Approximating Pandora's Box with Correlations

Shuchi Chawla
UT-Austin
shuchi@cs.utexas.edu

Evangelia Gergatsouli
UW-Madison
evagerg@cs.wisc.edu

Jeremy McMahan
UW-Madison
jmcmahan@wisc.edu

Christos Tzamos
UW-Madison & University of Athens
tzamos@wisc.edu

We revisit the classic Pandora's Box (PB) problem under correlated distributions on the box values. Recent work of [CGT+20] obtained constant approximate algorithms for a restricted class of policies for the problem that visit boxes in a fixed order. In this work, we study the complexity of approximating the optimal policy which may adaptively choose which box to visit next based on the values seen so far.

Our main result establishes an approximation-preserving equivalence of PB to the well studied Uniform Decision Tree (UDT) problem from stochastic optimization and a variant of the Min-Sum Set Cover (MSSC_f) problem. For distributions of support m, UDT admits a $\log m$ approximation, and while a constant factor approximation in polynomial time is a long-standing open problem, constant factor approximations are achievable in subexponential time [LLM20]. Our main result implies that the same properties hold for PB and MSSC_f.

We also study the case where the distribution over values is given more succinctly as a mixture of m product distributions. This problem is again related to a noisy variant of the Optimal Decision Tree which is significantly more challenging. We give a constant-factor approximation that runs in time $n^{\tilde{O}(m^2/\varepsilon^2)}$ when the mixture components on every box are either identical or separated in TV distance by ε.

1 Introduction

Many everyday tasks involve making decisions under uncertainty; for example driving to work using the fastest route or buying a house at the best price. Although we don't know how the future outcomes of our current decisions will turn out, we can often use some prior information to facilitate the decision making process. For example, having driven on the possible routes to work before, we know which is usually the busiest one. It is also common in such cases that we can remove part of the uncertainty by paying some additional cost. This type of online decision making in the presence of costly information can be modeled as the so-called PANDORA'S BOX problem, first formalized by Weitzman in [Wei79]. In this problem, the algorithm is given n alternatives called *boxes*, each containing a value from a known distribution. The exact value is not known, but can be revealed at a known *opening* cost specific to the box. The goal of the algorithm is to decide which box to open next and whether to select a value and stop, such that the total *opening cost plus the minimum value revealed* is minimized. In the case of independent distributions on the boxes' values, this problem has a very elegant and simple optimal solution, as described by Weitzman [Wei79]: calculate an index for each box[1], open the boxes in increasing order of index, and stop when the expected gain is worse than the value already obtained.

Weitzman's model makes the crucial assumption that the distributions on the values are independent across boxes. This, however, is not always the case in practice and as it turns out, the simple algorithm of the independent case fails to find the optimal solution under correlated distributions. Generally, the complexity of the PANDORA'S BOX with correlations is not yet well understood. **In this work we develop the first approximately-optimal policies for the Pandora's Box problem with correlated values.**

We consider two standard models of correlation where the distribution over values can be specified explicitly in a succinct manner. In the first, the distribution over values has a small support of size m. In the second the distribution is a mixture of m product distributions, each of which can be specified succinctly. We present approximations for both settings.

A primary challenge in approximating PANDORA'S BOX with correlations is that the optimal solution can be an adaptive policy that determines which box to open depending on the instantiations of values in all of the boxes opened previously. It is not clear that such a policy can even be described succinctly. Furthermore, the choice of which box to open is complicated by the need to balance two desiderata – finding a low value box quickly versus learning information about the values in unopened boxes (a.k.a. the state of the world or realized scenario) quickly. Indeed, the value contained in a box can provide the algorithm with crucial information about other boxes, and inform the choice of which box to open next; an aspect that is completely missing in the independent values setting studied by Weitzman.

Contribution 1: Connection to Decision Tree and a general purpose approximation.

Some aspects of the PANDORA'S BOX problem have been studied separately in other contexts. For example, in the OPTIMAL DECISION TREE problem (DT) [GB09, LLM20], the goal is to identify an unknown hypothesis, out of m possible ones, by performing a sequence of costly tests, whose outcomes depend on the realized hypothesis. This problem has an informational structure similar to that in PANDORA'S BOX. In particular, we can think of every possible joint instantiation of values in boxes as a possible hypothesis, and every opening of a box as a test. The difference between the two problems is that while in OPTIMAL DECISION TREE we want to identify the realized hypothesis

[1]This is a special case of Gittins index [GJ74].

exactly, in Pandora's Box it suffices to terminate the process as soon as we have found a low value box.

Another closely related problem is the Min Sum Set Cover [FLT04], where boxes only have two kinds of values – *acceptable* or *unacceptable* – and the goal is to find an *acceptable* value as quickly as possible. A primary difference relative to Pandora's Box is that unacceptable boxes provide no further information about the values in unopened boxes.

One of the main contributions of our work is to unearth connections between Pandora's Box and the two problems described above. We show that Pandora's Box is essentially equivalent to a special case of Optimal Decision Tree (called Uniform Decision Tree or UDT) where the underlying distribution over hypotheses is uniform – the approximation ratios of these two problems are related within log-log factors. Surprisingly, in contrast, the non-uniform DT appears to be harder than non-uniform Pandora's Box. We relate these two problems by showing that both are in turn related to a new version of Min Sum Set Cover, that we call Min Sum Set Cover with Feedback (MSSC$_f$). These connections are summarized in Figure 1. We can thus build on the rich history and large collection of results on these problems to offer efficient algorithms for Pandora's Box. We obtain a polynomial time $\tilde{O}(\log m)$ approximation for Pandora's Box, where m is the number of distinct value vectors (a.k.a. scenarios) that may arise; as well as constant factor approximations in subexponential time.

$$\underbrace{\mathrm{PB} \xleftrightarrow{\text{Section 4}} \mathrm{UMS}}_{\text{Log-log factors}}\underbrace{\mathrm{SC}_f \xleftrightarrow{\text{Section 5}} \mathrm{UDT}}_{\text{Constant factors}}$$

Figure 1: A summary of our approximation preserving reductions

It is an important open question whether constant factor approximations exist for Uniform Decision Tree: the best known lower-bound on the approximation ratio is 4 while it is known that it is not NP-hard to obtain super-constant approximations under the Exponential Time Hypothesis. The same properties transfer also to Pandora's Box and Min Sum Set Cover with Feedback. Pinning down the tight approximation ratio for any of these problems will directly answer these questions for any other problem in the equivalence class we establish.

The key technical component in our reductions is to find an appropriate stopping rule for Pandora's Box: after opening a few boxes, how should the algorithm determine whether a small enough value has been found or whether further exploration is necessary? We develop an iterative algorithm that in each phase finds an appropriate threshold, with the exploration terminating as soon as a value smaller than the threshold is found, such that there is a constant probability of stopping in each phase. Within each phase then the exploration problem can be solved via a reduction to UDT. The challenge is in defining the stopping thresholds in a manner that allows us to relate the algorithm's total cost to that of the optimal policy.

Contribution 2: Approximation for the mixture of distributions model.

Having established the general purpose reductions between Pandora's Box and DT, we turn to the mixture of product distributions model of correlation. This special case of Pandora's Box interpolates between Weitzman's independent values setting and the fully general correlated values setting. In this setting, we use the term "scenario" to denote the different product distributions in the mixture. The information gathering component of the problem is now about determining which product distribution in the mixture the box values are realized from. Once the algorithm has determined the realized scenario (a.k.a. product distribution), the remaining problem amounts to implementing Weitzman's strategy for that scenario.

We observe that this model of correlation for Pandora's Box is related to the noisy version of DT, where the results of some tests for a given realized hypothesis are not deterministic. One challenge for DT in this setting is that any individual test may give us very little information distinguishing different scenarios, and one needs to combine information across sequences of many tests in order to isolate scenarios. This challenge is inherited by Pandora's Box.

Previous work on noisy DT obtained algorithms whose approximations and runtimes depend on the amount of noise. In contrast, we consider settings where the level of noise is arbitrary, but where the mixtures satisfy a *separability assumption*. In particular, we assume that for any given box, if we consider the marginal distributions of the value in the box under different scenarios, these distributions are either identical or sufficiently different (e.g., at least ε in TV distance) across different scenarios. Under this assumption, we design a constant-factor approximation for Pandora's Box that runs in $n^{\tilde{O}(m^2/\varepsilon^2)}$ (Theorem 6.1), where n is the number of boxes. The formal result and the algorithm is presented in Section 6.

1.1 Related work

The Pandora's Box problem was first introduced by Weitzman in the Economics literature [Wei79]. Since then, there has been a long line of research studying Pandora's Box and its many variants ; non-obligatory inspection [Dov18, BK19, BC22, FLL22], with order constraints [HAKS13, BFLL20], with correlation [CGT+20, GT23], with combinatorial costs [BEFF23], competitive information design [DFH+23], delegated version [BDP22], and finally in an online setting [EHLM19]. Multiple works also study the generalized setting where more information can be obtained for a price [CFG+00, GK01, CJK+15, CHKK15] and in settings with more complex combinatorial constraints [Sin18, GGM06, GN13, ASW16, GNS16, GNS17, GJSS19].

Chawla et al. [CGT+20] were the first to study Pandora's Box with correlated values, but they designed approximations relative to a simpler benchmark, namely the optimal performance achievable using a so-called *Partially Adaptive* strategy that cannot adapt the order in which it opens boxes to the values revealed. In general, optimal strategies can decide both the ordering of the boxes and the stopping time based on the values revealed. [CGT+20] designed an algorithm with performance no more than a constant factor worse than the optimal Partially Adaptive strategy.

In Min Sum Set Cover the line of work was initiated by [FLT04], and continued with improvements and generalizations to more complex constraints by [AGY09, MBMW05, BGK10, SW11].

Optimal decision tree is an old problem studied in a variety of settings ([PKSR02, PD92, GB09, GKR10]), while its most notable application is in active learning settings. It was proven to be NP-Hard by Hyafil and Rivest [HR76]. Since then the problem of finding the best approximation algorithm was an active one [GG74, Lov85, KPB99, Das04, CPR+11, CPRS09, GB09, GNR17, CJLM10, AH12], where finally a greedy $\log m$ for the general case was given by [GB09]. This approximation ratio is proven to be the best possible [CPR+11]. For the case of Uniform decision tree less is known, until recently the best algorithm was the same as the optimal decision tree, and the lower bound was 4 [CPR+11]. The recent work of Li et al. [LLM20] showed that there is an algorithm strictly better than $\log m$ for the uniform decision tree.

The noisy version of optimal decision tree was first studied in [GKR10][2], which gave an algorithm with runtime that depends exponentially on the number of noisy outcomes. Subsequently, Jia et al. in [JNNR19] gave an $(\min(r, h) + \log m)$-approximation algorithm, where r (resp. h) is the

[2]This result is based on a result from [GK11] which turned out to be wrong [NS17]. The correct results are presented in [GK17]

maximum number of different test results per test (resp. scenario) using a reduction to Adaptive Submodular Ranking problem [KNN17]. In the case of large number of noisy outcome they obtain a $\log m$ approximation exploiting the connection to Stochastic Set Cover [LPRY08, INvdZ16].

2 Preliminaries

In this paper we study the connections between three different sequential decision making problems – Optimal Decision Tree, Pandora's Box, and Min Sum Set Cover. We describe these problems formally below.

Optimal Decision Tree

In the Optimal Decision Tree problem (denoted DT) we are given a set $\mathcal{S}$ of m scenarios $s \in \mathcal{S}$, each occurring with (known) probability p_s; and n tests $\mathcal{T} = \{T_i\}_{i\in[n]}$, each with cost 1. Nature picks a scenario $s \in \mathcal{S}$ from the distribution p but this scenario is unknown to the algorithm. The goal of the algorithm is to determine which scenario is realized by running a subset of the tests $\mathcal{T}$. When test T_i is run and the realized scenario is s, the test returns a result $T_i(s) \in \mathbb{R}$.

Output. The output of the algorithm is a decision tree where at each node there is a test that is performed, and the branches are the outcomes of the test. In each of the leaves there is an individual scenario that is the only one consistent with the results of the test in the unique path from the root to this leaf. Observe that there is a single leaf corresponding to each scenario s. We can represent the tree as an *adaptive policy* defined as follows:

Definition 2.1 (Adaptive Policy π)**.** An adaptive policy $\pi : \cup_{X\subseteq\mathcal{T}}\mathbb{R}^X \to \mathcal{T}$ is a function that given a set of tests done so far and their results, returns the next test to be performed.

Objective. For a given decision tree or policy π, let $\text{cost}_s(\pi)$ denote the total cost of all of the tests on the unique path in the tree from the root to the leaf labeled with scenario s. The objective of the algorithm is to find a policy π that minimizes the average cost $\sum_{s\in\mathcal{S}} p_s \text{cost}_s(\pi)$.
We use the term Uniform Decision Tree (UDT) to denote the special case of the problem where $p_s = 1/m$ for all scenarios s.

Pandora's Box

In the Pandora's Box problem we are given n boxes, each with cost $c_i \geq 0$ and value v_i. The values $\{v_i\}_{i\in[n]}$ are distributed according to known distribution $\mathcal{D}$. We assume that $\mathcal{D}$ is an arbitrary correlated distribution over vectors $\{v_i\}_{i\in[n]} \in \mathbb{R}^n$. We call vectors of values *scenarios* and use $s = \{v_i\}_{i\in[n]}$ to denote a possible realization of the scenario. As in DT, nature picks a scenario from the distribution D and this realization is a priori unknown to the algorithm. The goal of the algorithm is to pick a box of small value. The algorithm can observe the values realized in the boxes by opening any box i at its respective costs c_i.

Output. The output of the algorithm is an adaptive policy π for opening boxes and a stopping condition. The policy π takes as input a subset of the boxes and their associated values, and either returns the index of a box $i \in [n]$ to be opened next or stops and selects the minimum value seen so far. That is, $\pi : \cup_{X\subseteq[n]}\mathbb{R}^X \to [n] \cup \{\perp\}$ where $\perp$ denotes stopping.

Objective. For a given policy π, let $\pi(s)$ denote the set of boxes opened by the policy prior to stopping when the realized scenario is s. The objective of the algorithm is to minimize the expected cost of the boxes opened plus the minimum value discovered, where the expectation is taken over all possible realizations of the values in each box.[3] Formally the objective is given by

$$\mathbb{E}_{s\sim\mathcal{D}}\left[\min_{i\in\pi(s)} v_{is} + \sum_{i\in\pi(s)} c_i\right],$$

For simplicity of presentation, from now on we assume that $c_i = 1$ for all boxes, but we show in Section E how to adapt our results to handle non-unit costs, without any loss in the approximation factors.

We use UPB to denote the special case of the problem where the distribution $\mathcal{D}$ is uniform over m scenarios.

Min Sum Set Cover with Feedback

In Min Sum Set Cover, we are given n elements and a collection of m sets $\mathcal{S}$ over them, and a distribution $\mathcal{D}$ over the sets. The output of the algorithm is an ordering π over the elements. The cost of the ordering for a particular set $s \in \mathcal{S}$ is the index of the first element in the ordering that belongs to the set s, that is, $\text{cost}_s(\pi) = \min\{i : \pi(i) \in s\}$. The goal of the algorithm is to minimize the expected cost $\mathbb{E}_{s\sim\mathcal{D}}[\text{cost}_s(\pi)]$.

We define a variant of the Min Sum Set Cover problem, called Min Sum Set Cover with Feedback (MSSC$_f$). As in the original problem, we are given a set of n elements, a collection of m sets $\mathcal{S}$ and a distribution $\mathcal{D}$ over the sets. Nature instantiates a set $s \in \mathcal{S}$ from the distribution $\mathcal{D}$; the realization is unknown to the algorithm. Furthermore, in this variant, each element provides *feedback* to the algorithm when the algorithm "visits" this element; this feedback takes on the value $f_i(s) \in \mathbb{R}$ for element $i \in [n]$ if the realized set is $s \in \mathcal{S}$.

Output. The algorithm once again produces an ordering π over the elements. Observe that the feedback allows the algorithm to adapt its ordering to previously observed values. Accordingly, π is an adaptive policy that maps a subset of the elements and their associated feedback, to the index of another element $i \in [n]$. That is, $\pi : \cup_{X\subseteq[n]}\mathbb{R}^X \to [n]$.

Objective. As before, the cost of the ordering for a particular set $s \in \mathcal{S}$ is the index of the first element in the ordering that belongs to the set s, that is, $\text{cost}_s(\pi) = \min\{i : \pi(i) \in s\}$. The goal of the algorithm is to minimize the expected cost $\mathbb{E}_{s\sim\mathcal{D}}[\text{cost}_s(\pi)]$.

Commonalities and notation

As the reader has observed, we capture the commonalities between the different problems through the use of similar notation. Scenarios in DT correspond to value vectors in PB and to sets in MSSC$_f$; all are denoted by s, lie in the set $\mathcal{S}$, and are drawn by nature from a known joint distribution $\mathcal{D}$. Tests in DT correspond to boxes in PB and elements in MSSC$_f$; we index each by $i \in [n]$. The algorithm for each problem produces an adaptive ordering π over these tests/boxes/elements. Test outcomes $T_i(s)$ in DT correspond to box values $v_i(s)$ in PB and feedback $f_i(s)$ in MSSC$_f$.

[3]In the original version of the problem studied by Weitzman [Wei79] the values are independent across boxes, and the goal is to maximize the value collected minus the costs paid, in contrast to the minimization version we study here.

We will use the terminology and notation across different problems interchangeably in the rest of the paper.

2.1 Modeling Correlation

In this work we study two general ways of modeling the correlation between the values in the boxes.

Explicit Distributions. In this case, $\mathcal{D}$ is a distribution over m *scenarios* where the j'th scenario is realized with probability p_j, for $j \in [m]$. Every scenario corresponds to a fixed and known vector of values contained in each box. Specifically, box i has value $v_{ij} \in \mathbb{R}^+ \cup \{\infty\}$ for scenario j.

Mixture of Distributions. We also consider a more general setting, where $\mathcal{D}$ is a mixture of m product distributions. Specifically, each scenario j is a product distribution; instead of giving a deterministic value for every box i, the result is drawn from distribution $\mathcal{D}_{ij}$. This setting is a generalization of the explicit distributions setting described before.

3 Roadmap of the Reductions and Implications

In Figure 2, we give an overview of all the main technical reductions shown in Sections 4 and 5. An arrow $A \to B$ means that we gave an approximation preserving reduction from problem A to problem B. Therefore an algorithm for B that achieves approximation ratio α gives also an algorithm for A with approximation ratio $O(\alpha)$ (or $O(\alpha \log \alpha)$ in the case of black dashed lines). For the exact guarantees we refer to the formal statement of the respective theorem. The gray lines denote less important claims or trivial reductions (e.g. in the case of A being a subproblem of B).

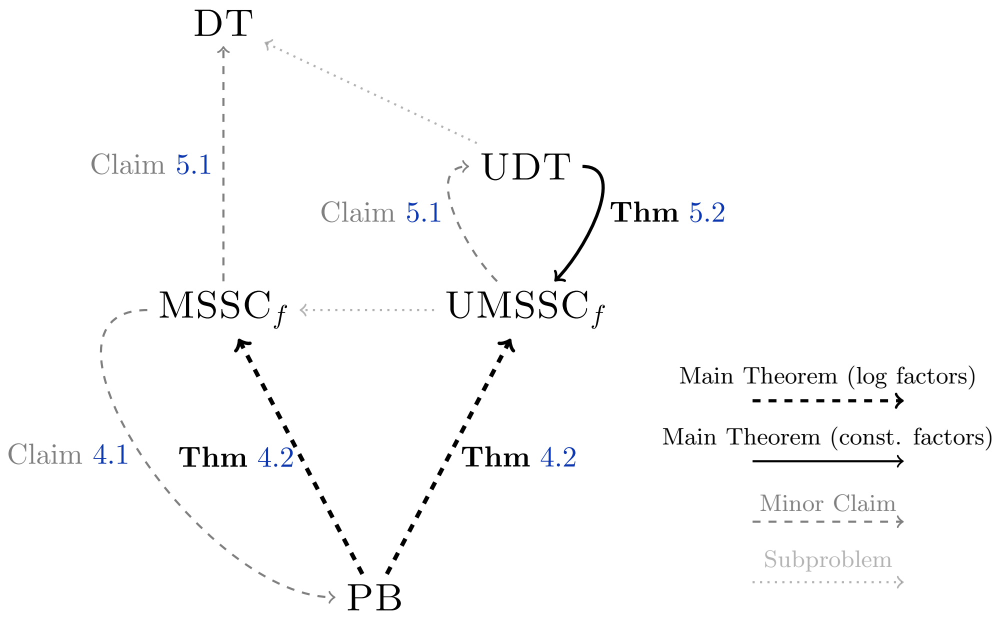

Figure 2: Summary of all our reductions. Bold black lines denote our main theorems, gray dashed are minor claims, and dotted lines are trivial reductions.

3.1 Approximating Pandora's Box

Given our reductions and using the best known results for Uniform Decision Tree from [LLM20] we immediately obtain efficient approximation algorithms for Pandora's Box. We repeat the results of [LLM20] below.

Theorem 3.1 (Theorems 3.1 and 3.2 from [LLM20])**.**

- *There is a $O(\log m / \log \mathrm{OPT})$-approximation algorithm for* UDT *that runs in polynomial time, where OPT is the cost of the optimal solution of the* UDT *instance.*
- *There is a $\frac{9+\varepsilon}{\alpha}$-approximation algorithm for* UDT *that runs in time $n^{\tilde{O}(m^\alpha)}$ for any $\alpha \in (0,1)$.*

Using the results of Theorem 3.1 combined with Theorem 4.2 and Claim 5.1 we get the following corollary.

Corollary 3.1. *From the best-known results for* UDT*, we have that*

- *There is a $\tilde{O}(\log m)$-approximation algorithm for* PB *that runs in polynomial time*[4]*.*
- *There is a $\tilde{O}(1/\alpha)$-approximation algorithm for* PB *that runs in time $n^{\tilde{O}(m^\alpha)}$ for any $\alpha \in (0,1)$.*

An immediate implication of the above corollary is that it is not NP-hard to obtain a superconstant approximation for PB, formally stated below.

Corollary 3.2. *It is not NP-hard to achieve any superconstant approximation for* PB *assuming the Exponential Time Hypothesis.*

Observe that the logarithmic approximation achieved in Corollary 3.1 loses a $\log\log m$ factor (hence the $\tilde{O}$) as it relies on the more complex reduction of Theorem 4.2. If we choose to use the more direct reduction of Theorem A.1 to the Optimal Decision Tree where the tests have non-unit costs (which also admits a $O(\log m)$-approximation [GNR17, KNN17]), we get the following corollary.

Corollary 3.3. *There exists an efficient algorithm that is $O(\log m)$-approximate for* Pandora's Box *and with or without unit-cost boxes.*

3.2 Constant approximation for Partially Adaptive PB

Moving on, we show how our reduction can be used to obtain and improve the results of [CGT+20]. Recall that in [CGT+20] the authors presented a constant factor approximation algorithm against a Partially Adaptive benchmark where the order of opening boxes must be fixed up front.

In such a case, the reduction of Section 4 can be used to reduce PB to the standard Min Sum Set Cover (i.e. without feedback), which admits a 4-approximation [FLT04].

Corollary 3.4. *There exists a polynomial time algorithm for* PB *that is $O(1)$-competitive against the partially adaptive benchmark.*

The same result applies even in the case of non-uniform opening costs. This is because a 4-approximate algorithm for Min Sum Set Cover is known even when elements have arbitrary costs [MBMW05]. The case of non-uniform opening costs has also been considered for Pandora's Box by [CGT+20] but only provide an algorithm to handle polynomially bounded opening costs.

[4]If additionally the possible number of outcomes is a constant K, this gives a $O(\log m)$ approximation without losing an extra logarithmic factor, since $\mathrm{OPT} \geq \log_K m$, as observed by [LLM20].

4 Connecting Pandora's Box and MSSC$_f$

In this section we establish the connection between Pandora's Box and Min Sum Set Cover with Feedback. We show that the two problems are equivalent up to logarithmic factors in approximation ratio.

One direction of this equivalence is easy to see in fact: Min Sum Set Cover with Feedback is a special case of Pandora's Box. Note that in both problems we examine boxes/elements in an adaptive order. In PB we stop when we find a sufficiently small value; in MSSC$_f$ we stop when we find an element that belongs to the instantiated scenario. To establish a formal connection, given an instance of MSSC$_f$, we can define the "value" of each element i in scenario s as being 0 if the element belongs to the set s and as being $L + f_i(s)$ for some sufficiently large value L where $f_i(s)$ is the feedback of element i for set s. This places the instance within the framework of PB and a PB algorithm can be used to solve it. We formally describe this reduction in Section B of the Appendix.

Claim 4.1. *If there exists an $\alpha(n, m)$-approximation algorithm for* PB *then there exists a $\alpha(n, m)$-approximation for* MSSC$_f$*.*

The more interesting direction is a reduction from PB to MSSC$_f$. In fact we show that a general instance of PB can be reduced to the simpler *uniform* version of Min Sum Set Cover with Feedback. We devote the rest of this section to proving the following theorem.

Theorem 4.2 (Pandora's Box to MSSC$_f$)**.** *If there exists an $a(n, m)$ approximation algorithm for* UMSSC$_f$ *then there exists a $O(\alpha(n + m, m^2) \log \alpha(n + m, m^2))$-approximation for* PB*.*

Guessing a stopping rule and an intermediate problem

The feedback structure in PB and MSSC$_f$ is quite similar, and the main component in which the two problems differ is the stopping condition. In MSSC$_f$, an algorithm can stop examining elements as soon as it finds one that "covers" the realized set. In PB, when the algorithm observes a value in a box, it is not immediately apparent whether the value is small enough to stop or whether the algorithm should probe further, especially if the scenario is not fully identified. The key to relating the two problems is to "guess" an appropriate stopping condition for PB, namely an appropriate threshold T such that as soon as the algorithm observes a value smaller than this threshold, it stops. We say that the realized scenario is "covered".

To formalize this approach, we introduce an intermediate problem called Pandora's Box *with costly outside option* T (also called *threshold*), denoted by PB$_{\leq T}$. In this version the objective is to minimize the cost of finding a value $\leq T$, while we have the extra option to quit searching by opening an *outside option* box of cost T. We say that a scenario is *covered* in a given run of the algorithm if it does not choose the outside option box T.

We show that Pandora's Box can be reduced to PB$_{\leq T}$ with a logarithmic loss in approximation factor, and then PB$_{\leq T}$ can be reduced to Min Sum Set Cover with Feedback with a constant factor loss. The following two results capture the details of these reductions.

Claim 4.3. *If there exists an $\alpha(n, m)$ approximation algorithm for* UMSSC$_f$ *then there exists an $3\alpha(n + m, m^2)$-approximation for* UPB$_{\leq T}$*.*

It is also worth noting that PB$_{\leq T}$ is a special case of the Adaptive Ranking problem which directly implies a $\log m$ approximation factor (given in [KNN17]).

Main Lemma 4.4. *Given a polynomial-time $\alpha(n, m)$-approximation algorithm for* $\text{UPB}_{\leq T}$*, there exists a polynomial-time $O(\alpha(n, m) \log \alpha(n, m))$-approximation for* PB*.*

The relationship between $\text{PB}_{\leq T}$ and Min Sum Set Cover with Feedback is relatively straightforward and requires explicitly relating the structure of feedback in the two problems. We describe the details in Section B of the Appendix.

Putting it all together. The proof of Theorem 4.2 follows by combining Claim 4.3 with Lemmas 4.5 and 4.4 presented in the following sections. Proofs of Claims 4.1, 4.3 deferred to Section B of the Appendix. The rest of this section is devoted to proving Lemmas 4.5 and 4.4. The landscape of reductions shown in this section is presented in Figure 3.

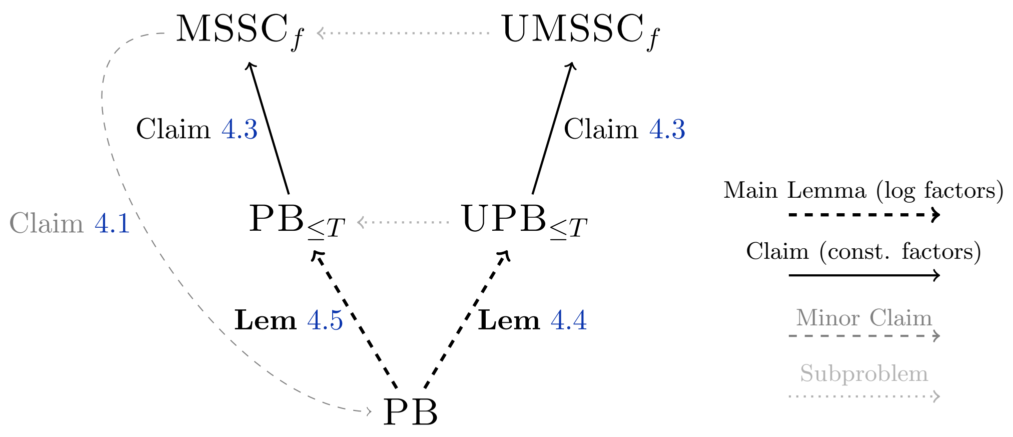

Figure 3: Reductions shown in this section. Claim 4.3 alongside Lemmas 4.5 and 4.4 are part of Theorem 4.2.

4.1 Reducing Pandora's Box to $\text{PB}_{\leq T}$

Recall that a solution to Pandora's Box involves two components ; (1) the order in which to open boxes and (2) a stopping rule. The goal of the reduction to $\text{PB}_{\leq T}$ is to simplify the stopping rule of the problem, by making values either 0 or ∞, therefore allowing us to focus on the order in which boxes are opened, rather than which value to stop at. We start by presenting our main tool, a reduction to Min Sum Set Cover with Feedback in Section 4.1.1 and then improve upon that to reduce from the **uniform** version of MSSC_f (Section 4.1.2).

4.1.1 Main Tool

The high level idea in this reduction is that we repeatedly run the algorithm for $\text{PB}_{\leq T}$ with increasingly larger value of T with the goal of covering some mass of scenarios at every step. The thresholds for every run have to be cleverly chosen to guarantee that enough mass is covered at every run. The distributions on the boxes remain the same, and this reduction does not increase the number of boxes, therefore avoiding the issues faced by the naive reduction given in Section A of the Appendix. Formally, we show the following lemma.

Main Lemma 4.5. *Given a polynomial-time $\alpha(n, m)$-approximation algorithm for* $\text{PB}_{\leq T}$*, there exists a polynomial-time $O(\alpha(n, m) \log \alpha(n, m))$-approximation for* PB*.*

Algorithm 1: Reduction from PB to $\text{PB}_{\leq T}$.

```
Input: Oracle $\mathcal{A}(T)$ for $\text{PB}_{\leq T}$, set of all scenarios $\mathcal{S}$.
1 $i \leftarrow 0$ // Number of current Phase
2 while $\mathcal{S} \neq \emptyset$ do
3     Use $\mathcal{A}$ to find smallest $T_i$ via Binary Search s.t.
        $\mathbf{Pr}$ [accepting the outside option $T_i$] $\leq 0.2$
4     Call the oracle $\mathcal{A}(T_i)$ on set $\mathcal{S}$ to obtain policy $\pi_i$
5     $\mathcal{S} \leftarrow \mathcal{S} \setminus$ {scenarios with total cost $\leq T_i$}
6 end
7 for $i \leftarrow 0$ to $\infty$ do
8     Run policy $\pi_i$ until it terminates and selects a box, or accumulates probing cost $T_i$.
9 end
```

We will now analyze the policy produced by this algorithm.

Proof of Main Lemma 4.5. We start with some notation. Given an instance $\mathcal{I}$ of PB, we repeatedly run $\text{PB}_{\leq T}$ in *phases*. Phase i consists of running $\text{PB}_{\leq T}$ with threshold T_i on a sub instance of the original problem where we are left with a smaller set of scenarios, with their probabilities reweighted to sum to 1. Call this set of scenarios $\mathcal{S}_i$ for phase i and the corresponding instance $\mathcal{I}_i$. After every phase i, we remove the probability mass that was covered[5], and run $\text{PB}_{\leq T}$ on this new instance with a new threshold T_{i+1}. In each phase, the boxes, costs and values remain the same, but the stopping condition changes: thresholds T_i increase in every subsequent phase. The thresholds are chosen such that at the end of each phase, 0.8 of the remaining probability mass is covered. The reduction process is formally shown in Algorithm 1.

Accounting for the cost of the policy. We first note that the total cost of the policy in phase i conditioned on reaching that phase is at most $2T_i$: if the policy terminates in that phase, it selects a box with value at most T_i. Furthermore, the policy incurs probing cost at most T_i in the phase. We can therefore bound the total cost of the policy as $\leq 2\sum_{i=0}^{\infty}(0.2)^i T_i$.

We will now relate the thresholds T_i to the cost of the optimal PB policy for $\mathcal{I}$. To this end, we define corresponding thresholds for the optimal policy that we call *p-thresholds*. Let $\pi^*_{\mathcal{I}}$ denote the optimal PB policy for $\mathcal{I}$ and let c_s denote the cost incurred by $\pi^*_{\mathcal{I}}$ when scenario i is realized. A p-threshold is the minimum possible threshold T such that at most p mass of the scenarios has cost more than T in PB, formally defined below.

Definition 4.6 (p-Threshold)**.** Let $\mathcal{I}$ be an instance of PB and c_s be the cost of scenario $s \in \mathcal{S}$ in $\pi^*_{\mathcal{I}}$, we define the p-threshold as

$$t_p = \min\{T : \mathbf{Pr}[c_s > T] \leq p\}.$$

The following two lemmas relate the cost of the optimal policy to the p-thresholds, and the p-thresholds to the thresholds T_i our algorithm finds. The proofs of both lemmas are deferred to Section B.1 of the Appendix. We first formally define a *sub-instance* of the given Pandora's Box instance.

[5]Recall, a scenario is *covered* if it does not choose the outside option box.

Definition 4.7 (Sub-instance)**.** Let $\mathcal{I}$ be an instance of $\{\text{PB}_{\leq T}, \text{PB}\}$ with set of scenarios $\mathcal{S}_\mathcal{I}$ each with probability $p_s^\mathcal{I}$. For any $q \in [0, 1]$ we call $\mathcal{I}'$ a q-sub instance of $\mathcal{I}$ if $\mathcal{S}_{\mathcal{I}'} \subseteq \mathcal{S}_\mathcal{I}$ and $\sum_{s \in \mathcal{S}_{\mathcal{I}'}} p_s^\mathcal{I} = q$.

Lemma 4.8. *(Optimal Lower Bound) Let $\mathcal{I}$ be the instance of* PB*. For any $q < 1$, any $\alpha > 1$, and $\beta \geq 2$, for the optimal policy $\pi^*_\mathcal{I}$ for* PB *it that*

$$\text{cost}(\pi^*_\mathcal{I}) \geq \sum_{i=0}^{\infty} \frac{1}{\beta\alpha} \cdot (q)^i \, t_{q^i/\beta\alpha}.$$

Lemma 4.9. *Given an instance $\mathcal{I}$ of* PB*; an α-approximation algorithm $\mathcal{A}_T$ to* $\text{PB}_{\leq T}$*; and any $q < 1$ and $\beta \geq 2$, suppose that the threshold T satisfies*

$$T \geq t_{q/(\beta\alpha)} + \beta\alpha \sum_{\substack{c_s \in [t_q, t_{q/(\beta\alpha)}] \\ s \in \mathcal{S}}} c_s \frac{p_s}{q}.$$

Then if $\mathcal{A}_T$ is run on a q-sub instance of $\mathcal{I}$ with threshold T, at most a total mass of $(2/\beta)q$ of the scenarios pick the outside option box T.

Calculating the thresholds. For every phase i we choose a threshold T_i such that $T_i = \min\{T : \mathbf{Pr}\left[c_s > T\right] \leq 0.2\}$ i.e. at most 0.2 of the probability mass of the scenarios are not covered. In order to select this threshold, we do binary search starting from $T = 1$, running every time the α-approximation algorithm for $\text{PB}_{\leq T}$ with outside option box T and checking how many scenarios select it. We denote by $\text{Int}_i = [t_{(0.2)^i}, t_{(0.2)^i/(10\alpha)}]$ the relevant interval of costs at every run of the algorithm, then by Lemma 4.9 for $\beta = 10$, we know that for remaining total probability mass $(0.2)^i$, any threshold which satisfies

$$T_i \geq t_{(0.2)^{i-1}/10a} + 10\alpha \sum_{\substack{s \in \mathcal{S} \\ c_s \in \text{Int}_i}} c_s \frac{p_s}{(0.2)^i}$$

also satisfies the desired covering property, i.e. at least 0.8 mass of the current scenarios is covered. Therefore the threshold

T_i found by our binary search satisfies the following

$$T_i = t_{(0.2)^{i-1}/10a} + 10\alpha \sum_{\substack{s \in \mathcal{S} \\ c_s \in \text{Int}_i}} c_s \frac{p_s}{(0.2)^i}. \tag{1}$$

Bounding the final cost. To bound the final cost, we recall that at the end of every phase we cover 0.8 of the remaining scenarios. Furthermore, we observe that each threshold T_i is charged in the above Equation (1) to optimal costs of scenarios corresponding to intervals of the form $\text{Int}_i = [t_{(0.2)^i}, t_{(0.2)^i/(10\alpha)}]$. Note that these intervals are overlapping. We therefore get

$$\begin{aligned} \text{cost}(\pi_\mathcal{I}) &\leq 2 \sum_{i=0}^{\infty} (0.2)^i T_i \\ &= 2 \sum_{i=0}^{\infty} \left((0.2)^i t_{(0.2)^{i-1}/10a} + 10\alpha \sum_{\substack{s \in \mathcal{S} \\ c_s \in \text{Int}_i}} c_s p_s \right) && \text{From equation (1)} \end{aligned}$$

$$\leq 4 \cdot 10\alpha\pi^*_{\mathcal{I}} + 20\alpha \sum_{i=0}^{\infty} \sum_{\substack{s\in\mathcal{S} \\ c_s\in \text{Int}_i}} c_s p_s \qquad \text{Using Lemma 4.8 for } \beta = 10, q = 0.2$$

$$\leq 40\alpha \log\alpha \cdot \pi^*_{\mathcal{I}}.$$

Where the last inequality follows since each scenario with cost c_s can belong to at most $\log\alpha$ intervals, therefore we get the theorem. □

Notice the generality of this reduction; the distributions on the values are preserved, and we did not make any more assumptions on the scenarios or values throughout the proof. Therefore we can apply this tool regardless of the type of correlation or the way it is given to us, e.g. we could be given a parametric distribution, or an explicitly given distribution, as we see in the next section.

4.1.2 An Even Stronger Tool

Moving one step further, we show that if we instead of $\text{PB}_{\leq T}$ we had an α-approximation algorithm for $\text{UPB}_{\leq T}$ we can obtain the same guarantees as the ones described in Lemma 4.5. Observe that we cannot directly use Algorithm 1 since the oracle now requires that all scenarios have the same probability, while this might not be the case in the initial PB instance. The theorem stated formally follows.

Main Lemma 4.4. *Given a polynomial-time $\alpha(n,m)$-approximation algorithm for* $\text{UPB}_{\leq T}$*, there exists a polynomial-time $O(\alpha(n,m)\log\alpha(n,m))$-approximation for* PB*.*

We are going to highlight the differences with the proof of Main Lemma 4.5, and show how to change Algorithm 1 to work with the new oracle, that requires the scenarios to have uniform probability. The function **Expand** shown in Algorithm 2 is used to transform the instance of scenarios to a uniform one where every scenario has the same probability by creating multiple copies of the more likely scenarios. The function is formally described in Algorithm 3 in Section B.2 of the Appendix, alongside the proof of Main Lemma 4.4.

Algorithm 2: Reduction from PB to $\text{UPB}_{\leq T}$.

```
Input: Oracle $\mathcal{A}(T)$ for $\text{UPB}_{\leq T}$, set of all scenarios $\mathcal{S}$, $c = 1/10, \delta = 0.1$.
1 $i \leftarrow 0$ // Number of current Phase
2 while $\mathcal{S} \neq \emptyset$ do
3     Let $\mathcal{L} = \left\{ s \in \mathcal{S} : p_s \leq c \cdot \frac{1}{|\mathcal{S}|} \right\}$ // Remove low probability scenarios
4     $\mathcal{S}' = \mathcal{S} \setminus \mathcal{L}$
5     $\mathcal{UI} = \text{Expand}(\mathcal{S}')$
6     In instance $\mathcal{UI}$ use $\mathcal{A}$ to find smallest $T_i$ via Binary Search s.t. Pr [accepting $T_i$] $\leq \delta$
7     Call the oracle $\mathcal{A}(T_i)$
8     $\mathcal{S} \leftarrow \left(\mathcal{S}' \setminus \{s \in \mathcal{S}' : c_s \leq T_i\}\right) \cup \mathcal{L}$
9 end
```

5 Connecting MSSC_f and Optimal Decision Tree

In this section we establish the connection between Min Sum Set Cover with Feedback and Optimal Decision Tree. We show that the uniform versions of these problems are equivalent up to constant factors in approximation ratio. The results of this section are summarized in Figure 4 and the two results below.

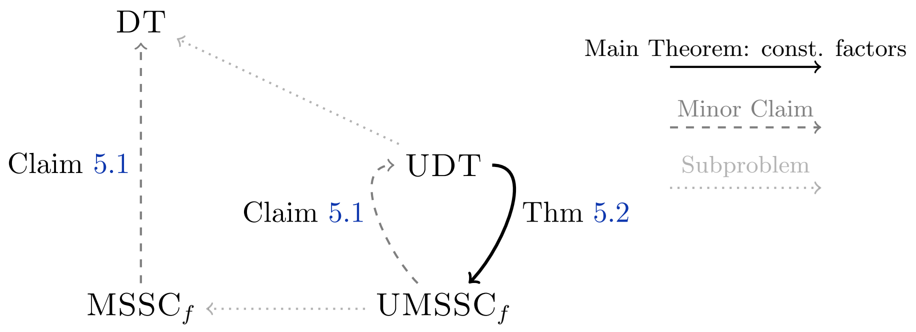

Figure 4: Summary of reductions in Section 5

Claim 5.1. *If there exists an $\alpha(n, m)$-approximation algorithm for* DT *(*UDT*) then there exists a $(1 + \alpha(n, m))$-approximation algorithm for* MSSC$_f$ *(resp.* UMSSC$_f$*).*

Theorem 5.2 (Uniform Decision Tree to UMSSC$_f$)**.** *Given an $\alpha(m, n)$-approximation algorithm for* UMSSC$_f$ *then there exists an $O(\alpha(n + m, m))$-approximation algorithm for* UDT*.*

The formal proofs of these statements can be found in Section C of the Appendix. Here we sketch the main ideas.

One direction of this equivalence is again easy to see. The main difference between Optimal Decision Tree and MSSC$_f$ is that the former requires scenarios to be exactly identified whereas in the latter it suffices to simply find an element that covers the scenario. In particular, in MSSC$_f$ an algorithm could cover a scenario without identifying it by, for example, covering it with an element that covers multiple scenarios. To reduce MSSC$_f$ to DT we simply introduce extra feedback into all of the elements of the MSSC$_f$ instance such that the elements isolate any scenarios they cover. (That is, if the algorithm picks an element that covers some subset of scenarios, this element provides feedback about which of the covered scenarios materialized.) This allows us to relate the cost of isolation and the cost of covering to within the cost of a single additional test, implying Claim 5.1.

Proof Sketch of Theorem 5.2. The other direction is more complicated, as we want to ensure that covering implies isolation. Given an instance of UDT, we create a special element for each scenario which is the unique element covering the scenario and also isolates the scenario from all other scenarios. The intention is that an algorithm for MSSC$_f$ on this new instance only chooses the special isolating element in a scenario after it has identified the scenario. If that happens, then the algorithm's policy is a feasible solution to the UDT instance and incurs no extra cost. The problem is that an algorithm for MSSC$_f$ over the modified instance may use the special covering element before isolating a scenario. We argue that this choice can be "postponed" in the policy to a point at which isolation is nearly achieved without incurring too much extra cost. This involves careful analysis of the policy's decision tree and we present details in the appendix.

Why our reduction does not work for DT. Our analysis above heavily uses the fact that the probabilities of all scenarios in the UDT instance are equal. This is because the "postponement" of elements charges increased costs of some scenarios to costs of other scenarios. In fact, our reduction above fails in the case of non-uniform distributions over scenarios – it can generate an MSSC_f instance with optimal cost much smaller than that of the original DT instance.

To see this, consider an example with m scenarios where scenarios 1 through $m-1$ happen with probability $\varepsilon/(m-1)$ and scenario m happens with probability $1-\varepsilon$. There are $m-1$ tests of cost 1 each. Test i for $i \in [m-1]$ isolates scenario i from all others. Observe that the optimal cost of this DT instance is at least $(1-\varepsilon)(m-1)$ as all $m-1$ tests need to be run to isolate scenario m. Our construction of the MSSC_f instance adds another isolating test for scenario m. A solution to this instance can use this new test at the beginning to identify scenario m and then run other tests with the remaining ε probability. As a result, it incurs cost at most $(1-\varepsilon)+\varepsilon(m-1)$, which is a factor of $1/\varepsilon$ cheaper than that of the original DT instance.

6 Mixture of Product Distributions

In this section we switch gears and consider the case where we are given a mixture of m product distributions. Observe that using the tool described in Section 4.1.1, we can reduce this problem to $\text{PB}_{\leq T}$. This now is equivalent to the noisy version of DT [GK17, JNNR19] where for a specific scenario, the result of each test is not deterministic and can get different values with different probabilities.

Comparison with previous work: previous work on noisy decision tree, considers limited noise models or the runtime and approximation ratio depends on the type of noise. For example in the main result of [JNNR19], the noise outcomes are binary with equal probability. The authors mention that it is possible to extend the following ways:

- to probabilities within $[\delta, 1-\delta]$, incurring an extra $1/\delta$ factor in the approximation
- to non-binary noise outcomes, incurring an extra at most m factor in the approximation

Additionally, their algorithm works by expanding the scenarios for every possible noise outcome (e.g. to 2^m for binary noise). In our work the number of noisy outcomes does not affect the number of scenarios whatsoever.

In our work, we obtain a **constant approximation** factor, that does not depend in any way on the type of the noise. Additionally, the outcomes of the noisy tests can be arbitrary, and do not affect either the approximation factor or the runtime. We only require a *separability* condition to hold ; the distributions either differ *enough* or are exactly the same. Formally, we require that for any two scenarios $s_1, s_2 \in \mathcal{S}$ and for every box i, the distributions $\mathcal{D}_{is_1}$ and $\mathcal{D}_{is_2}$ satisfy $|\mathcal{D}_{is_1} - \mathcal{D}_{is_2}| \in \mathbb{R}_{\geq \varepsilon} \cup \{0\}$, where $|\mathcal{A} - \mathcal{B}|$ is the total variation distance of distributions $\mathcal{A}$ and $\mathcal{B}$.

6.1 A DP Algorithm for noisy $\text{PB}_{\leq T}$

We move on to designing a dynamic programming algorithm to solve the $\text{PB}_{\leq T}$ problem, in the case of a mixtures of product distributions. The guarantees of our dynamic programming algorithm are given in the following theorem.

Theorem 6.1. *For any $\beta > 0$, let π_{DP} and π^* be the policies produced by Algorithm* DP(β) *described by Equation (2) and the optimal policy respectively and* $\text{UB} = \frac{m^2}{\varepsilon^2} \log \frac{m^2 T}{c_{\min}\beta}$. *Then it holds that*

$$c(\pi_{\text{DP}}) \leq (1+\beta)c(\pi^*).$$

and the DP *runs in time n^{UB}, where n is the number of boxes and $c_{\min}$ is the minimum cost box.*

Using the reduction described in Section 4.1.1 and the previous theorem we can get a constant-approximation algorithm for the initial PB problem given a mixture of product distributions. Observe that in the reduction, for every instance of $\text{PB}_{\leq T}$ it runs, the chosen threshold T satisfies that $T \leq (\beta+1)c(\pi_T^*)/0.2$ where π_T^* is the optimal policy for the threshold T. The inequality holds since the algorithm for the threshold T is a $(\beta+1)$ approximation and it covers 80% of the scenarios left (i.e. pays $0.2T$ for the rest). This is formalized in the following corollary.

Corollary 6.1. *Given an instance of* PB *on m scenarios, and the DP algorithm described in Equation (2), then using Algorithm 1 we obtain an $O(1)$-approximation algorithm for* PB *that runs in $n^{\tilde{O}(m^2/\varepsilon^2)}$.*

Observe that the naive DP, that keeps track of all the boxes and possible outcomes, has space exponential in the number of boxes, which can be very large. In our DP, we exploit the separability property of the distributions by distinguishing the boxes in two different types based on a given set of scenarios. Informally, the *informative* boxes help us distinguish between two scenarios, by giving us enough TV distance, while the *non-informative* always have zero TV distance. The formal definition follows.

Definition 6.2 (Informative and non-informative boxes)**.** Let $S \subseteq \mathcal{S}$ be a set of scenarios. Then we call a box k *informative* if there exist $s_i, s_j \in \mathcal{S}$ such that

$$|\mathcal{D}_{ks_i} - \mathcal{D}_{ks_j}| \geq \varepsilon.$$

We denote the set of all *informative* boxes by $\text{IB}(S)$. Similarly, the boxes for which the above does not hold are called *non-informative* and the set of these boxes is denoted by $\text{NIB}(S)$.

Recursive calls of the DP: Our dynamic program chooses at every step one of the following options:

1. open an **informative** box: this step contributes towards *eliminating* improbable scenarios. From the definition of informative boxes, every time such a box is opened, it gives TV distance at least ε between at least two scenarios, making one of them more probable than the other. We show (Lemma 6.3) that it takes a finite amount of these boxes to decide, with high probability, which scenario is the one realized (i.e. eliminating all but one scenarios).

2. open a **non-informative** box: this is a greedy step; the best non-informative box to open next is the one that maximizes the probability of finding a value smaller than T. Given a set S of scenarios that are not yet eliminated, there is a unique next non-informative box which is best. We denote by $\text{NIB}^*(S)$ the function that returns this next best non-informative box. Observe that the non-informative boxes do not affect the greedy ordering of which is the next best, since they do not affect which scenarios are eliminated.

State space of the DP: the DP keeps track of the following three quantities:

1. **a list** M which consists of sets of informative boxes opened and numbers of non-informative ones opened in between the sets of informative ones. Specifically, M has the following form: $M = S_1|x_1|S_2|x_2|\ldots|S_L|x_L$[6] where S_i is a set of informative boxes, and $x_i \in \mathbb{N}$ is the number of non-informative boxes opened exactly after the boxes in set S_i. We also denote by $\text{IB}(M)$ the informative boxes in the list M.

[6]If b_i for $i \in [n]$ are boxes, the list M looks like this: $b_3 b_6 b_{13}|5|b_{42} b_1|6|b_2$

In order to update M at every recursive call, we either append a new informative box b_i opened (denoted by $M|b_i$) or, when a non-informative box is opened, we add 1 at the end, denoted by $M+1$.

2. **a list** E of m^2 tuples of integers (z_{ij}, t_{ij}), one for each pair of distinct scenarios (s_i, s_j) with $i, j \in [m]$. The number z_{ij} keeps track of the number of informative boxes between s_i and s_j that the value discovered had higher probability for scenario s_i, and the number t_{ij} is the total number of informative for scenarios s_i and s_j opened. Every time an informative box is opened, we increase the t_{ij} variables for the scenarios the box was informative and add 1 to the z_{ij} if the value discovered had higher probability in s_i. When a non-informative box is opened, the list remains the same.We denote this update by E^{++}.

3. **a list** S of the scenarios not yet eliminated. Every time an informative test is performed, and the list E updated, if for some scenario s_i there exists another scenario s_j such that $t_{ij} > 1/\varepsilon^2 \log(1/\delta)$ and $|z_{ij} - \mathbb{E}[z_{ij}|s_i]| \le \varepsilon/2$ then s_j is removed from S, otherwise s_i is removed[7]. This update is denoted by S^{++}.

Base cases: if a value below T is found, the algorithm stops. The other base case is when $|S| = 1$, which means that the scenario realized is identified, we either take the outside option T or search the boxes for a value below T, whichever is cheapest. If the scenario is identified correctly, the DP finds the expected optimal for this scenario. We later show that we make a mistake only with low probability, thus increasing the cost only by a constant factor. We denote by $\text{Nat}(\cdot,\cdot,\cdot)$ the "nature's" move, where the value in the box we chose is realized, and $\text{Sol}(\cdot,\cdot,\cdot)$ is the minimum value obtained by opening boxes. The recursive formula is shown below.

$$\text{Sol}(M,E,S) = \begin{cases} \min(T, c_{\text{NIB}^*(S)} + \text{Nat}(M{+}1, E, S)) & \text{if } |S| = 1 \\ \min\Big(T, \min_{i \in \text{IB}(M)} (c_i{+}\text{Nat}(M|i, E, S)) \\ \qquad , c_{\text{NIB}^*(S)} + \text{Nat}(M{+}1, E, S)\Big) & \text{else} \end{cases} \tag{2}$$
$$\text{Nat}(M,E,S) = \begin{cases} 0 & \text{if } v_{\text{last box opened}} \le T \\ \text{Sol}(M, E^{++}, S^{++}) & \text{else} \end{cases}$$

The final solution is $\text{DP}(\beta) = \text{Sol}(\emptyset, E^0, \mathcal{S})$, where E^0 is a list of tuples of the form $(0,0)$, and in order to update S we set $\delta = \beta c_{\min}/(m^2 T)$.

Lemma 6.3. *Let $s_1, s_2 \in \mathcal{S}$ be any two scenarios. Then after opening $\frac{\log(1/\delta)}{\varepsilon^2}$ informative boxes, we can eliminate one scenario with probability at least $1-\delta$.*

We defer the proof of this lemma and Theorem 6.1 to Section D of the Appendix.

References

[AGY09] Yossi Azar, Iftah Gamzu, and Xiaoxin Yin. Multiple intents re-ranking. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 669–678. ACM, 2009.

[7]This is the process of elimination in the proof of Lemma 6.3

[AH12] Micah Adler and Brent Heeringa. Approximating optimal binary decision trees. *Algorithmica*, 62(3-4):1112–1121, 2012.

[ASW16] Marek Adamczyk, Maxim Sviridenko, and Justin Ward. Submodular stochastic probing on matroids. *Math. Oper. Res.*, 41(3):1022–1038, 2016.

[BC22] Hedyeh Beyhaghi and Linda Cai. Pandora's problem with nonobligatory inspection: Optimal structure and a PTAS. *CoRR*, abs/2212.01524, 2022.

[BDP22] Curtis Bechtel, Shaddin Dughmi, and Neel Patel. Delegated pandora's box. In David M. Pennock, Ilya Segal, and Sven Seuken, editors, *EC '22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 - 15, 2022*, pages 666–693. ACM, 2022.

[BEFF23] Ben Berger, Tomer Ezra, Michal Feldman, and Federico Fusco. Pandora's problem with combinatorial cost. *CoRR*, abs/2303.01078, 2023.

[BFLL20] Shant Boodaghians, Federico Fusco, Philip Lazos, and Stefano Leonardi. Pandora's box problem with order constraints. In Péter Biró, Jason D. Hartline, Michael Ostrovsky, and Ariel D. Procaccia, editors, *EC '20: The 21st ACM Conference on Economics and Computation, Virtual Event, Hungary, July 13-17, 2020*, pages 439–458. ACM, 2020.

[BGK10] Nikhil Bansal, Anupam Gupta, and Ravishankar Krishnaswamy. A constant factor approximation algorithm for generalized min-sum set cover. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1539–1545, 2010.

[BK19] Hedyeh Beyhaghi and Robert Kleinberg. Pandora's problem with nonobligatory inspection. In Anna Karlin, Nicole Immorlica, and Ramesh Johari, editors, *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019*, pages 131–132. ACM, 2019.

[CFG$^+$00] Moses Charikar, Ronald Fagin, Venkatesan Guruswami, Jon M. Kleinberg, Prabhakar Raghavan, and Amit Sahai. Query strategies for priced information (extended abstract). In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 582–591, 2000.

[CGT$^+$20] Shuchi Chawla, Evangelia Gergatsouli, Yifeng Teng, Christos Tzamos, and Ruimin Zhang. Pandora's box with correlations: Learning and approximation. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1214–1225. IEEE, 2020.

[CHKK15] Yuxin Chen, S. Hamed Hassani, Amin Karbasi, and Andreas Krause. Sequential information maximization: When is greedy near-optimal? In *Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015*, pages 338–363, 2015.

[CJK$^+$15] Yuxin Chen, Shervin Javdani, Amin Karbasi, J. Andrew Bagnell, Siddhartha S. Srinivasa, and Andreas Krause. Submodular surrogates for value of information. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 3511–3518, 2015.

[CJLM10] Ferdinando Cicalese, Tobias Jacobs, Eduardo Sany Laber, and Marco Molinaro. On greedy algorithms for decision trees. In Otfried Cheong, Kyung-Yong Chwa, and Kunsoo Park, editors, *Algorithms and Computation - 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part II*, volume 6507 of *Lecture Notes in Computer Science*, pages 206–217. Springer, 2010.

[CPR+11] Venkatesan T. Chakaravarthy, Vinayaka Pandit, Sambuddha Roy, Pranjal Awasthi, and Mukesh K. Mohania. Decision trees for entity identification: Approximation algorithms and hardness results. *ACM Trans. Algorithms*, 7(2):15:1–15:22, 2011.

[CPRS09] Venkatesan T. Chakaravarthy, Vinayaka Pandit, Sambuddha Roy, and Yogish Sabharwal. Approximating decision trees with multiway branches. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, volume 5555 of *Lecture Notes in Computer Science*, pages 210–221. Springer, 2009.

[Das04] Sanjoy Dasgupta. Analysis of a greedy active learning strategy. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, pages 337–344, 2004.

[DFH+23] Bolin Ding, Yiding Feng, Chien-Ju Ho, Wei Tang, and Haifeng Xu. Competitive information design for pandora's box. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 353–381. SIAM, 2023.

[Dov18] Laura Doval. Whether or not to open pandora's box. *J. Econ. Theory*, 175:127–158, 2018.

[EHLM19] Hossein Esfandiari, Mohammad Taghi Hajiaghayi, Brendan Lucier, and Michael Mitzenmacher. Online pandora's boxes and bandits. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 1885–1892. AAAI Press, 2019.

[FLL22] Hu Fu, Jiawei Li, and Daogao Liu. Pandora box problem with nonobligatory inspection: Hardness and improved approximation algorithms. *CoRR*, abs/2207.09545, 2022.

[FLT04] Uriel Feige, László Lovász, and Prasad Tetali. Approximating min sum set cover. *Algorithmica*, 40(4):219–234, 2004.

[GB09] Andrew Guillory and Jeff A. Bilmes. Average-case active learning with costs. In Ricard Gavaldà, Gábor Lugosi, Thomas Zeugmann, and Sandra Zilles, editors, *Algorithmic Learning Theory, 20th International Conference, ALT 2009, Porto, Portugal, October 3-5, 2009. Proceedings*, volume 5809 of *Lecture Notes in Computer Science*, pages 141–155. Springer, 2009.

[GG74] M. R. Garey and Ronald L. Graham. Performance bounds on the splitting algorithm for binary testing. *Acta Informatica*, 3:347–355, 1974.

[GGM06] Ashish Goel, Sudipto Guha, and Kamesh Munagala. Asking the right questions: model-driven optimization using probes. In *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*, pages 203–212, 2006.

[GJ74] J.C. Gittins and D.M. Jones. A dynamic allocation index for the sequential design of experiments. *Progress in Statistics*, pages 241–266, 1974.

[GJSS19] Anupam Gupta, Haotian Jiang, Ziv Scully, and Sahil Singla. The markovian price of information. In *Integer Programming and Combinatorial Optimization - 20th International Conference, IPCO 2019, Ann Arbor, MI, USA, May 22-24, 2019, Proceedings*, pages 233–246, 2019.

[GK01] Anupam Gupta and Amit Kumar. Sorting and selection with structured costs. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 416–425, 2001.

[GK11] Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *J. Artif. Intell. Res.*, 42:427–486, 2011.

[GK17] Daniel Golovin and Andreas Krause. Adaptive submodularity: A new approach to active learning and stochastic optimization. *CoRR*, abs/1003.3967, 2017.

[GKR10] Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. In John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta, editors, *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, pages 766–774. Curran Associates, Inc., 2010.

[GN13] Anupam Gupta and Viswanath Nagarajan. A stochastic probing problem with applications. In *Integer Programming and Combinatorial Optimization - 16th International Conference, IPCO 2013, Valparaíso, Chile, March 18-20, 2013. Proceedings*, pages 205–216, 2013.

[GNR17] Anupam Gupta, Viswanath Nagarajan, and R. Ravi. Approximation algorithms for optimal decision trees and adaptive TSP problems. *Math. Oper. Res.*, 42(3):876–896, 2017.

[GNS16] 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 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1731–1747, 2016.

[GNS17] Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Adaptivity gaps for stochastic probing: Submodular and XOS functions. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1688–1702, 2017.

[GT23] Evangelia Gergatsouli and Christos Tzamos. Weitzman's rule for pandora's box with correlations. *CoRR*, abs/2301.13534, 2023.

[HAKS13] Noam Hazon, Yonatan Aumann, Sarit Kraus, and David Sarne. Physical search problems with probabilistic knowledge. *Artif. Intell.*, 196:26–52, 2013.

[HR76] Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is np-complete. *Inf. Process. Lett.*, 5(1):15–17, 1976.

[INvdZ16] Sungjin Im, Viswanath Nagarajan, and Ruben van der Zwaan. Minimum latency submodular cover. *ACM Trans. Algorithms*, 13(1):13:1–13:28, 2016.

[JNNR19] Su Jia, Viswanath Nagarajan, Fatemeh Navidi, and R. Ravi. Optimal decision tree with noisy outcomes. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3298–3308, 2019.

[KNN17] Prabhanjan Kambadur, Viswanath Nagarajan, and Fatemeh Navidi. Adaptive submodular ranking. In *Integer Programming and Combinatorial Optimization - 19th International Conference, IPCO 2017, Waterloo, ON, Canada, June 26-28, 2017, Proceedings*, pages 317–329, 2017.

[KPB99] S. Rao Kosaraju, Teresa M. Przytycka, and Ryan S. Borgstrom. On an optimal split tree problem. In Frank K. H. A. Dehne, Arvind Gupta, Jörg-Rüdiger Sack, and Roberto Tamassia, editors, *Algorithms and Data Structures, 6th International Workshop, WADS '99, Vancouver, British Columbia, Canada, August 11-14, 1999, Proceedings*, volume 1663 of *Lecture Notes in Computer Science*, pages 157–168. Springer, 1999.

[LLM20] Ray Li, Percy Liang, and Stephen Mussmann. A tight analysis of greedy yields subexponential time approximation for uniform decision tree. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 102–121. SIAM, 2020.

[Lov85] Donald W. Loveland. Performance bounds for binary testing with arbitrary weights. *Acta Informatica*, 22(1):101–114, 1985.

[LPRY08] Zhen Liu, Srinivasan Parthasarathy, Anand Ranganathan, and Hao Yang. Near-optimal algorithms for shared filter evaluation in data stream systems. In Jason Tsong-Li Wang, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 133–146. ACM, 2008.

[MBMW05] Kamesh Munagala, Shivnath Babu, Rajeev Motwani, and Jennifer Widom. The pipelined set cover problem. In Thomas Eiter and Leonid Libkin, editors, *Database Theory - ICDT 2005, 10th International Conference, Edinburgh, UK, January 5-7, 2005, Proceedings*, volume 3363 of *Lecture Notes in Computer Science*, pages 83–98. Springer, 2005.

[NS17] Feng Nan and Venkatesh Saligrama. Comments on the proof of adaptive stochastic set cover based on adaptive submodularity and its implications for the group identification problem in "group-based active query selection for rapid diagnosis in time-critical situations". *IEEE Trans. Inf. Theory*, 63(11):7612–7614, 2017.

[PD92] Krishna R. Pattipati and Mahesh Dontamsetty. On a generalized test sequencing problem. *IEEE Trans. Syst. Man Cybern.*, 22(2):392–396, 1992.

[PKSR02] Vili Podgorelec, Peter Kokol, Bruno Stiglic, and Ivan Rozman. Decision trees: An overview and their use in medicine. *Journal of medical systems*, 26:445–63, 11 2002.

[Sin18] Sahil Singla. The price of information in combinatorial optimization. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2523–2532, 2018.

[SW11] Martin Skutella and David P. Williamson. A note on the generalized min-sum set cover problem. *Oper. Res. Lett.*, 39(6):433–436, 2011.

[Wei79] Martin L Weitzman. Optimal Search for the Best Alternative. *Econometrica*, 47(3):641–654, May 1979.

A A Naive Reduction to $\mathrm{PB}_{\leq T}$

In this section we present a straightforward reduction from Pandora's Box to $\mathrm{PB}_{\leq T}$ as an alternative to Theorem 4.2. This reduction has a simpler construction compared to the reduction of Section 4, and does not lose a logarithmic factor in the approximation, it however faces the following issues.

1. It incurs an extra computational cost, since it adds a number of boxes that depends on the size of the values' support.
2. It requires opening costs, which means that the oracle for Pandora's Box with outside option should be able to handle non-unit costs on the boxes, even if the original PB problem had unit-cost boxes.

We denote by $\mathrm{PB}^c_{\leq T}$ the version of Pandora's Box with outside option that has **non-unit** cost boxes, and formally state the guarantees of our naive reduction below.

Theorem A.1. *For n boxes and m scenarios, given an $\alpha(n,m)$-approximation algorithm for $\mathrm{PB}^c_{\leq T}$ for arbitrary T, there exits a $2\alpha(n \cdot |supp(v)|, m)$-approximation for* PB *that runs in polynomial time in the number of scenarios, number of boxes, and the number of values.*

Figure 5 summarizes all the reductions from PB to $\mathrm{PB}_{\leq T}$ and in Table 1 we compare the properties of the naive reduction of this section, to the ones show in Section 4. The main differences are that there is a blow-up in the number of boxes that depends on the support, while losing only constant factors in the approximation.

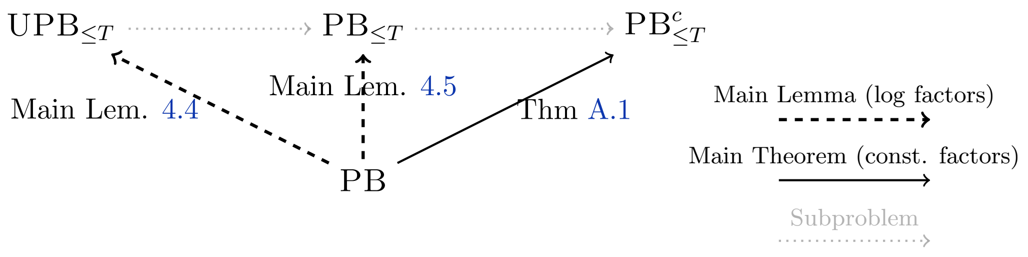

Figure 5: Reductions shown in Section 4.1

	Reducing PB to	
	$\mathrm{PB}^c_{\leq T}$, Theorem A.1	**$(\mathcal{U})\mathrm{PB}_{\leq T}$, Main Lemma 4.5 (4.4)**
Costs of boxes	Introduces non-unit costs	Maintains costs
Probabilities	Maintains probabilities	Maintains probabilities (Makes probabilities uniform)
# of extra scenarios	0	0
# of extra boxes	$n \cdot \mathrm{supp}(v)$	0
Approximation loss	$2\alpha(n \cdot \mathrm{supp}(v), m)$	$O(\alpha(n,m) \log a(n,m))$

Table 1: Differences of reductions of Theorems A.1, and the Main Lemmas 4.5 and 4.4 that comprise Theorem 4.2.

The main idea is that we can move the information about the values contained in the boxes into the cost of the boxes. We do achieve this effect by creating one new box for every (box, value)-pair.

Note, that doing this risks losing the information about the realized scenario that the original boxes revealed. To retain this information, we keep the original boxes, but replace their values by high values. The high values guarantee the effect of the new boxes is retained. Now, we can formalize this intuition.

PB$_{\leq T}$ Instance. Given an instance $\mathcal{I}$ of PB, we construct an instance $\mathcal{I}'$ of PB$_{\leq T}$. We need T to be sufficiently large so that the outside option is never chosen. The net effect is that a policy for PB is easily inferred from a policy for PB$_{\leq T}$. We define the instance $\mathcal{I}'$ to have the same scenarios s_i and same scenario probabilities p_i as $\mathcal{I}$. We choose $T = \infty$[8], and define the new values by $v'_{i,j} = v_{i,j} + T + 1$. Note that all of these values will be larger than T and so a feasible policy cannot terminate after receiving such a value. At the same time, these values ensure the same branching behaviour as before since each distinct value is mapped one to one to a new distinct value. Next, we add additional "final" boxes for each pair (j, v) where j is a box and v a potential value of box j. Each "final" box (j, v) has cost $c_j + v$. Box (j, v) has value 0 for the scenarios where box j gives exactly value v and values $T + 1$ for all other scenarios. Formally,

$$v'_{i,(j,v)} = \begin{cases} 0 & \text{if } v_{i,j} = v \\ T+1 & \text{else} \end{cases}$$

Intuitively, these "final" boxes indicate to a policy that this will be the last box opened, and so its values, which are at least that of the best values of the boxes chosen, should now be taken into account in the cost of the solution.

In order to prove Theorem A.1, we use two key lemmas. In Lemma A.2 we show that the optimal value for the transformed instance $\mathcal{I}'$ of PB$_{\leq T}$ is not much higher than the optimal value for original instance $\mathcal{I}$. In Lemma A.3 we show how to obtain a policy for the initial instance with values, given a policy for the problem with a threshold.

Lemma A.2. *Given the instance $\mathcal{I}$ of* PB *and the constructed instance $\mathcal{I}'$ of* PB$_{\leq T}$ *it holds that*

$$c(\pi^*_{\mathcal{I}'}) \leq 2c(\pi^*_{\mathcal{I}}).$$

Proof. We show that given an optimal policy for PB, we can construct a feasible policy π' for $\mathcal{I}'$ such that $c(\pi_0) \leq 2c(\pi^*_{\mathcal{I}})$. We construct the policy π' by opening the same boxes as π and finally opening the corresponding "values" box, in order to find the 0 needed to stop.

Fix any scenario i, and suppose box j achieved the smallest value $V_{i,j}$ of all boxes opened under scenario i. Since j is opened, in the instance $\mathcal{I}'$ we open box $(j, v_{i,j})$, and from the construction of $\mathcal{I}'$ we have that $v'_{i,(j,v_{i,j})} = 0$. Since on every branch we open a box with values 0[9], we see that π' is a feasible policy for $\mathcal{I}'$. Under scenario i, we have that the cost of $\pi(i)$ is

$$c(\pi(i)) = \min_{k\in\pi(i)} v_{i,k} + \sum_{k\in\pi(i)} c_k.$$

In contrast, the minimum cost following $\pi'(i)$ is 0 and there is the additional cost of the "values" box. Formally, the cost of $\pi'(i)$ is

$$c(\pi'(i)) = 0 + \sum_{k\in\pi(i)} c_k + c_{(j,v_{i,j})} = \min_{k\in\pi(i)} v_{i,k} + \sum_{k\in\pi(i)} c_k + c_j = c(\pi(i)) + c_j$$

[8]We set T to a value larger than $\sum_i c_i + \max_{i,j} v_{ij}$.

[9]π opens at least one box.

Since c_j appears in the cost of $\pi(i)$, we know that $c(\pi(i)) \geq c_j$. Thus, $c(\pi'(i)) = c(\pi(i)) + c_j \leq 2c(\pi(i))$, which implies that $c(\pi') \leq 2c(\pi^*_{\mathcal{I}})$ for our feasible policy π'. Observing that $c(\pi') \geq c(\pi^*_{\mathcal{I}'})$ for any policy, completes the proof. □

Lemma A.3. *Given a policy π' for the constructed instance $\mathcal{I}'$ of* $\text{PB}_{\leq T}$, *there exists a feasible policy π for the instance $\mathcal{I}$ of* PB *with no larger expected cost. Furthermore, any branch of π can be constructed from π' in polynomial time.*

Proof of Lemma A.3. We construct a policy π for $\mathcal{I}$ using the policy π'. Fix some branch of π'. If π' opens box j along this branch, we define policy π to open the same box along this branch. When π' opens a "final" box (j, v), we define the policy π to open box j if it has not been opened already.

Next, we show this policy π has no larger expected cost than π'. There are two cases to consider depending on where the "final" box (j, v) is opened:

1. "Final" box (j, v) is at a leaf of π': since π' has finite expected cost and this is the first "final" box we encountered, the result must be 0. Therefore, under π the values will be v by definition of $\mathcal{I}'$. Observe that in this case, $c(\pi) \leq c(\pi')$ since the (at most) extra v paid by π for the value term, has already been paid by the box cost in π' when box (j, v) was opened.

2. "Final" box (j, v) is at an intermediate node of π': after π opens box j, we copy the subtree of π' that follows the 0 branch into the branch of π that follows the v branch. Also, we copy the subtree of π' that follows the ∞_1 branch into each branch that has a value different from v (the non-v branches). The cost of this new subtree is c_j instead of the original $c_j + v$. The v branch may accrue an additional cost of v or smaller if j was not the smallest values box on this branch, so in total, the v branch has cost at most its original cost.

   However, the non-v branches have a v term removed going down the tree. Specifically, since the feedback of (j, v) down the non-v branch was ∞_1, some other box with 0 values had to be opened at some point, and this box is still available to be used as the final values for this branch later on (since if this branch already had a 0, it would have stopped). Thus, the cost of this subtree is at most that originally, and has one fewer "final" box opened.

Putting these cases together implies that $c(\pi) \leq c(\pi')$.

Lastly, we argue that any branch of π can be computed efficiently. To compute a branch for π, we follow the corresponding branch of π'. As we go along this branch, we open box j whenever π' opens box (j, v) and remember the feedback. We use the feedback to know which boxes of π' to open in the future. Hence, we can compute a branch of π from π' in polynomial time. □

We are now ready to give the proof of Lemma A.1.

Proof of Lemma A.1. Suppose we have an α-approximation for $\text{PB}_{\leq T}$. Given an instance $\mathcal{I}$ to PB, we construct the instance $\mathcal{I}'$ for $\text{PB}_{\leq T}$ as described and then run the approximation algorithm on $\mathcal{I}'$ to get a policy $\pi_{\mathcal{I}'}$. Next, we prune the tree as described in Lemma A.3 to get a policy, $\pi_{\mathcal{I}}$ of no worse cost. Our policy will use time at most polynomially more than the policy for $\text{PB}_{\leq T}$ since each branch of $\pi_{\mathcal{I}}$ can be computed in polynomial time from $\pi_{\mathcal{I}'}$. Hence, the runtime is polynomial in the size of $\mathcal{I}'$. We also note that we added at most mn total "final" boxes to construct our new instance $\mathcal{I}'$, and so this algorithm will run in polynomial time in m and n. Thus, by Lemma A.3 and Lemma A.2 we know the cost of the constructed policy is

$$c(\pi) \leq c(\pi') \leq \alpha c(\pi^*_{\mathcal{I}'}) \leq 2\alpha c(\pi^*_{\mathcal{I}})$$

Hence, this algorithm is a 2α-approximation for PB.

□

B Proofs from Section 4

Claim 4.1. *If there exists an $\alpha(n,m)$-approximation algorithm for* PB *then there exists a $\alpha(n,m)$-approximation for* MSSC_f*.*

Proof of Claim 4.1. Let $\mathcal{I}$ be an instance of MSSC_f. We create an instance $\mathcal{I}'$ of PB the following way: for every set s_j of $\mathcal{I}$ that gives feedback f_{ij} when element e_i is selected, we create a scenario s_j with the same probability and whose value for box i, is either 0 if $e_i \in s_j$ or $\infty_{f_{ij}}$ otherwise, where $\infty_{f_{ij}}$ denotes an extremely large value which is different for different values of the feedback f_{ij}. Observe that any solution to the PB instance gives a solution to the MSSC_f at the same cost and vice versa. □

Claim 4.3. *If there exists an $\alpha(n,m)$ approximation algorithm for* UMSSC_f *then there exists an $3\alpha(n+m,m^2)$-approximation for* $\mathrm{UPB}_{\leq T}$*.*

Before formally proving this claim, recall the correspondence of scenarios and boxes of PB-type problems, to elements and sets of MSSC-type problems. The idea for the reduction is to create T copies of sets for each scenario in the initial $\mathrm{PB}_{\leq T}$ instance and one element per box, where if the price a box gives for a scenario i is $< T$ then the corresponding element belongs to all T copies of the set i. The final step is to "simulate" the outside option T, for which we we create T elements where the k'th one belongs only to the k'th copy of each set.

Proof of Claim 4.3. Given an instance $\mathcal{I}$ of $\mathrm{UPB}_{\leq T}$ with outside cost box b_T, we construct the instance $\mathcal{I}'$ of UMSSC_f as follows.

Construction of the instance. For every scenario s_i in the initial instance, we create T sets denoted by s_{ik} where $k \in [T]$. Each of these sets has equal probability $p_{ik} = 1/(mT)$. We additionally create one element e^B per box B, which belongs to every set s_{ik} for all k iff $v_{Bi} < T$ in the initial instance, otherwise gives feedback v_{Bi}. In order to simulate box b_T without introducing an element with non-unit cost, we use a sequence of T *outside option* elements e^T_k where $e^T_k \in s_{ik}$ for all $i \in [m]$ i.e. element e^T_{ik} belongs to "copy k" of every set [10].

Construction of the policy. We construct policy $\pi_{\mathcal{I}}$ by ignoring any outside option elements that $\pi_{\mathcal{I}'}$ selects until $\pi_{\mathcal{I}'}$ has chosen at least $T/2$ such elements, at which point $\pi_{\mathcal{I}}$ takes the outside option box b_T. To show feasibility we need that for every scenario either b_T is chosen or some box with $v_{ij} \leq T$. If b_T is not chosen, then less than $T/2$ isolating elements were chosen, therefore in instance of UMSSC_f some sub-sets will have to be covered by another element e^B, corresponding to a box. This corresponding box however gives a value $\leq T$ in the initial $\mathrm{UPB}_{\leq T}$ instance.

Approximation ratio. Let s_i be any scenario in $\mathcal{I}$. We distinguish between the following cases, depending on whether there are outside option tests on s_i's branch.

[10]Observe that there are exactly T possible options for k for any set. Choosing all these elements costs T and covers all sets thus simulating b_T.

1. **No outside option tests** on s_i's branch: scenario s_i contributes equally in both policies, since absence of *isolating elements* implies that all copies of scenario s_i will be on the same branch (paying the same cost) in both $\pi_{\mathcal{I}'}$ and $\pi_{\mathcal{I}}$

2. **Some outside option tests** on i's branch: for this case, from Lemma B.1 we have that $c(\pi_{\mathcal{I}}(s_i)) \leq 3c(\pi_{\mathcal{I}'}(s_i))$.

Putting it all together we get

$$c(\pi_{\mathcal{I}}) \leq 3c(\pi_{\mathcal{I}'}) \leq 2\alpha(n+m, m^2)c(\pi^*_{\mathcal{I}'}) \leq 3\alpha(n+m, m^2)c(\pi^*_{\mathcal{I}}),$$

where the second inequality follows since we are given an α approximation and the last inequality since if we are given an optimal policy for UPB$_{\leq T}$, the exact same policy is also feasible for any $\mathcal{I}'$ instance of UDT, which has cost at least $c(\pi^*_{\mathcal{I}'})$. We also used that $T \leq m$, since otherwise the initial policy would never take the outside option. □

Lemma B.1. *Let $\mathcal{I}$ be an instance of* UPB$_{\leq T}$*, and $\mathcal{I}'$ the instance of* UMSSC$_f$ *constructed by the reduction of Claim 4.3. For a scenario s_i, if there is at least one outside option test run in $\pi_{\mathcal{I}}$, then $c(\pi_{\mathcal{I}}(s_i)) \leq 3c(\pi_{\mathcal{I}'}(s_i))$.*

Proof. For the branch of scenario s_i, denote by M the box elements chosen before there were $T/2$ *outside option* elements, and by N the number of *outside option* elements in $\pi_{\mathcal{I}'}$. Note that the smallest cost is achieved if all the outside option elements are chosen first[11]. The copies of scenario s_i can be split into two groups; those that were isolated **before** $T/2$ *outside option* elements were chosen, and those that were isolated **after**. We distinguish between the following cases, based on the value of N.

1. $N \geq T/2$: in this case each of the copies of s_i that are isolated after pays at least $M + T/2$ for the initial box elements and the initial sequence of *outside option* elements. For the copies isolated before, we lower bound the cost by choosing all *outside option* elements first.

   The cost of all the copies in $\pi_{\mathcal{I}'}$ then is at least

$$\begin{aligned}\sum_{j=1}^{K_i}\sum_{k=1}^{T/2}\frac{cp_\ell}{T}k + \sum_{j=1}^{K_i}\sum_{k=T/2+1}^{T}\frac{cp_\ell}{T}(T/2+M) &= cp_i\frac{\frac{T}{2}(\frac{T}{2}+1)}{2T} + cp_i\frac{\frac{T}{2}(T/2+M)}{T}\\ &\geq cp_i(3T/8 + M/2)\\ &\geq \frac{3}{8}p_i(T+M)\end{aligned}$$

   Since $N \geq T/2$, policy $\pi_{\mathcal{I}}$ will take the outside option box for s_i, immediately after choosing the M initial boxes corresponding to the box elements. So, the total contribution s_i has on the expected cost of $\pi_{\mathcal{I}}$ is at most $p_i(M+T)$ in this case. Hence, we have that s_i's contribution in $\pi_{\mathcal{I}}$ is at most $\frac{8}{3} \leq 3$ times s_i's contribution in $\pi_{\mathcal{I}'}$.

2. $N < T/2$: policy $\pi_{\mathcal{I}}$ will only select the M boxes (corresponding to *box* elements) and this was sufficient for finding a value less than T. The total contribution of s_i on $c(\pi_{\mathcal{I}})$ is exactly p_iM. On the other hand, since $N < T/2$ we know that at least half of the copies will pay M for all of the box elements. The cost of all the copies is at least

$$\sum_{j=1}^{K_i}\sum_{k=N}^{T}\frac{cp_\ell}{T}M = cp_i\frac{T-N}{T}M \geq cp_iM/2,$$

[11] Since the outside option tests cause some copies to be isolated and so can reduce their cost.

therefore, the contribution s_i has on $c(\pi_{\mathcal{I}'})$ is at least $cp_iM/2$. Hence, we have $c(\pi_{\mathcal{I}}) \leq 3c(\pi_{\mathcal{I}'})$

□

B.1 Proofs from subsection 4.1.1

Lemma 4.9. *Given an instance $\mathcal{I}$ of* PB*; an α-approximation algorithm $\mathcal{A}_T$ to* $\text{PB}_{\leq T}$*; and any $q < 1$ and $\beta \geq 2$, suppose that the threshold T satisfies*

$$T \geq t_{q/(\beta\alpha)} + \beta\alpha \sum_{\substack{c_s \in [t_q, t_{q/(\beta\alpha)}] \\ s \in \mathcal{S}}} c_s \frac{p_s}{q}.$$

Then if $\mathcal{A}_T$ is run on a q-sub instance of $\mathcal{I}$ with threshold T, at most a total mass of $(2/\beta)q$ of the scenarios pick the outside option box T.

Proof. Consider a policy $\pi_{\mathcal{I}_q}$ which runs $\pi^*_{\mathcal{I}}$ on the instance $\mathcal{I}_q$; and for scenarios with cost $c_s \geq t_{q/(\beta\alpha)}$ aborts after spending this cost and chooses the outside option T. The cost of this policy is:

$$c(\pi^*_{\mathcal{I}_q}) \leq c(\pi_{\mathcal{I}_q}) = \frac{T + t_{q/(\beta\alpha)}}{\beta\alpha} + \sum_{\substack{c_s \in [t_q, t_{q/(10\alpha)}] \\ s \in \mathcal{S}}} c_s \frac{p_s}{q}, \tag{3}$$

By our assumption on T, this cost is at most $2T/\beta\alpha$. On the other hand since $\mathcal{A}_T$ is an α-approximation to the optimal we have that the cost of the algorithm's solution is at most

$$\alpha c(\pi^*_{\mathcal{I}_q}) \leq \frac{2T}{\beta}$$

Since the expected cost of $\mathcal{A}_T$ is at most $2T/\beta$, then using Markov's inequality, we get that $\mathbf{Pr}\left[c_s \geq T\right] \leq (2T/\beta)/T = 2/\beta$. Therefore, $\mathcal{A}_T$ covers at least $1 - 2/\beta$ mass every time. □

Lemma 4.8. *(Optimal Lower Bound) Let $\mathcal{I}$ be the instance of* PB*. For any $q < 1$, any $\alpha > 1$, and $\beta \geq 2$, for the optimal policy $\pi^*_{\mathcal{I}}$ for* PB *it that*

$$\text{cost}(\pi^*_{\mathcal{I}}) \geq \sum_{i=0}^{\infty} \frac{1}{\beta\alpha} \cdot (q)^i \, t_{q^i/\beta\alpha}.$$

Proof. In every interval of the form $\mathcal{I}_i = [t_{q^i}, t_{q^i/(\beta\alpha)}]$ the optimal policy for PB covers at least $1/(\beta\alpha)$ of the probability mass that remains. Since the values belong in the interval $\mathcal{I}_i$ in phase i, it follows that the minimum possible value that the optimal policy might pay is t_{q^i}, i.e. the lower end of the interval. Summing up for all intervals, we get the lemma. □

B.2 Proofs from subsection 4.1.2

Algorithm 3: Expand: rescales and returns an instance of UPB.

```
  Input: Set of scenarios $\mathcal{S}$
1 Scale all probabilities by $c$ such that $c\sum_{s\in\mathcal{S}} p_s = 1$
2 Let $p_{\min} = \min_{s\in\mathcal{S}} p_s$
3 $\mathcal{S}'$ = for each $s \in \mathcal{S}$ create $p_s/p_{\min}$ copies
4 Each copy has probability $1/|\mathcal{S}'|$
5 return $\mathcal{S}'$
```

Main Lemma 4.4. *Given a polynomial-time $\alpha(n,m)$-approximation algorithm for* $\text{UPB}_{\leq T}$*, there exists a polynomial-time $O(\alpha(n,m)\log\alpha(n,m))$-approximation for* PB.

Proof. The proof in this case follows the steps of the proof of Theorem 4.5, and we are only highlighting the changes. The process of the reduction is the same as Algorithm 1 with the only difference that we add two extra steps; (1) we initially remove all low probability scenarios (line 3 - remove at most c fraction) and (2) we add them back after running $\text{UPB}_{\leq T}$ (line 8). The reduction process is formally shown in Algorithm 2.

Calculating the thresholds. For every phase i we choose a threshold T_i such that $T_i = \min\{T : \mathbf{Pr}\left[c_s > T\right] \leq \delta\}$ i.e. at most δ of the probability mass of the scenarios are not covered, again using binary search as in Algorithm 1. We denote by $\text{Int}_i = [t_{(1-c)(\delta+c)^i}, t_{(1-c)(\delta+c)^i/(\beta\alpha)}]$ the relevant interval of costs at every run of the algorithm, then by Lemma 4.9, we know that for remaining total probability mass $(1-c)(\delta+c)^i$, any threshold which satisfies

$$T_i \geq t_{(1-c)(\delta+c)^{i-1}/\beta\alpha} + \beta\alpha \sum_{\substack{s\in\mathcal{S}\\ c_s\in\text{Int}_i}} c_s \frac{p_s}{(1-c)(\delta+c)^i}$$

also satisfies the desired covering property, i.e. at least $(1-2/\beta)(1-c)(\delta+c)$ mass of the current scenarios is covered. Therefore the threshold T_i found by our binary search satisfies

$$T_i = t_{(1-c)(\delta+c)^{i-1}/\beta\alpha} + \beta\alpha \sum_{\substack{s\in\mathcal{S}\\ c_s\in\text{Int}_i}} c_s \frac{p_s}{(1-c)(\delta+c)^i}. \tag{4}$$

Following the proof of Theorem 4.5, **Constructing the final policy** and **Accounting for the values** remain exactly the same as neither of them uses the fact that the scenarios are uniform.

Bounding the final cost. Using the guarantee that at the end of every phase we cover $(\delta+c)$ of the scenarios, observe that the algorithm for $\text{PB}_{\leq T}$ is run in an interval of the form $\text{Int}_i = [t_{(1-c)(\delta+c)^i}, t_{(1-c)(\delta+c)^i/(\beta\alpha)}]$. Note also that these intervals are overlapping. Bounding the cost of the final policy $\pi_{\mathcal{I}}$ for all intervals we get

$$\begin{aligned}
\pi_{\mathcal{I}} &\leq \sum_{i=0}^{\infty}(1-c)(\delta+c)^i T_i \\
&= \sum_{i=0}^{\infty}\left((1-c)(\delta+c)^i t_{(1-c)(\delta+c)^{i-1}/\beta\alpha} + \beta\alpha \sum_{\substack{s\in\mathcal{S}\\ c_s\in\text{Int}_i}} c_s p_s\right) && \text{From equation (4)} \\
&\leq 2\cdot\beta\alpha\pi^*_{\mathcal{I}} + \beta\alpha\sum_{i=0}^{\infty}\sum_{\substack{s\in\mathcal{S}\\ c_s\in\text{Int}_i}} c_s p_s && \text{Using Lemma 4.8} \\
&\leq 2\beta\alpha\log\alpha\cdot\pi^*_{\mathcal{I}},
\end{aligned}$$

where the inequalities follow similarly to the proof of Theorem 4.5. Choosing $c=\delta=0.1$ and $\beta=20$ we get the theorem. □

C Proofs from Section 5

Claim 5.1. *If there exists an $\alpha(n,m)$-approximation algorithm for* DT *(*UDT*) then there exists a $(1+\alpha(n,m))$-approximation algorithm for* MSSC$_f$ *(resp.* UMSSC$_f$*).*

Proof of Claim 5.1. Let $\mathcal{I}$ be an instance of MSSC$_f$. We create an instance $\mathcal{I}'$ of DT the following way: for every set s_j we create a scenario s_j with the same probability and for every element e_i we create a test T_{e_i} with the same cost, that gives full feedback whenever an element belongs to a set, otherwise returns only the element's feedback f_{ij}. Formally, the i-test under scenario s_j returns

$$T_{e_i}(s_j) = \begin{cases} \text{"The feedback is } f_{ij}\text{"} & \text{If } e_i \notin s_j \\ \text{"The scenario is } j\text{"} & \text{else ,} \end{cases}$$

therefore the test isolates scenario j when $e_i \in s_j$.

Constructing the policy. Given a policy π' for the instance $\mathcal{I}'$ of DT, we can construct a policy π for $\mathcal{I}$ by selecting the element that corresponds to the test π' chose. When π' finishes, all scenarios are identified and for any scenario s_j either (1) there is a test in π' that corresponds to an element in s_j (in the instance $\mathcal{I}$) or (2) there is no such test, but we can pay an extra $\min_{i\in s_j} c_i$ to select the lowest cost element in this set[12].

Observe also that in this instance of DT if we were given the optimal solution, it directly translates to *a* solution for MSSC$_f$ with the same cost, therefore

$$c(\pi^*_{\mathcal{I}}) \le c(\pi'_{\mathcal{I}'}) = c(\pi^*_{\mathcal{I}'}) \tag{5}$$

Bounding the cost of the policy. As we described above the total cost of the policy is

$$\begin{aligned} c(\pi) &\le c(\pi_{\mathcal{I}'}) + \mathbb{E}_{s\in\mathcal{S}}\left[\min_{i\in s} c_i\right] \\ &\le c(\pi_{\mathcal{I}'}) + c(\pi^*_{\mathcal{I}}) \\ &\le a(n,m)c(\pi^*_{\mathcal{I}'}) + c(\pi^*_{\mathcal{I}}) \\ &= (1+a(n,m))c(\pi^*_{\mathcal{I}}), \end{aligned}$$

where in the last inequality we used equation (5).

Note that for this reduction we did not change the probabilities of the scenarios, therefore if we had started with uniform probabilities and had an oracle to UDT, we would still get an $a(n,m)+1$ algorithm for UMSSC$_f$. □

In the reduction proof of Theorem 5.2, we are using the following two lemmas, that show that the policy constructed for UDT via the reduction is feasible and has bounded cost.

Lemma C.1. *Given an instance $\mathcal{I}'$ of* UDT *and the corresponding instance $\mathcal{I}$ of* UMSSC$_f$ *in the reduction of Theorem 5.2, the policy $\pi_{\mathcal{I}'}$ constructed for* UDT *is feasible.*

Proof of Lemma C.1. It suffices to show that every scenario is isolated. Fix a scenario s_i. Observe that s_i's branch has chosen the isolating element E^i in the UMSSC$_f$ solution, since that is the the only element that belongs to set s_i. Let S be the set of scenarios just before E^i is chosen and note that by definition $s_i \in S$.

[12]Since the scenario is identified, we know exactly which element this is.

If $|S| = 1$, then since $\pi_{\mathcal{I}'}$ runs tests giving the same branching behavior by definition of $\pi_{\mathcal{I}'}$, and s_i is the only scenario left, we have that the branch of $\pi_{\mathcal{I}'}$ isolates scenario s_i.

If $|S| > 1$ then all scenarios/sets in $S \setminus \{s_i\}$ are not covered by choosing element E^i, therefore they are covered at strictly deeper leaves in the tree. By induction on the depth of the tree, we can assume that for each scenario $s_j \in (S \setminus \{s_i\})$ is isolated in $\pi_{\mathcal{I}'}$. We distinguish the following cases based on when we encounter E^i among the isolating elements on s_i's branch.

1. **E^i was the first isolating element chosen on the branch**: then policy $\pi_{\mathcal{I}'}$ ignores element E^i. Since every leaf holds a unique scenario in $S \setminus \{s_i\}$, if we ignore s_i it follows some path of tests and either be isolated or end up in a node that originally would have had only one scenario, as shown in Figure 6. Since there are only two scenarios at that node, policy $\pi_{\mathcal{I}'}$ runs the cheapest test distinguishing s_i from that scenario.

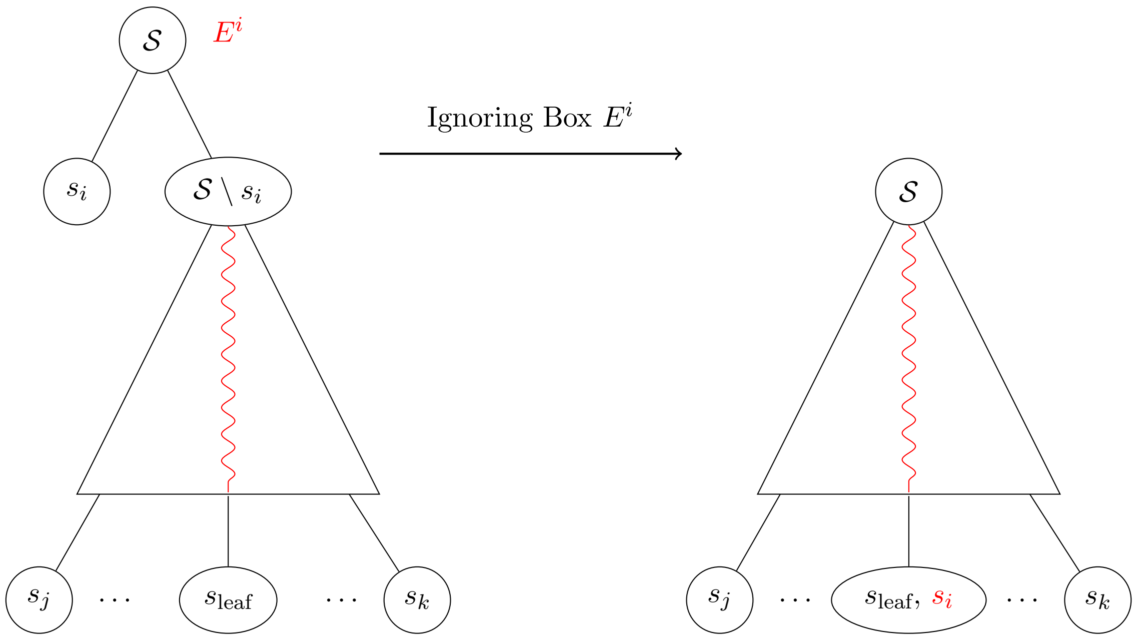

Figure 6: Case 1: $\mathcal{S}$ is the set of scenarios remaining when E^i is chosen, s_{leaf} is the scenario that s_i ends up with.

2. **A different element E^j was chosen before E^i**: by our construction, instead of ignoring E^i we now run the cheapest test that distinguishes s_i from s_j, causing i and j to go down separate branches, as shown in figure 7. We apply the induction hypothesis again to the scenarios in these sub-branches, therefore, both s_i and s_j are either isolated or end up in a node with a single scenario and then get distinguished by the last case of $\pi_{\mathcal{I}'}$'s construction.

Hence, $\pi_{\mathcal{I}'}$ is isolating for any scenario s_i. Also, notice that any two scenarios that have isolating boxes on the same branch will end up in distinct subtrees of the lower node. □

Lemma C.2. *Given an instance $\mathcal{I}$ of* UMSSC$_f$ *and an instance $\mathcal{I}'$ of* UDT*, in the reduction of Theorem 5.2 it holds that*

$$c(\pi_{\mathcal{I}'}) \leq 2c(\pi_{\mathcal{I}}).$$

Proof of Lemma C.2. Let s_i be any scenario in $\mathcal{S}$. We use induction on the number of isolating boxes along s_i's branch in $\mathcal{I}'$. Initially observe that E^i will always exist in s_i's branch, in any

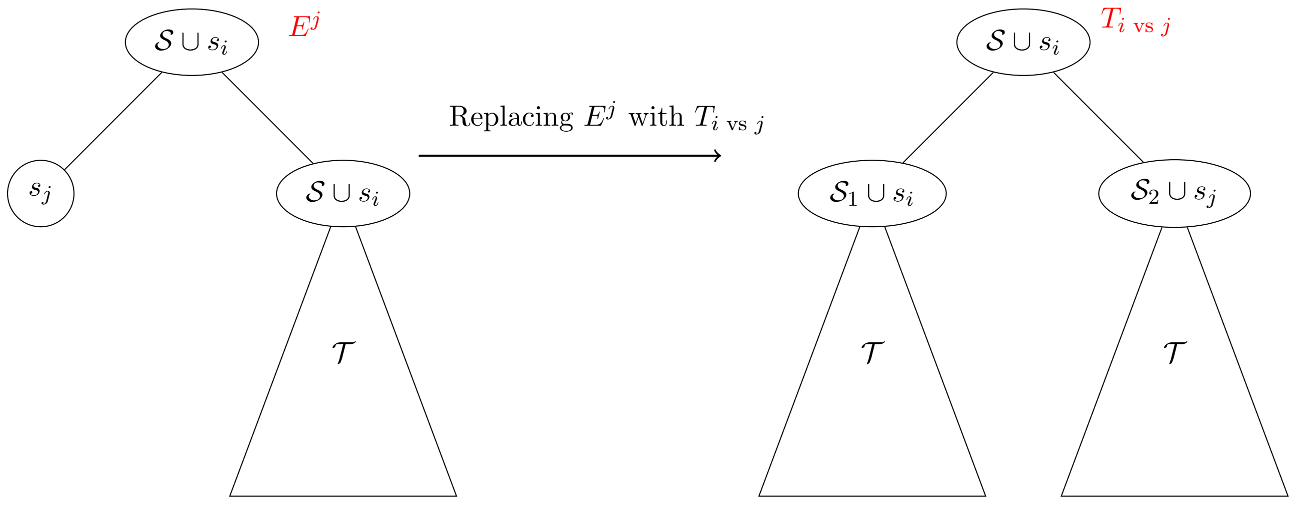

Figure 7: Case 2: run test $T_{i \text{ vs } j}$ to distinguish s_i and s_j. Sets $\mathcal{S}_1$ and $\mathcal{S}_2$ partition $\mathcal{S}$

feasible solution to $\mathcal{I}$. We use $c(E^j)$ and $c(T_k)$ to denote the costs of box E^j and test T_k, for any $k \in [n]$ and $j \in [n+m]$.

1. **Only E^i is on the branch**: since E^i will be ignored, we end up with s_i and some other not yet isolated scenario, let s_{leaf} be that scenario. To isolate s_i and s_{leaf} we run the cheapest test that distinguishes between these. From the definition of the cost of E^i we know that $c(T_{s_i \text{ vs } s_{\text{leaf}}}) \leq c(E^i)$. Additionally, since $c(s_i) \leq c(s_{\text{leaf}})$ and both s_{leaf} and s_i have probability $1/m$, overall we have $c(\pi_{\mathcal{I}}) \leq 2c(\pi_{\mathcal{I}'})$. This is also shown in Figure 6.

2. **More than one isolating elements are on the branch**: similarly, observe that for any extra isolating element E^j we encounter, we substitute it with a test that distinguishes between s_i and s_j and costs at most $c(E^j)$. Given that $c(s_i) \leq c(s_{\text{leaf}})$ and scenarios are uniform, we again have $c(\pi_{\mathcal{I}}) \leq 2c(\pi_{\mathcal{I}'})$.

□

Theorem 5.2 (Uniform Decision Tree to UMSSC$_f$). *Given an $\alpha(m,n)$-approximation algorithm for* UMSSC$_f$ *then there exists an $O(\alpha(n+m,m))$-approximation algorithm for* UDT.

Proof of Theorem 5.2. We begin by giving the construction of the policy in the reduction, and showing the final approximation ratio.

Constructing the policy. Given a policy $\pi_{\mathcal{I}}$ for the instance of UMSSC$_f$, we construct a policy $\pi_{\mathcal{I}'}$. For any test element B_j that $\pi_{\mathcal{I}}$ selects, $\pi_{\mathcal{I}'}$ runs the equivalent test T_j. For the *isolating elements* E^i we distinguish the following cases.

1. If $\pi_{\mathcal{I}}$ selects an *isolating element* E^i for the first time on the current branch, then $\pi_{\mathcal{I}'}$ ignores this element but remembers the set/scenario s_i, which E^i belonged to.

2. If $\pi_{\mathcal{I}}$ selects another *isolating element* E^j after some E^i on the branch, then $\pi_{\mathcal{I}'}$ runs the minimum cost test that distinguishes scenario s_j from s_k where E^k was the most recent *isolating element* chosen on this branch prior to E^j.

3. If we are at the end of $\pi_{\mathcal{I}}$, there can be at most 2 scenarios remaining on the branch, so $\pi_{\mathcal{I}'}$ runs the minimum cost test that distinguishes these two scenarios.

By Lemma C.1, we have that the above policy is feasible for UDT.

Approximation ratio. From Lemma C.2 we have that $c(\pi_{\mathcal{I}'}) \le 2c(\pi_{\mathcal{I}})$. For the optimal policy, we have that $c(\pi^*_{\mathcal{I}}) \le 3c(\pi^*_{\mathcal{I}'})$. This holds since if we have an optimal solution to UDT, we can add an *isolating element* at every leaf to make it feasible for UMSSC$_f$, by only increasing the cost by a factor of 3[13], which means that $c(\pi^*_{\mathcal{I}})$ will be less than this transformed UMSSC$_f$ solution. Overall, if $\pi_{\mathcal{I}}$ is computed from an $\alpha(n,m)$-approximation for UMSSC$_f$, we have

$$c(\pi_{\mathcal{I}'}) \le 2c(\pi_{\mathcal{I}}) \le 2\alpha(n+m,m)c(\pi^*_{\mathcal{I}}) \le 6\alpha(n+m,m)c(\pi^*_{\mathcal{I}'})$$

□

D Proofs from Section 6

Lemma 6.3. *Let $s_1, s_2 \in \mathcal{S}$ be any two scenarios. Then after opening $\frac{\log(1/\delta)}{\varepsilon^2}$ informative boxes, we can eliminate one scenario with probability at least $1-\delta$.*

Proof. Let $s_1, s_2 \in \mathcal{S}$ be any two scenarios in the instance of PB and let v_i be the value returned by opening the i'th informative box, which has distributions $\mathcal{D}_{is_1}$ and $\mathcal{D}_{is_2}$ for scenarios s_1 and s_2 respectively. Then by the definition of informative boxes for every such box opened, there is a set of values v for which $\mathbf{Pr}_{\mathcal{D}_{is_1}}[v] \ge \mathbf{Pr}_{\mathcal{D}_{is_2}}[v]$ and a set for which the reverse holds. Denote these sets by $M_i^{s_1}$ and $M_i^{s_2}$ respectively. We also define the indicator variables $X_i^{s_1} = \mathbb{1}\{v_i \in M_i^{s_1}\}$. Define $\overline{X} = \sum_{i\in[k]} X_i^{s_1}/k$, and observe that $\mathbb{E}\left[\overline{X}|s_1\right] = \sum_{i\in[k]} \mathbf{Pr}\left[M_i^{s_1}\right]/k$. Since for every box we have an ε gap in TV distance between the scenarios s_1, s_2 we have that

$$\left|\mathbb{E}\left[\overline{X}|s_1\right] - \mathbb{E}\left[\overline{X}|s_2\right]\right| \ge \varepsilon,$$

therefore if $\left|\overline{X} - \mathbb{E}\left[\overline{X}|s_1\right]\right| \le \varepsilon/2$ we conclude that scenario s_2 is eliminated, otherwise we eliminate scenario s_1. The probability of error is $\mathbf{Pr}_{\mathcal{D}_{is_1}}\left[\overline{X} - \mathbb{E}\left[\overline{X}|s_1\right] > \varepsilon/2\right] \le e^{-2k(\varepsilon/2)^2}$, where we used Hoeffding's inequality since $X_i \in \{0,1\}$. Since we want the probability of error to be less than δ, we need to open $O\left(\frac{\log 1/\delta}{\varepsilon^2}\right)$ informative boxes. □

Proof of Theorem 6.1. We describe how to bound the final cost, and calculate the runtime of the DP. Denote by $L = m^2/\varepsilon^2 \log 1/\delta$ where we show that in order to get $(1+\beta)$-approximation we set $\delta = \frac{\beta c_{\min}}{m^2 T}$.

Cost of the final solution. Observe that the only case where the DP limits the search space is when $|S| = 1$. If the scenario is identified correctly, the DP finds the optimal solution by running the greedy order; every time choosing the box with the highest probability of a value below T[14].

In order to eliminate all scenarios but one, we should eliminate all but one of the m^2 pairs in the list E. From Lemma 6.3, and a union bound on all m^2 pairs, the probability of the last scenario being the wrong one is at most $m^2\delta$. By setting $\delta = \beta c_{\min}/(m^2 T)$, we get that the probability of error is at most $\beta c_{\min}/T$, in which case we pay at most T, therefore getting an extra $\beta c_{\min} \le \beta c(\pi^*)$ factor.

[13]This is because for every two scenarios, the UDT solution must distinguish between them, but one of these scenarios is the max scenario from the definition of T_j, for which we pay less than T_j

[14]When there is only one scenario, this is exactly Weitzman's algorithm.

Runtime. The DP maintains a list M of sets of informative boxes opened, and numbers of non informative ones. Recall that M has the following form $M = S_1|x_1|S_2|x_2|\dots|S_k|x_k$, where $k \leq L$ from Lemma 6.3 and the fact that there are m^2 pairs in E. There are in total n boxes, and L "positions" for them, therefore the size of the state space is $\binom{n}{L} = O(n^L)$. There is also an extra n factor for searching in the list of informative boxes at every step of the recursion. Observe that the numbers of non-informative boxes also add a factor of at most n in the state space. The list E adds another factor at most n^{m^2}, and the list S a factor of 2^m making the total runtime to be $n^{\tilde{O}(m^2/\varepsilon^2)}$. □

E Boxes with Non-Unit Costs: Revisiting our Results

In the original Pandora's Box problem, denoted by PB^c, each box i has a different known cost $c_i > 0$. Similarly we denote the non-unit cost version of both decision tree-like problems and Min Sum Set Cover-like problems by adding a superscript ^{c} to the problem name. Specifically, we now define DT^c, UDT^c, MSSC$_f^c$ and UMSSC$_f^c$, where the tests (elements) have non-unit cost for the decision tree (min sum set cover) problems. We revisit our results and describe how our reductions change to incorporate non-unit cost boxes (summary in Figure 8).

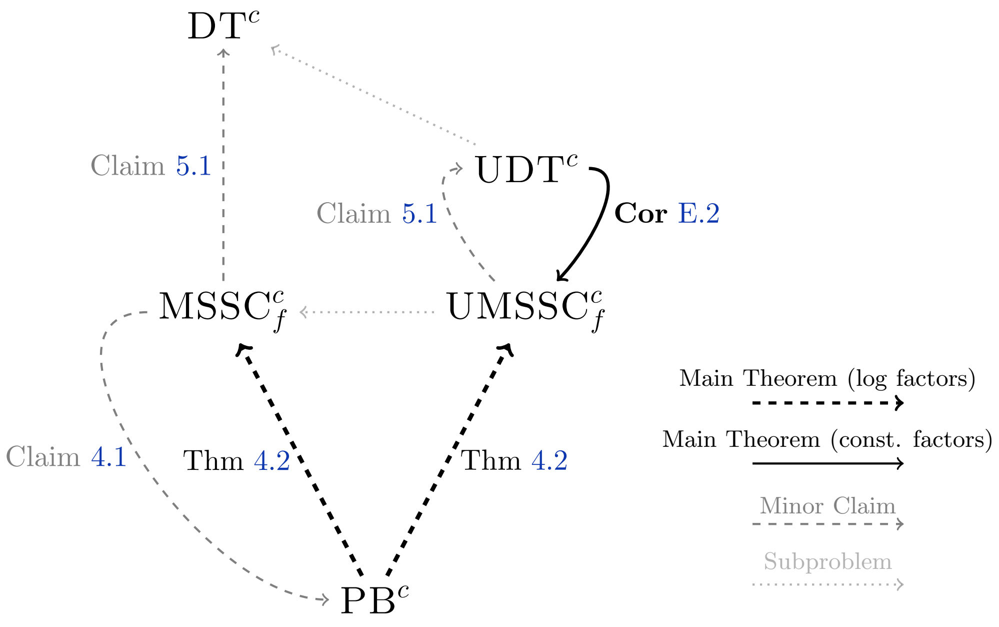

Figure 8: Summary of all the reductions with non-unit costs. The only result that needs a changed proof is Corollary E.2 highlighted in bold (previously Theorem 5.2).

Note also, that even though the known results for Optimal Decision Tree (e.g. [GB09, GNR17]) handle non-unit test costs, the currently known works for Uniform Decision Tree do not. If however there is an algorithm for Uniform Decision Tree with non-unit costs, our reductions will handle this obtaining the same approximation guarantees.

E.1 Connecting Pandora's Box and MSSC$_f$

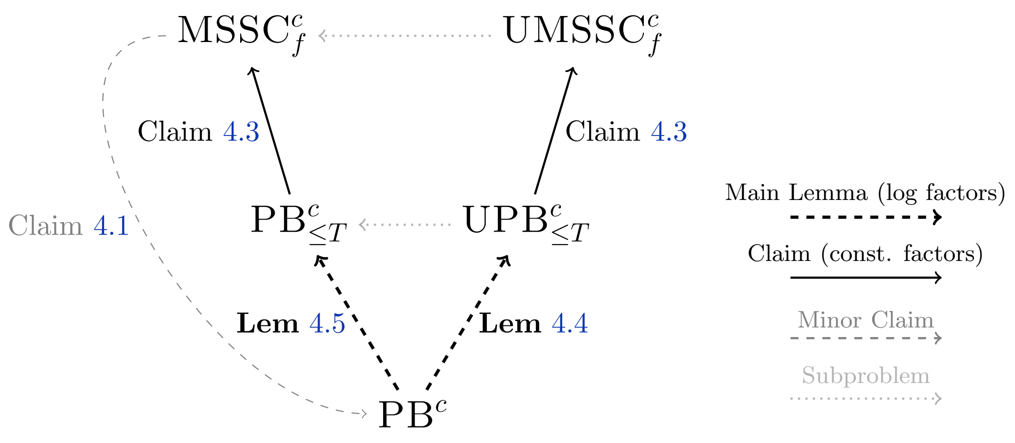

Figure 9: Reductions shown in this section. The solid lines are part of Corollary E.1.

All the results of this section hold as they are when we change all versions to incorporate costs. We did not use the fact that the costs are unit in any of the proofs of Claim 4.1, Claim 4.3 or Lemmas 4.5, 4.4. We formally restate the main theorem of Section 4 as the following corollary, where the only change is that it now holds for the cost versions of the problems.

Corollary E.1 (Pandora's Box to MSSC$_f$ with non-unit costs)**.** *If there exists an $a(n, m)$ approximation algorithm for* MSSC_f^c *then there exists a $O(\alpha(n+m, m^2) \log \alpha(n+m, m^2))$-approximation for* PB^c*. The same result holds if the initial algorithm is for* UMSSC_f^c*.*

E.2 Connecting MSSC$_f$ and Optimal Decision Tree

In this section the reduction of Theorem 5.2 uses the fact that the costs are uniform. However we can easily circumvent this and obtain corollary E.2. Using this, the results for the non-unit costs versions are summarized in Figure 10.

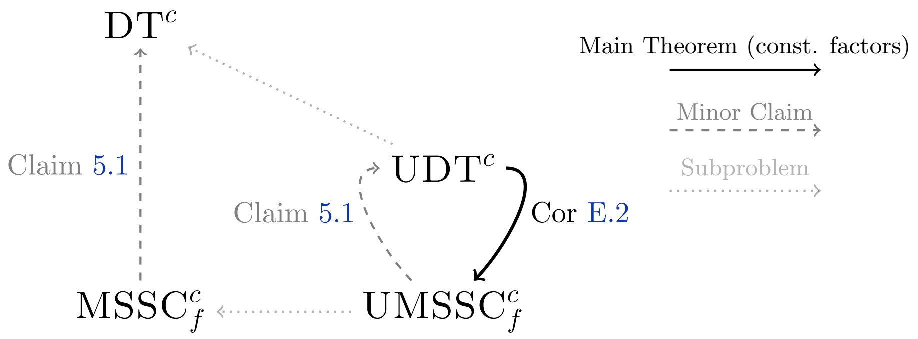

Figure 10: Summary of reductions for non unit cost boxes.

Corollary E.2 (Uniform Decision Tree with costs to UMSSC_f^c)**.** *Given an $\alpha(m, n)$-approximation algorithm for* UMSSC_f^c *then there exists an $O(\alpha(n + m, m))$-approximation algorithm for* UDT^c*.*

Proof. The proof follows exactly the same way as the proof of Theorem 5.2 with one change: the cost of an isolating element is the minimum cost test needed to isolate s_i from scenario s_k where

s_k is the scenario that maximizes this quantity. Formally, if $c(i,k) = \min\{c_j | T_j(i) \neq T_j(k)\}$, then $c(B^i) = \max_{k \in [m]} c(i,k)$. The reduction follows the exact steps as the one we described in Section C. $\square$