mathrm{MS}}$ probes an edge e_i, the probability that it materializes is $p(e_i) = \mathbf{Pr}(X_{e_i} > c_{e_i})$, with X_{e_i} being an exponentially distributed random variable with parameter 1. If $\mathrm{OPT}_{\mathrm{MS}}$ succeeds up to edge e_r, it acquires reward up to W. Otherwise, the probability that $\mathrm{OPT}_{\mathrm{MS}}$ succeeds for more than r edges is at most

$$\prod_{i=1}^{r+1} p(e_i) = \prod_{i=1}^{r+1} e^{-c_{e_i}} = e^{-\sum_{i=1}^{r+1} c_{e_i}} < e^{-2\ln(nR)} = (nR)^{-2}.$$

The maximum reward that it can acquire is upper bounded by nR, therefore, the expected reward of $\mathrm{OPT}_{\mathrm{MS}}$ (given the values c_{e_i}) is

$$W + \frac{nR}{(nR)^2} = W + (nR)^{-1}.$$

Therefore, for each materialization of the edge costs, we have that the expected reward of $\mathrm{OPT}_{\mathrm{MS}}$ on MINESWEEPER is at most $(nR)^{-1}$ higher than the reward of strategy S for STOCHASTICEXPLORATION, implying that the expected reward of S is at most $(nR)^{-1}$ less than the expected reward of $\mathrm{OPT}_{\mathrm{MS}}$. ◀

▶ **Theorem 9.** *Consider the instance $\mathcal{I} = (G, s, p, w)$ of the minesweeper problem, where $G = (V, E)$ is an undirected graph. An $O(\log nR)$-approximate strategy can be computed in polynomial time.*

Proof. Assume that the optimal solution is the sequence of edges $S^* = (e_1, \ldots, e_k)$. Define $\mathcal{M}(E')$ to be the event that all the edges in the set E' materialize. Also let $w(e_1, \ldots, e_i) = \sum_{j=1}^{i} w(e_i)$. Then S^* is a sequence that maximizes

$$O^* = \sum_{i=1}^{k} \mathbf{Pr}(\mathcal{M}(\{e_1, \ldots, e_i\}), \neg\mathcal{M}(\{e_{i+1}\}))) \cdot w(e_1, \ldots, e_i)\,.$$

For $\ell = 0, 1, \ldots, \log nR$, define $I(\ell)$ to be all values j such that $w(e_1, \ldots, e_j) \in [2^\ell, 2^{\ell+1}-1]$, and $\iota(\ell)$ to be the smallest such j.

We can write

$$\begin{aligned} O^* &= \sum_{\ell=0}^{\log nR} \sum_{i \in I(\ell)} \mathbf{Pr}(\mathcal{M}(\{e_1, \ldots, e_j\}), \neg\mathcal{M}(\{e_{i+1})\}) \cdot w(e_1, \ldots, e_j) \\ &\le \sum_{\ell=0}^{\log nR} 2w(e_1, \ldots, e_{\iota(\ell)}) \cdot \sum_{i \in I(\ell))} \mathbf{Pr}(\mathcal{M}(\{e_1, \ldots, e_j\}), \neg\mathcal{M}(\{e_{i+1})\}) \\ &\le \sum_{\ell=0}^{\log nR} 2w(e_1, \ldots, e_{\iota(\ell)}) \cdot \mathbf{Pr}\big(\mathcal{M}(\{e_1, \ldots, e_{\iota(\ell)})\}\big)\,. \end{aligned}$$

Note that the optimal sequence of edges S^* cannot contain a cycle because for each node $v \in V$ we either reach it through a path and we collect value $w(v)$, or we die before and the process stops. Furthermore, we collect value $w(v)$ only for nodes reachable from s through edges of S^* that have materialized. Therefore, the edges in S^* must form a tree.

Let $\tilde{E} \subset E$ be the set of edges that defines a tree that contains s and maximizes

$$w(\tilde{E}) \cdot \mathbf{Pr}\big(\mathcal{M}(\tilde{E})\big)\,.$$

For each $\ell = 0, 1, \ldots, \log nR$ we have that

$$w(\tilde{E}) \cdot \mathbf{Pr}\big(\mathcal{M}(\tilde{E})\big) \ge w(e_1, \ldots, e_{\iota(\ell)}) \cdot \mathbf{Pr}\big(\mathcal{M}(e_1, \ldots, e_{\iota(\ell)})\big)$$

and we obtain that

$$O^* \le 2\log(nR) \cdot w(\tilde{E}) \cdot \mathbf{Pr}\big(\mathcal{M}(\tilde{E})\big)\,. \tag{1}$$

Therefore, our goal becomes that of finding that set of edges $\tilde{E}$ that forms a tree and maximizes

$$w(\tilde{E}) \cdot \mathbf{Pr}\big(\mathcal{M}(\tilde{E})\big)\,.$$

We next show how to find a tree that approximates this quantity. Consider the graph with the same vertex and edge set as G, with the same rewards on the vertices $w(\cdot)$, and with edge costs $\psi : E \to \mathbb{R}_{\ge 0}$, defined as $\psi(e) = -\log p(e)$. Note that

$$\mathbf{Pr}\big(\mathcal{M}(\tilde{E})\big) = \prod_{e \in \tilde{E}} p(e) = 2^{-\sum_{e \in \tilde{E}} \psi(e)}.$$

Let $\Psi_{\min} = \min_{e\in E}\psi(e)$, and $\Psi_{\max} = \sum_{e\in E}\psi(e)$. For $\ell \in \{\Psi_{\min}, \Psi_{\min}+1, \ldots, \Psi_{\max}\}$, let T^ℓ be the tree T that (1) has cost $\sum_{e\in T^\ell}\psi(e) \le \ell$, (2) contains node s, and (3) maximizes $w(T)$, defined as $\sum_{v\in T} w(v)$, where we say that $v \in T$ if node v belongs to the tree T.

This is precisely the max-prize–tree problem, and by the discussion just before the proof we can deduce that for each $\ell \in \{\Psi_{\min}, \Psi_{\min}+1, \ldots, \Psi_{\max}\}$ we can compute a tree $\hat{T}^\ell$ with cost at most ℓ and total weight at least $w(T^\ell)/\gamma$. Note that $\Psi_{\max} \le -n\log(\max_e p(e))$, so we solve only a polynomial (in the input length) number of max-prize–tree problems.

Let

$$\ell^* = \arg\max_{\ell}\{2^{-\ell}\cdot w(T^\ell)\}.$$

We have $\sum_{e\in\hat{T}^{\ell^*}}\psi(e) \le \ell^*$, which implies that $\mathbf{Pr}\Big(\mathcal{M}(\hat{T}^{\ell^*})\Big) = \prod_{e\in\hat{T}^{\ell^*}} p(e) \ge 2^{-\ell^*}$. Define

$$\tilde{\ell} = \big\lfloor -\log \mathbf{Pr}\big(\mathcal{M}(\tilde{E})\big)\big\rfloor$$

and notice that $\tilde{E} = T^{\tilde{\ell}}$. Also, notice that we have

$$2^{-\tilde{\ell}} \ge \frac{1}{2}\mathbf{Pr}\big(\mathcal{M}(\tilde{E})\big).$$

Putting everything together, we obtain that, for the solution $\hat{T}^{\ell^*}$ that we compute, we have

$$\mathbf{Pr}\Big(\mathcal{M}(\hat{T}^{\ell^*})\Big)\cdot w(\hat{T}^{\ell^*}) \ge 2^{-\ell^*}\cdot w(\hat{T}^{\ell^*}) \ge \frac{2^{-\ell^*}}{\gamma}\cdot w(T^{\ell^*}) \ge \frac{2^{-\tilde{\ell}}}{\gamma}\cdot w(T^{\tilde{\ell}}) \ge \frac{1}{2\gamma}\mathbf{Pr}\big(\mathcal{M}(\tilde{E})\big)\cdot w(\tilde{E}).$$

Using this in Equation (1), we conclude that

$$\mathbf{Pr}\Big(\mathcal{M}(\hat{T}^{\ell^*})\Big)\cdot w(\hat{T}^{\ell^*}) \ge \frac{1}{4\gamma\log(nR)}\cdot O^*.$$

◀

▶ **Lemma 10.** *Let $\mathcal{I} = (G, s, C, w)$ be a* SGE *instance, where G is a tree. Let OPT_{set} be the optimal set strategy for $\mathcal{I}$. Then, in $O(n^4/\epsilon^2)$ time we can compute an adaptive strategy S, such that $r(\mathcal{I}, S, 1+\epsilon) \ge r(\mathcal{I}, OPT_{set}, 1)$. Moreover, if edge costs are not stochastic, that is, the support of each distribution π_e has size* 1*, the algorithm runs in $O(n^3/\epsilon)$ time and the resulting strategy is not adaptive.*

Proof. We now describe the algorithm computing the strategy. First, we quantize the edge costs, by rounding them up to multiples of ϵ/n. Namely, each time a cost of c is incurred, we actually deduct $\lceil c/(\epsilon/n)\rceil(\epsilon/n)$ from the budget. Because each strategy acquires at most n edges, this turns a strategy using a budget of 1 into a strategy using budget of $1+\epsilon$. Hence, by using budget of $1+\epsilon$, we can assume that the cost of each edge is a multiple of ϵ/n. In particular, at every step the remaining budget in has one of $(1+\epsilon)/(\epsilon/n) = O(n/\epsilon)$ distinct values.

The dynamic programming uses an array $D(e, b)$, indexed by an edge e and the remaining budget b. Note that the array has size $O(n^2/\epsilon)$. The value $D(e, b)$ denotes what is the maximum expected reward that a strategy can get in the remaining part of the game if the first edge to be probed is e and the remaining budget is b. Of course the next edge to be probed after e has to belong to the set F_e (see Lemma 11). Denote by r_e the reward that we obtain from acquiring the edge e. We use the following recursive formula:

$$D(e, b) = \sum_{i=0}^{b/(\epsilon/n)} \mathbf{Pr}(C(e) = i(\epsilon/n))\left(r_e + \max_{f\in F_e} D(f, b - i(\epsilon/n))\right).$$

Observe that there are $O(n^2/\epsilon)$ values of $D(e, b)$ to compute and evaluating each of them requires $O(n^2/\epsilon)$ time. Hence, the dynamic programming requires $O(n^4/\epsilon^2)$ time. The expected payoff of the strategy can be obtained by taking maximum of $D(e, 1+\epsilon)$ over all edges e incident to s. The recursive formula directly translates to an adaptive algorithm. After acquiring the edge e, if the remaining budget is $b - i(\epsilon/n)$, the next edge to probe is $\arg\max_{f \in F_e} D(f, b - i(\epsilon/n))$. Note that this can only be evaluated once we know the remaining budget and thus the obtained strategy is adaptive.

Clearly, the obtained strategy is the optimal adaptive strategy, among strategies that probe edges according to the ordering $\prec$. Because, without loss of generality, we can assume that each set strategy probes edges according to this order, we immediately get that the $r(\mathcal{I}, 1+\epsilon, S) \geq r(\mathcal{I}, 1, \mathrm{OPT}_{\mathrm{set}})$.

Finally, let us consider the case of non-stochastic edge costs. Observe that the sum in the formula for $D(e, b)$, has only single summand, which improves the running time by a factor of n/ϵ. Moreover, the choices of the algorithm can be simulated beforehand, so the final strategy is nonadaptive. ◀

▶ **Lemma 11.** *Let A be a nonempty feasible set of edges of T and let e be the maximal edge of A. Given e (and without knowing A) we can compute the set F_e of all edges f such that $e \prec f$ and $A \cup \{f\}$ is a feasible set.*

Proof. Consider the path S that starts at the root of T and ends with e. Observe that every edge whose one endpoint is on the path and that comes after e in the order $\prec$ belongs to the set F_e. Indeed, by adding each such edge to F we obtain a feasible set. It is also easy to see that adding any other edge larger than e to A would not yield a feasible set. ◀

We use the following theorem in the proof of the next lemma.

▶ **Theorem 20** ([23]). *Let X be a martingale such that $\sum_{i=1}^{t} \mathbf{Var}[X_i \mid X_{i-1}] \leq V$ and $X_i - X_{i-1} \leq M$. Then,*

$$\mathbf{Pr}(X_t - \mathbf{E}[X] \leq -\lambda) \leq \exp\left(\frac{-\lambda^2}{2 \cdot V + M\lambda/3}\right).$$

▶ **Lemma 10.** *Let $\mathcal{I} = (G, s, C, w)$ be a* SGE *instance, where G is a tree. Let OPT_{set} be the optimal set strategy for $\mathcal{I}$. Then, in $O(n^4/\epsilon^2)$ time we can compute an adaptive strategy S, such that $r(\mathcal{I}, S, 1+\epsilon) \geq r(\mathcal{I}, OPT_{set}, 1)$. Moreover, if edge costs are not stochastic, that is, the support of each distribution π_e has size 1, the algorithm runs in $O(n^3/\epsilon)$ time and the resulting strategy is not adaptive.*

Proof. We now describe the algorithm computing the strategy. First, we quantize the edge costs, by rounding them up to multiples of ϵ/n. Namely, each time a cost of c is incurred, we actually deduct $\lceil c/(\epsilon/n) \rceil (\epsilon/n)$ from the budget. Because each strategy acquires at most n edges, this turns a strategy using a budget of 1 into a strategy using budget of $1+\epsilon$. Hence, by using budget of $1+\epsilon$, we can assume that the cost of each edge is a multiple of ϵ/n. In particular, at every step the remaining budget in has one of $(1+\epsilon)/(\epsilon/n) = O(n/\epsilon)$ distinct values.

The dynamic programming uses an array $D(e, b)$, indexed by an edge e and the remaining budget b. Note that the array has size $O(n^2/\epsilon)$. The value $D(e, b)$ denotes what is the maximum expected reward that a strategy can get in the remaining part of the game if the first edge to be probed is e and the remaining budget is b. Of course the next edge to be probed after e has to belong to the set F_e (see Lemma 11). Denote by r_e the reward that we obtain from acquiring the edge e. We use the following recursive formula:

$$D(e,b) = \sum_{i=0}^{b/(\epsilon/n)} \mathbf{Pr}(C(e) = i(\epsilon/n)) \left(r_e + \max_{f \in F_e} D(f, b - i(\epsilon/n)) \right).$$

Observe that there are $O(n^2/\epsilon)$ values of $D(e,b)$ to compute and evaluating each of them requires $O(n^2/\epsilon)$ time. Hence, the dynamic programming requires $O(n^4/\epsilon^2)$ time. The expected payoff of the strategy can be obtained by taking maximum of $D(e, 1+\epsilon)$ over all edges e incident to s. The recursive formula directly translates to an adaptive algorithm. After acquiring the edge e, if the remaining budget is $b - i(\epsilon/n)$, the next edge to probe is $\arg\max_{f \in F_e} D(f, b - i(\epsilon/n))$. Note that this can only be evaluated once we know the remaining budget and thus the obtained strategy is adaptive.

Clearly, the obtained strategy is the optimal adaptive strategy, among strategies that probe edges according to the ordering $\prec$. Because, without loss of generality, we can assume that each set strategy probes edges according to this order, we immediately get that the $r(\mathcal{I}, 1+\epsilon, S) \geq r(\mathcal{I}, 1, \mathrm{OPT}_{\mathrm{set}})$.

Finally, let us consider the case of non-stochastic edge costs. Observe that the sum in the formula for $D(e,b)$, has only single summand, which improves the running time by a factor of n/ϵ. Moreover, the choices of the algorithm can be simulated beforehand, so the final strategy is nonadaptive. ◀

▶ **Lemma 11.** *Let A be a nonempty feasible set of edges of T and let e be the maximal edge of A. Given e (and without knowing A) we can compute the set F_e of all edges f such that $e \prec f$ and $A \cup \{f\}$ is a feasible set.*

Proof. Consider the path S that starts at the root of T and ends with e. Observe that every edge whose one endpoint is on the path and that comes after e in the order $\prec$ belongs to the set F_e. Indeed, by adding each such edge to F we obtain a feasible set. It is also easy to see that adding any other edge larger than e to A would not yield a feasible set. ◀

We use the following theorem in the proof of the next lemma.

▶ **Theorem 21** ([23]). *Let X be a martingale such that $\sum_{i=1}^{t} \mathbf{Var}[X_i \mid X_{i-1}] \leq V$ and $X_i - X_{i-1} \leq M$. Then,*

$$\mathbf{Pr}(X_t - \mathbf{E}[X] \leq -\lambda) \leq \exp\left(\frac{-\lambda^2}{2 \cdot V + M\lambda/3} \right).$$

▶ **Lemma 13.** *Let $0 < \epsilon < 1/3$ and let $\mathcal{I} = (G, s, C, w)$ be an instance of* SGE*, in which $B \geq 5c/\epsilon^2 \cdot \ln n$. Let F be a set of edges acquired by some adaptive strategy. If $\mu(F) \geq (1+\epsilon) \cdot B$ then the probability that $C(F) \leq B$ is at most n^{-c}.*

Proof. In order to bound the probability, it is crucial to exploit the property of irrevocable decisions, which forces the adaptive strategy to keep an item even if its size turns out to be very large. Therefore, we would use the *martingale* framework. For an adaptive strategy let F_t denote the set of the first t items chosen by the strategy. Note that no further items are added to F_t once the cost of the edges in F_t exceeds the budget. Define $X_t = \sum_{e \in F_t} (c(e) - \mu_e)$. It is easy to verify that X_t is a martingale, since $\mathbf{E}[X_{t+1} \mid X_t] = X_t$. Note that $X_0 = 0$ by definition. Next, we bound the variance of

$$\mathbf{Var}[X_t \mid X_{t-1}] = \mathbf{Var}[(C(e) - \mu_e)] = \mathbf{E}\left[C(e)^2\right] - \mathbf{E}[C(e)]^2 \leq \mathbf{E}\left[C(e)^2\right] \leq \mathbf{E}[C(e)],$$

where the last inequality holds since the cost of each edge is bounded by 1.

We now apply Theorem 21 with $X_t = C(F_t) - \mu(F_t)$, $M = 1$ and $\lambda = \epsilon B$. Observe that $\sum_{i=1}^{t} \mathbf{Var}[X_i \mid X_{i-1}] \leq (1+\epsilon) \cdot B$. Hence, we have

$$\begin{aligned}\mathbf{Pr}(C(F_t) \leq B) &\leq \mathbf{Pr}(C(F_t) \leq \mu(F_t) - \epsilon B) \leq \mathbf{Pr}(C(F_t) - \mu(F_t) \leq -\epsilon B)\\ &\leq \exp\left(\frac{-\epsilon^2 \cdot B^2}{2(1+\epsilon)B + \epsilon \cdot B/3}\right) \leq \exp\left(\frac{-\epsilon^2 \cdot B}{2(1+\epsilon) + \epsilon/3}\right)\\ &\leq \exp\left(\frac{-4c \log n}{2(1+\epsilon) + \epsilon/3}\right) \leq n^{-c}.\end{aligned}$$

◀

▶ **Lemma 22** (Chernoff bound). *Let X be the sum of binary independent random variables $X_1, X_2, ..., X_n$ where $X_i \in [0,1]$ Then, for any $\delta > 1$, $\mathbf{Pr}(X > (1+\delta) \cdot \mathbf{E}[X]) \leq \exp(-\delta \cdot \mathbf{E}[X]/3)$.*

▶ **Lemma 14.** *Let $\mathcal{I} = (G, s, C, w)$ be an instance of* SGE. *For any set of edges F and any $\tilde{B} \geq 5c/\epsilon^2 \cdot \ln n$, if $\mu(F) = \tilde{B}$ then the probability that $C(F) \geq (1+\epsilon)\tilde{B}$ is at most n^{-c}.*

Proof. We apply Chernoff bound by setting $X_i = C(e)$. We have $X = C(S)$ and

$$\mathbf{Pr}\big(C(F) > (1+\epsilon) \cdot \tilde{B}\big) \leq \exp\left(\frac{-\epsilon \cdot \tilde{B}}{3}\right) \leq n^{-c}.$$

◀

▶ **Lemma 15.** *Let $\mathcal{I} = (G, s, C, w)$ be an instance of* SGE*, where $B \geq 5c/\epsilon^2 \ln n$, the maximum reward R satisfies $R \leq \epsilon n^{c-1}$, and the minimum reward is 1. Let $\mathcal{I}_e$ be obtained from $\mathcal{I}$ by replacing each edge cost with its expected value. Let OPT^{ϵ}_{set} be the optimal set strategy using budget $(1+\epsilon)B$ for $\mathcal{I}_e$ and OPT_{ad} be the optimal adaptive strategy using budget B for $\mathcal{I}$. Then, $(1+\epsilon)r(\mathcal{I}, OPT^{\epsilon}_{set}, (1+\epsilon)B) \geq r(\mathcal{I}, OPT_{ad}, B)$.*

Proof. First, we bound $r(\mathcal{I}, \mathrm{OPT}_{\mathrm{ad}}, B)$. This payoff is the sum of two terms: the expected payoff when the expected cost of the acquired edges is at most $(1+\epsilon) \cdot B$ and the payoff when the expected cost of edges is greater than $(1+\epsilon)B$. In the following we use $r(\mathcal{I}, S, B \mid \mathcal{E})$ to denote the expected payoff of strategy S, conditioned on the event $\mathcal{E}$.

Let F be the set of edges acquired by $\mathrm{OPT}_{\mathrm{ad}}$. Observe that $r(\mathcal{I}, \mathrm{OPT}_{\mathrm{ad}}, B \mid \mu(F) \leq (1+\epsilon)B) \leq r(\mathcal{I}_e, \mathrm{OPT}^{\epsilon}_{\mathrm{set}}, (1+\epsilon)B)$. This follows from the fact that the conditioning on $\mu(F) \leq (1+\epsilon)B$ limits the possible sets of edges that the adaptive strategy can acquire to the sets of edges considered by the set strategy. We now obtain

$$\begin{aligned} r(\mathcal{I}, \mathrm{OPT}_{\mathrm{ad}}, B) &\leq r(\mathcal{I}, \mathrm{OPT}_{\mathrm{ad}}, B \mid \mu(F) \leq (1+\epsilon)B) + r(\mathcal{I}, \mathrm{OPT}_{\mathrm{ad}} \mid \mu(F) > (1+\epsilon)B)\\ &\leq r(\mathcal{I}, \mathrm{OPT}_{\mathrm{ad}}, B \mid \mu(F) \leq (1+\epsilon)B) + n \cdot R \cdot n^{-c}\\ &\leq r(\mathcal{I}_e, \mathrm{OPT}^{\epsilon}_{\mathrm{set}}, (1+\epsilon)B) + \epsilon\\ &\leq (1+\epsilon) r(\mathcal{I}_e, \mathrm{OPT}^{\epsilon}_{\mathrm{set}}, (1+\epsilon)B),\end{aligned}$$

where the second inequality follows from Lemma 13 and the last one from the fact that the set strategy obtains at least a reward of 1. ◀

▶ **Theorem 12.** *Let $\mathcal{I} = (G, s, C, w)$ be an instance of* SGE*, where $C(e) = O(\frac{\epsilon^2}{\ln n})$ (for each edge e and some $0 < \epsilon = O(1)$), $R \leq \epsilon n^{O(1)}$, and the smallest reward is 1. Then, in polynomial time, we can compute a nonadaptive $(O(1), 1+\epsilon)$-approximate strategy for $\mathcal{I}$. Additionally, if G is a tree, then in time $O(n^3/\epsilon)$ we can compute a nonadaptive $(1+\epsilon, 1+\epsilon)$-approximate strategy for $\mathcal{I}$.*

Proof. By combining Lemma 15 with Lemma 10 we obtain an algorithm which can compute a nonadaptive $(1+\epsilon,(1+\epsilon)^2)$-approximate algorithm in $O(n^3/\epsilon)$ time. By tuning constants, the approximation can be improved to $(1+\epsilon, 1+\epsilon)$.

On the other hand, for general graphs we combine Lemma 15 with the 8-approximate algorithm for the max-prize problem (see Section 4.3) and obtain a $(8(1+\epsilon), 1+\epsilon)$ strategy, which by tuning constants can be improved to $(8+\epsilon, 1+\epsilon)$. ◀

▶ **Lemma 16.** *Assume that for each edge e_i, $i \in [n]$ we have $c_i \in [0,1]$. Then*

$$P_n(3) \geq P_n(1)\left(1 - \ln(P_n(1))\right).$$

Proof. First we show that, for all $j \in [n]$ we have that:

$$\mathbf{Pr}\big(C^n \leq 3 \,\big|\, C^{j-1} \leq 1, C^j > 1\big) \geq \mathbf{Pr}\big(C^n \leq 1 \,\big|\, C^j \leq 1\big). \tag{2}$$

The events $C^j \leq 2$ and $C^n_{j+1} \leq 1$ imply $C^n \leq 3$. The event $C^{j-1} \leq 1$ implies that $C^j \leq 2$ because $c_j \leq 1$. For $i \neq j$, c_i and c_j are independent random variables, thus C^j and C^n_{j+1} are also independent. Using these observations we obtain

$$\begin{aligned}
\mathbf{Pr}\big(C^n \leq 3 \,\big|\, C^{j-1} \leq 1, C^j > 1\big) &\geq \mathbf{Pr}\big(C^j \leq 2, C^r_{j+1} \leq 1 \,\big|\, C^{j-1} \leq 1, C^j > 1\big) \\
&= \mathbf{Pr}\big(C^r_{j+1} \leq 1 \,\big|\, C^{j-1} \leq 1, C^j > 1\big) \\
&= \mathbf{Pr}\big(C^r_{j+1} \leq 1\big) \\
&= \mathbf{Pr}\big(C^n_{j+1} \leq 1\big) \cdot \sum_{t \leq 1} \mathbf{Pr}\big(C^j = t \,\big|\, C^j \leq 1\big) \\
&= \sum_{t \leq 1} \mathbf{Pr}\big(C^n_{j+1} \leq 1\big) \cdot \mathbf{Pr}\big(C^j = t \,\big|\, C^j \leq 1\big) \\
&\geq \sum_{t \leq 1} \mathbf{Pr}\big(C^n_{j+1} \leq 1 - t\big) \cdot \mathbf{Pr}\big(C^j = t \,\big|\, C^j \leq 1\big) \\
&= \sum_{t \leq 1} \mathbf{Pr}\big(C^n_{j+1} \leq 1 - t \,\big|\, C^j \leq 1\big) \cdot \mathbf{Pr}\big(C^j = t \,\big|\, C^j \leq 1\big) \\
&= \mathbf{Pr}\big(C^n \leq 1 \,\big|\, C^j \leq 1\big).
\end{aligned}$$

To simplify the notation in the following, we define $N_1 = \mathbf{Pr}\big(C^1 \leq 1\big)$ and for $r = 2, \ldots, n$, $N_r = \mathbf{Pr}\big(C^n \leq 1 \,\big|\, C^{r-1} \leq 1\big)$.

We will next make use of the following equalities.

$$\begin{aligned}
&\mathbf{Pr}(C^n \leq 1) = \prod_{i=1}^{n} N_i, \\
&\mathbf{Pr}\big(C^n \leq 1 \,\big|\, C^j \leq 1\big) = \prod_{i=j+1}^{n} N_i, \\
&\mathbf{Pr}\big(C^n > 1, C^{r-1} \leq 1\big) = (1 - N_r) \cdot \prod_{i=1}^{r-1} N_i.
\end{aligned}$$

We partition the event $C^n \leq 3$ into disjoint events according to the first index i (if any) such that $C^i > 1$, we get

$$
\begin{aligned}
\mathbf{Pr}(C^n \le 3) &= \mathbf{Pr}(C^n \le 3, C^n \le 1) + \sum_{j=1}^{n} \mathbf{Pr}\big(C^n \le 3, C^j > 1, C^{j-1} \le 1\big) \\
&= \mathbf{Pr}(C^n \le 1) + \sum_{j=1}^{n} \mathbf{Pr}\big(C^n \le 3 \,\big|\, C^j > 1, C^{j-1} \le 1\big) \cdot \mathbf{Pr}\big(C^j > 1, C^{j-1} \le 1\big) \\
&= \mathbf{Pr}(C^n \le 1) + \sum_{j=1}^{n} \left(\mathbf{Pr}\big(C^n \le 3 \,\big|\, C^j > 1, C^{j-1} \le 1\big) \cdot (1 - N_j) \cdot \prod_{i=1}^{j-1} N_i \right) \\
&\ge \mathbf{Pr}(C^n \le 1) + \sum_{j=1}^{n} \left(\mathbf{Pr}\big(C^n \le 1 \,\big|\, C^j \le 1\big) \cdot (1 - N_j) \cdot \prod_{i=1}^{j-1} N_i \right) \\
&= \mathbf{Pr}(C^n \le 1) + \sum_{j=1}^{n} \left(\prod_{i=j+1}^{n} N_i \cdot (1 - N_j) \cdot \prod_{i=1}^{j-1} N_i \right) \\
&= \mathbf{Pr}(C^n \le 1) + \sum_{j=1}^{n} \frac{1 - N_j}{N_j} \cdot \left(\prod_{i=1}^{n} N_i \right) \\
&= \mathbf{Pr}(C^n \le 1) + \sum_{j=1}^{n} \frac{1 - N_j}{N_j} \cdot \mathbf{Pr}(C^n \le 1) = \mathbf{Pr}(C^n \le 1) \cdot \left(1 + \sum_{j=1}^{n} \frac{1 - N_j}{N_j} \right) \\
&\ge \mathbf{Pr}(C^n \le 1) \cdot \left(1 - \sum_{j=1}^{n} \ln N_j \right) = \mathbf{Pr}(C^n \le 1) \cdot \left(1 - \ln \left(\prod_{j=1}^{n} N_j \right) \right) \\
&= \mathbf{Pr}(C^n \le 1) \cdot (1 - \ln(\mathbf{Pr}(C^n \le 1))) \,,
\end{aligned}
$$

where the first inequality follows from Equation (2) and the second from $e^y \ge y + 1$, which for $y = -\ln x$ gives $\frac{1-x}{x} \ge -\ln x$. ◀

▶ **Corollary 17.**

$$
P_n(9) \ge P_n(1) \frac{(1 - \ln(P_n(1)))^2}{2}
$$

Proof. Let for $i \in [n]$ define $\tilde{c}_i = c_i/3$, and define analogously $\tilde{C}^n$. Notice that $\tilde{c}_i \in [0, 1/3] \subset [0, 1]$. We apply Lemma 16 twice (first for $\tilde{C}^n$ and then for C^n and we obtain By rescaling the costs by factor of $1/3$ (denoted with a tilde), we get that $\tilde{c}_i \in [0, 1/3]$, and therefore $\tilde{c}_i \in [0, 1]$ and we apply Lemma 16 we have

$$
\begin{aligned}
P_n(9) &= \mathbf{Pr}(C^n \le 9) \\
&= \mathbf{Pr}\big(\tilde{C}^n \le 3\big) \\
&\ge \mathbf{Pr}\big(\tilde{C}^n \le 1\big) \cdot (1 - \ln(\mathbf{Pr}\big(\tilde{C}^n \le 1\big)) \\
&\ge P_n(3) \cdot (1 - \ln(P_n(3)) \\
&\ge P_n(1) \cdot (1 - \ln(P_n(1))) \cdot (1 - \ln(P_n(1) \cdot (1 - \ln(P_n(1)))) \\
&= P_n(1) \cdot (1 - \ln(P_n(1))) \cdot (1 - \ln(P_n(1)) - \ln(1 - \ln(P_n(1)))) \\
&\ge P_n(1) \cdot \frac{(1 - \ln(P_n(1)))^2}{2},
\end{aligned}
$$

where the last inequality uses the fact that $x/2 \ge \ln(x)$, where we apply $x = 1 - \ln(P_n(1))$. ◀

▶ **Lemma 18.**

$$\max_k\{r(\mathcal{I}, S_k, 9B)\} \geq 0.46 \cdot r(\mathcal{I}, S_{ls}, B).$$

Proof. First note that $\mathbf{Pr}\big(C^j \leq 1\big)$ is a nonincreasing function of j. We partition the edge set into classes $A_0, A_1, \ldots, A_\ell$ such that $j \in A_i$ if $e^{-i-1} < \mathbf{Pr}\big(C^j \leq B\big) \leq e^{-i}$. Let $w(A_i) = \sum_{j\in A_i} w(v_j)$ and $T(r) = \sum_{i=0}^{r} w(A_i)$. Then

$$r(\mathcal{I}, \mathrm{S}_{ls}, B) \leq \sum_{i=1}^{\ell} e^{-i} w(A_i) \leq \sum_{i=1}^{\ell} e^{-i} T(i).$$

Let $k^* = \arg\max_k\{r(\mathcal{I}, S_k, 9B)\}$. For all i, let $a(i) = \max\{j : j \in A_i\}$, and $M = r(\mathcal{I}, S_{k^*}, 9B)$. Then

$$M = r(\mathcal{I}, S_{k^*}, 9B) \geq T(i) \cdot \mathbf{Pr}\Big(C^{a(i)} \leq 9B\Big) \geq T(i) \cdot \frac{(i+2)^2}{2e^{i+1}}.$$

Here, the first inequality follows from the fact that k^* corresponds to the maximum value. For the second inequality, note that $a(i) \in A_i$ and, hence, $\mathbf{Pr}\big(C^{a(i)} \leq B\big) \geq e^{-i-1}$. By Corollary 17, we obtain that $\mathbf{Pr}\big(C^{a(i)} \leq 9B\big) \geq e^{-i-1} \cdot (1 - \ln(e^{-i-1}))^2/2 = \frac{(i+2)^2}{2e^{i+1}}$.

Therefore, we have that $T(i) \leq \frac{2\cdot Me^{1+i}}{(i+2)^2}$, and

$$\begin{aligned} r(\mathcal{I}, \mathrm{S}_{ls}, B) &\leq \sum_{i=1}^{\ell} e^{-i} \cdot T(i) \leq \sum_{i=1}^{\ell} e^{-i} \frac{2 \cdot Me^{i+1}}{(i+2)^2} \leq 2Me \cdot \sum_{i=1}^{\ell} \frac{1}{(i+2)^2} \\ &\leq r(\mathcal{I}, S_{k^*}, 9B)/0.46. \end{aligned}$$

◀

B MineSweeper on Trees

In this section we describe the optimal algorithm for MineSweeper problem, in the case when the input graph is a tree.

B.1 Scheduling with Tree-Like Precedence Constraints

We now introduce and solve a problem of n jobs with tree-like precedence constraints on a single machine. Some special cases of the problem we consider here have been solved before [2, 3, 18] and actually the main idea of the algorithm is the same as in the previous works.

Let A be a set of n *jobs*. An *ordering* is a sequence of length n consisting of distinct elements of A (a permutation of A). We denote by $Ord(A)$ the set of all orderings of A and by $Seq(A)$ we denote the set consisting of all sequences that contain distinct elements of A. Moreover, if X and Y are sequences we use $X \cdot Y$ to denote their concatenation.

The input to the scheduling problem we consider is a tuple (A, n, T, c), where A is a set of n jobs, T is a rooted tree, whose vertex set is A, and $c : Ord(A) \to \mathbb{R}$ is the cost function.

Let $par_T(a_i)$ be the parent of a_i in T. We say that an ordering $a_1, \ldots, a_n$ is *valid* (w.r.t. T) iff a_1 is the root of T and for each $2 \leq i \leq n$ we have that $par_T(a_i) \in \{a_1, \ldots, a_{i-1}\}$. The goal is to compute a valid ordering $a_1, \ldots, a_n$ such that the cost $c(a_1, \ldots, a_n)$ is minimized. We call every such sequence an *optimal ordering*.

Let $S = W \cdot X \cdot Y \cdot Z$ be a valid ordering. We say that X and Y are *swappable* (with respect to S) iff $W \cdot Y \cdot X \cdot Z$ is a valid ordering. Observe that X and Y are swappable when for each $x \in X$ and $y \in Y$, x and y are not in ancestor–descendant (or descendant–ancestor) relation in T.

▶ **Definition 23.** *Consider a scheduling problem (A, n, T, c). Let $W \cdot X \cdot Y \cdot Z \in Ord(A)$. We say that a function $u : Seq(A) \to \mathbb{R}$ is a* utility function *for (A, n, T, c) when*

$$c(W \cdot X \cdot Y \cdot Z) \leq c(W \cdot Y \cdot X \cdot Z) \Leftrightarrow u(X) \geq u(Y).$$

▶ **Corollary 24.** *Let S be an optimal ordering. Assume that $S = W \cdot X \cdot Y \cdot Z$, where X and Y are swappable. Then $u(X) \geq u(Y)$.*

Proof. Assuming $u(X) < u(Y)$ immediately gives a contradiction to the assumption that S is optimal. ◀

In the remaining of this section we show an efficient algorithm, which, given (A, n, T, c) and a utility function, computes an optimal ordering. Note that the previous works only considered concrete cost functions (which, although not stated explicitly, did admit utility functions).

▶ **Lemma 25.** *Let a_1 be a job with a single child a_2 in T. Moreover, assume that $u(a_1) \leq u(a_2)$. Then, a_2 is scheduled immediately after a_1 in some optimal ordering.*

Proof. Let S be an optimal ordering. Clearly, a_1 has to appear before a_2, so S is of the form $S = X \cdot a_1 \cdot Y \cdot a_2 \cdot Z$ for some $X, Y, Z \in Seq(A)$. Since a_2 is the only child of a_1, we know that a_1 and Y are swappable. Similarly, Y and a_2 are swappable.

Thus, the order $S' = X \cdot Y \cdot a_1 \cdot a_2 \cdot Z$ is valid. Moreover, by Corollary 24 applied to S we get $u(Y) \geq u(a_2)$, which implies $u(Y) \geq u(a_1)$. Thus, by Definition 23, $c(S') \leq c(S)$.

Hence, S' is an optimal ordering, in which a_2 is scheduled immediately after a_1. ◀

Consider an algorithm that computes a valid ordering $a_1, \ldots, a_n$ greedily. It runs in n steps. In the ith step it selects a job a_i that is either the root of T or which has the maximum utility $u(a_i)$ among jobs whose parents were already added to the ordering. Note that the resulting ordering is not necessarily unique, as the algorithm may have to choose between jobs of the same utility. We call every ordering that may be produced by the algorithm a *greedy ordering* of A. Let $a \in A$ and let D_a be the set of jobs (vertices) that are *proper* descendants of a in T (hence, $a \notin D_a$). The greedy algorithm described above can also be used to order jobs in D_a, and we extend the definition of greedy orderings to such sets.

▶ **Lemma 26.** *Let $a \in A$ be a job. Assume that for each proper descendant b of a such that $par_T(b) \neq a$ we have $u(b) < u(par_T(b))$. Let S_a be some greedy ordering of the set D_a of proper descendants of a. Then, there exists an optimal ordering of A containing S_a as a subsequence.*

Proof. Let k denote the number of descendants of a in T and let $a'_1, \ldots, a'_k$ be their greedy ordering. Moreover, let $a_1, \ldots, a_k$ be the ordering of all proper descendants of a in some optimal ordering S. Thus, $S = X_1 \cdot a_1 \cdot X_2 \cdot \ldots \cdot X_k \cdot a_k \cdot X_{k+1}$ for some $X_1, \ldots, X_{k+1} \in Seq(A)$. By definition, each descendant of a'_i is some other a'_j, which means that each a'_i is swappable with X_ℓ for $\ell \geq 2$. Therefore, $S' = X_1 \cdot a'_1 \cdot X_2 \cdot \ldots \cdot X_k \cdot a'_k \cdot X_{k+1}$ is a valid ordering. To complete the proof, it remains to show that $c(S') = c(S)$.

First, consider $S = X_1 \cdot a_1 \cdot X_2 \cdot \ldots \cdot X_k \cdot a_k \cdot X_{k+1}$. For $2 \le i \le k$ we have that X_i is swappable with a_i and for each $1 \le i \le k$, a_i is swappable with X_{i+1}. By applying Corollary 24 we conclude that $u(a_i) \ge u(a_{i+1})$ for $1 \le i < k$.

From the assumption that $u(b) < u(par_T(b))$, combined with the algorithm for computing greedy orderings, we also have $u(a'_i) \ge u(a'_{i+1})$. Hence, in both S and S' the jobs are sorted in nonincreasing order of utilities. This means that S' can be obtained from S by permuting jobs that have equal utilities.

Recall that $S' = X_1 \cdot a'_1 \cdot X_2 \cdot \ldots \cdot X_k \cdot a'_k \cdot X_{k+1}$. To complete the proof, we show that if $u(a'_i) = u(a'_{i+1})$, then the two jobs can be swapped and the resulting ordering is still valid and has the same cost. The assumption $u(a'_i) = u(a'_{i+1})$ combined with the fact that $u(a'_i) \ge u(X_{i+1}) \ge u(a'_{i+1})$, yields $u(a'_i) = u(X_{i+1}) = u(a'_{i+1})$.

We now swap a'_i with a'_{i+1} by performing 3 swaps of adjacent elements a'_i, X_{i+1}, and a'_{i+1}. Since the utilities of each these elements are equal, swapping does not influence the cost of the ordering. We only have to show that each time we swap a swappable pair. Clearly, both a'_i and a'_{i+1} can be swapped with X_{i+1}. In addition a'_i and a'_{i+1} can be swapped with each other. The fact that $u(a'_i) = u(a'_{i+1})$ combined with the assumption that $u(b) < u(par_T(b))$ implies that neither of these elements is an ancestor of the other. The lemma follows. ◀

Lemmas 26 and 25 motivate two reductions, which we can apply to our problem, without changing the cost of the optimal solution.

We first describe a *merging reduction*, based on Lemma 26. Let $a \in A$ be a job satisfying the assumptions of Lemma 26. The merging reduction consists in replacing the subtree of a by a path containing the jobs ordered in a greedy way.

▶ **Corollary 27.** *Consider a scheduling problem (A, n, T, c). Let $a \in A$ be a job satisfying the assumptions of Lemma 26 and T_a be the tree obtained from T by performing a merging step on a. Then, every optimal ordering for (A, n, T_a, c) is an optimal ordering for (A, n, T, c).*

Proof. Observe that the precedence constraints defined by T_a can only be stricter than the ones given by T. Thus, every optimal ordering for (A, n, T_a, c) is valid for (A, n, T, c). From Lemma 26 it follows that the costs of optimal orderings in both problems are equal. ◀

Now consider two jobs a_1 and a_2 satisfying the assumptions of Lemma 25. A *contracting reduction* contracts two jobs a_1 and a_2 into one job a_3 of utility $u(a_1 \cdot a_2)$. More generally, we define $u(X \cdot a_3 \cdot Y) := u(X \cdot a_1 \cdot a_2 \cdot Y)$. By Lemma 25 and Corollary 27 both reductions do not change the cost of the optimal ordering. In order to compute the optimal ordering, we apply the reductions repeatedly.

▶ **Lemma 28.** *Consider a job $a \in A$ and the subtree T_a of T rooted in a. If the contracting reduction cannot be applied within T_a, then either T_a imposes a linear order of jobs or the merge reduction can be applied within T_a.*

Proof. Consider the lowest node x of T_a that has at least two children. Note that such node exists if T_a is not a single path (which would impose a linear order on the jobs). From the choice of x we have that each subtree rooted at x is a path. Moreover, since the contracting reduction cannot be applied, we immediately have that the merge reduction can be applied. ◀

It follows that, by applying the reductions, we obtain a tree T' that imposes a linear order of jobs. It remains to show that the final ordering can be computed efficiently. To that end, we need to assume that the utility function can be computed efficiently. In our algorithm, we compute the utility of sequences consisting of a single job and merge jobs during a merge reduction. When this happens, two jobs a_1 and a_2 are replaced with one, which is equivalent to executing a_1 immediately followed by a_2. Jobs a_1 and a_2 are discarded, which implies that this step can be executed at most $n-1$ times, where n is the initial number of jobs. We say that a utility function can be *maintained efficiently* if we can compute the utility of each individual job in $O(1)$ time, including jobs that are created during a merge reduction.

▶ **Theorem 29.** *Let (A, n, T, c) be an instance of the scheduling problem. Assume that there exists a utility function u for (A, n, T, c) that can be maintained efficiently. Then, the optimal ordering for (A, n, T, c) can be computed in $O(n \log n)$ time.*

Proof. The algorithm is a recursive function that, given a node of T, replaces the subtree rooted at T with a path (subtree imposing a linear order of jobs) without affecting the cost of the optimal ordering. It turns out that we can reuse the existing algorithm from [2] that proceeds analogously (the only difference is that it uses some particular utility function). The algorithm is a simple implementation of the recursive function, using leftist trees. As shown in [2] it runs in $O(n \log n)$ time. ◀

B.2 Optimal Algorithm for MineSweeper on Trees

▶ **Theorem 8.** *Consider the instance $\mathcal{I} = (T, s, p, w)$ of the minesweeper problem, where T is a tree. The optimal strategy, OPT_{MS}, for MineSweeper on T can be computed in $O(n \log n)$ time, where n is the number of vertices of T.*

Proof. Observe that the minesweeper problem can be viewed as a job scheduling problem with precedence constraints, where the jobs to be scheduled are nodes of T. In the minesweeper problem, the goal is to produce an ordering of edges, but in the case of trees this is the same as ordering the vertices. Namely, having ordered the vertices, we can produce the ordering of edges by taking the edges connecting each vertex to the parent (and ignoring the first vertex in the sequence).

Let $\{a_i; 1 \le i \le n\}$ be the set of vertices of T, $a_1 = s$ being the root of T. To simplify notation, in the following, for $1 \le i \le n$ we denote by $p(a_i)$ the probability that the edge $(par_T(a_i), a_i)$ materializes (we also set $p(a_1) = 1$). An ordering $a_1 \cdot a_2 \cdot \ldots \cdot a_n$ of vertices of T defines a (nonadaptive) strategy for the minesweeper problem.

For a sequence of vertices $b_1 \cdot \ldots \cdot b_k$ in T define

$$W(b_1, \ldots, b_k) = \sum_{i=1}^{k} \left(w(b_i) \prod_{j=1}^{i} p(b_i) \right).$$

Then notice that $W(a_1 \cdot \ldots \cdot a_n)$ is the expected payoff of the nonadaptive strategy that probes the edges according to order $a_1 \cdot \ldots \cdot a_n$: Executing this strategy we collect a weight $w(a_i)$ iff the process does not stop before reaching a_i, that is, iff all the edges $(par_T(a_j), a_j)$ for $j = 2, \ldots, i$ materialize.

To compute the optimal ordering of vertices of T, we use Theorem 29. To apply it, it suffices to show that the job-scheduling problem we obtain admits a utility function and that the function can be maintained efficiently.

Consider an ordering $S = X_1 \cdot X_2 \cdot X_3 \cdot X_4$ and let P_i be the probability that all parent edges of vertices of X_i materialize. That is, for $X_i = b_1, \ldots, b_k$, $P_i = \prod_{j=1}^{k} p(b_j)$.

Notice that we can write

$$W(S) = W(X_1 \cdot X_2 \cdot X_3 \cdot X_4) = W(X_1) + P_1 W(X_2) + P_1 P_2 W(X_3) + P_1 P_2 P_3 W(X_4).$$

Furthermore, we have that the following inequalities are equivalent:

$$\begin{aligned}
W(X_1 \cdot X_2 \cdot X_3 \cdot X_4) &\le W(X_1 \cdot X_3 \cdot X_2 \cdot X_4) \\
P_1 W(X_2) + P_1 P_2 W(X_3) &\le P_1 W(X_3) + P_1 P_3 W(X_2) \\
W(X_2) + P_2 W(X_3) &\le W(X_3) + P_3 W(X_2) \\
W(X_2)(1 - P_3) &\le W(X_3)(1 - P_2) \\
\frac{W(X_2)}{1 - P_2} &\le \frac{W(X_3)}{1 - P_3}
\end{aligned}$$

Hence, $u(X_i) = W(X_i)/(1 - P_i)$ is a utility function for this problem. Note that if $P_i = 1$, then we can set $u(X_i)$ to be equal to a fixed value M that is larger than any other utility (this value can be the same, regardless of X_i). It is easy to see that this utility function can be maintained efficiently by storing P_i and $W(X_i)$ for each job X_i that we obtain as a result of merging jobs. Thus, we can now apply Theorem 29 to complete the proof. ◀