On the Complexity of Knapsack Under Explorable Uncertainty: Hardness and Algorithms

Jens Schlöter
Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands

Abstract

In the *knapsack problem under explorable uncertainty*, we are given a knapsack instance with uncertain item profits. Instead of having access to the precise profits, we are only given *uncertainty intervals* that are guaranteed to contain the corresponding profits. The actual item profit can be obtained via a *query*. The goal of the problem is to adaptively query item profits until the revealed information suffices to compute an optimal (or approximate) solution to the underlying knapsack instance. Since queries are costly, the objective is to minimize the number of queries.

In the offline variant of this problem, we assume knowledge of the precise profits and the task is to compute a query set of minimum cardinality that a third party without access to the profits could use to identify an optimal (or approximate) knapsack solution. We show that this offline variant is complete for the second-level of the polynomial hierarchy, i.e., Σ_2^p-complete, and cannot be approximated within a non-trivial factor unless $\Sigma_2^p = \Delta_2^p$. Motivated by these strong hardness results, we consider a "resource-augmented" variant of the problem where the requirements on the query set computed by an algorithm are less strict than the requirements on the optimal solution we compare against. More precisely, a query set computed by the algorithm must reveal sufficient information to identify an approximate knapsack solution, while the optimal query set we compare against has to reveal sufficient information to identify an optimal solution. We show that this resource-augmented setting allows interesting non-trivial algorithmic results.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Explorable uncertainty, knapsack, queries, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.6

Related Version *Full Version*: `https://arxiv.org/abs/2507.02657` [29]

Funding This work was partially supported by the research project *Optimization for and with Machine Learning (OPTIMAL)*, funded by the Dutch Research Council (NWO), grant number OCENW.GROOT.2019.015.

Acknowledgements Parts of the work on this project have been done while the author was affiliated with the University of Bremen. The author thanks Nicole Megow for initial discussions.

1 Introduction

The field of *explorable uncertainty* considers optimization problems with uncertainty in the numeric input parameters. Initially, the precise values of the uncertain parameters are unknown. Instead, for each uncertain parameter, we are given an *uncertainty interval* that contains the precise value of that parameter. Each uncertain parameter can be queried to reveal its precise value. The goal is to adaptively query uncertain parameters until we have sufficient information to solve the underlying optimization problem.

In this paper, we consider *knapsack under explorable uncertainty* (KnapExp) with uncertain item profits. That is, we are given a set of items $\mathcal{I}$ and a knapsack capacity $B \in \mathbb{N}$. Each item $i \in \mathcal{I}$ has a known weight $w_i \in \mathbb{N}$ and an uncertain profit $p_i \in \mathbb{R}$ that is initially hidden within the known uncertainty interval I_i, i.e., $p_i \in I_i$. A *query* of an item i reveals the profit p_i. Our goal is to compute a set $P \subseteq \mathcal{I}$ of items with $w(P) := \sum_{i \in P} w_i \leq B$ that maximizes the profit $p(P) := \sum_{i \in P} p_i$. We refer to this problem as the *underlying knapsack problem.* Since the profits are initially hidden within their uncertainty intervals, we do not always have sufficient information to compute an optimal or even approximate solution for the underlying knapsack problem. Instead, an algorithm for KnapExp can adaptively query items to reveal their profits until the revealed information suffices to compute an optimal solution for the underlying knapsack instance. As queries are costly, the goal is to minimize the number of queries.

The *offline version*, sometimes also called *verification problem*, of knapsack under explorable uncertainty (OfflineKnapExp) assumes full initial access to the profits p_i and asks for a *query set* $Q \subseteq \mathcal{I}$ of minimum cardinality such that access to the profits of the items in Q and access to the uncertainty intervals of the items in $\mathcal{I} \setminus Q$ suffices to compute an optimal solution to the underlying knapsack instance, independent of what the precise profits of the items in $\mathcal{I} \setminus Q$ are. In this work, we mainly focus on studying the offline version of knapsack under explorable uncertainty. Most commonly, problems under explorable uncertainy are studied in an *adversarial online setting*, where the uncertain values are unknown, query outcomes are returned in a worst-case manner, and algorithms are compared against the optimal solution for the corresponding offline version by using competitive analysis. The complexity of the offline version is a natural barrier for efficiently solving the online version.

So far, most problems that have been studied under explorable uncertainty have an underlying problem that belongs to the complexity class P, i.e., can be solved in polynomial time. The seminal work by Kahan [20] on computing the minimum in a set of uncertain values was followed by works on computing the k-th smallest uncertain value [16, 20], computing a minimum spanning tree with uncertain edge weights [10, 12, 14, 24, 25, 28], sorting [13, 19], shortest path [15], finding the cheapest set in a given family of sets [11, 26], simple geometric problems [7], stable matchings [2], and other selection problems [3, 13]. If we remove the explorable uncertainty aspect, then all of these problems can be solved in polynomial time.

Even tough these underlying problems are in P, the offline versions of the corresponding problems under explorable uncertainty are often NP-hard. For instance, the offline version of identifying the set of maximal points under explorable uncertainty is NP-hard [8], the offline version of the selection problem in [3, 13] is NP-hard, and the offline version of the minimum-spanning tree problem under vertex uncertainty is NP-hard [10]. The offline version of selecting the cheapest set is NP-hard [11] and even hard to approximate within a factor of $o(\log m)$, where m is the number of sets [26]. Similarly, the offline version of stable matching under uncertainty is NP-hard to approximate [2]. For all of these problems, adding the layer of explorable uncertainty increases the complexity from polynomial-time solvable to NP-hard and leads to interesting algorithmic challenges even tough the underlying problems are easy. However, this observation also raises the following question:

> If the underlying problem is already NP-hard, does adding the layer of explorable uncertainty still increase the complexity?

As a first main result, we answer this question in the affirmative for the offline version of knapsack under explorable uncertainty. More precisely, we show that OfflineKnapExp is complete for the second level of the polynomial hierarchy, i.e., Σ_2^p-complete. We even show that, under a certain conjecture ($\Sigma_2^p \neq \Delta_2^p$), no $n^{1-\epsilon}$-approximation is possible for any $\epsilon > 0$, where n is the number of items. The latter can be seen as a natural next step from the inapproximability result given in [26]: They show that approximating the offline version of the cheapest set problem is hard to approximate within a factor of $o(\log m)$ by exploiting that it is equivalent to solving a covering integer linear program (ILP) with m constraints, whereas we show our inapproximability result by exploiting that offline KnapExp can be represented as a covering ILP with an exponential number of constraints. Unfortunately, these extremely strong hardness results pose further challenges:

If the hardness of the offline version prevents any non-trivial approximation, is there any hope for interesting algorithmic results in the offline, online or stochastic version?

Our approach for answering this question is to consider a form of *resource augmentation*. More precisely, we relax the requirements on a solution Q for OfflineKnapExp: Instead of requiring that querying Q reveals sufficient information to identify an optimal solution for the underlying knapsack problem, we only require sufficient information to identify an α-approximate solution. Unfortunately, we can show that, unless P=NP, there is no non-trivial approximation for this relaxed problem variant if we compare against an optimal solution for the relaxed problem variant. However, as a second main result, we show that non-trivial algorithmic results are possible if the requirements on the algorithm's solution are less strict than the requirements on the optimal solution we compare against; we make this more precise in the next section.

1.1 Problem Definition

An *instance* $\mathcal{K}$ of KnapExp and OfflineKnapExp is a quintuple $\mathcal{K} = (\mathcal{I}, B, w, p, \mathcal{A})$, where $\mathcal{I} = \{1, \dots, n\}$ is a set of n items, $B \in \mathbb{N}$ is the knapsack capacity, w is the weight vector with $w_i \in \mathbb{N}_{\leq B}$ for all items $i \in \mathcal{I}$, p is the profit vector with $p_i \in \mathbb{R}_{\geq 0}$ for all $i \in \mathcal{I}$, and $\mathcal{A} = \{I_1, \dots, I_n\}$ is the set of uncertainty intervals such that $p_i \in I_i$ for all $i \in \mathcal{I}$. The quadruple $(\mathcal{I}, B, w, p)$ characterizes the underlying knapsack problem.

As is common in the area of explorable uncertainty, we assume that each uncertainty interval I_i is either open or *trivial*. That is, we either have $I_i = (L_i, U_i)$ for a *lower limit* L_i and an *upper limit* U_i, or $I_i = \{p_i\}$. In the latter case, we call both the item i and the uncertainty interval I_i *trivial* and define $U_i = L_i = p_i$. All items that are not trivial are called *non-trivial*. We use $\mathcal{I}_T$ to refer to the set of trivial items. A *query* of an item i reveals the profit p_i and can be seen as replacing the uncertainty interval $I_i = (L_i, U_i)$ with $I_i = \{p_i\}$.

In OfflineKnapExp, all input parameters are known to the algorithm, while in KnapExp the profits p_i are initially uncertain. Both problems ask for a *feasible query set* $Q \subseteq \mathcal{I}$ of minimum cardinality. Intuitively, a query set $Q \subseteq \mathcal{I}$ is feasible if the revealed information suffices to identify an optimal solution to the underlying knapsack problem *and* to determine the profit of such a solution. We proceed by making this definition more formal.

Packings and Feasible Query Sets. To formally define feasible query sets, we first define *packings*. A subset of items $P \subseteq \mathcal{I}$ is a *packing* if $\sum_{i \in P} w_i \leq B$. That is, packings are feasible solutions to the underlying knapsack problem. For $P \subseteq \mathcal{I}$ let $p(P) = \sum_{i \in P} p_i$ denote the profit of P. We call a packing *optimal* if it maximizes the profit over all packings. We usually use P^* to refer to an optimal packing and $p^* := p(P^*)$ to refer to the *optimal profit*.

For each packing $P \subseteq \mathcal{I}$ define $U_P := \sum_{i \in P} U_i$, i.e., the term U_P describes an upper limit on the maximum possible profit the packing could potentially have. Note that U_p can be computed even without access to the profits. By querying items in P, the upper limit U_P decreases as we gain more information and can replace the non-trivial uncertainty interval $I_i = (L_i, U_i)$ with $I_i = \{p_i\}$ after we query i and learn the profit p_i. For $Q \subseteq \mathcal{I}$, we use $U_i(Q)$ to denote the upper limit of i after querying Q, i.e., $U_i(Q) = p_i$ if $i \in Q$ and $U_i(Q) = U_i$ otherwise. The upper limit $U_P(Q)$ of packing P after querying a set $Q \subseteq \mathcal{I}$, is

$$U_P(Q) := \sum_{i \in P} U_i(Q) = \sum_{i \in P \setminus Q} U_i + \sum_{i \in P \cap Q} p_i = \sum_{i \in P} U_i - \sum_{i \in P \cap Q} (U_i - p_i) = U_P - \sum_{i \in P \cap Q} (U_i - p_i).$$

▶ **Definition 1.** *A query set $Q \subseteq \mathcal{I}$ is* feasible *if the following two conditions hold:*
1. *There is a packing $P \subseteq Q \cup \mathcal{I}_T$ with $p(P) = p^*$.*
2. *$U_P(Q) \leq p^*$ holds for every packing $P \subseteq \mathcal{I}$.*

The first condition of Definition 1 ensures that querying Q reveals sufficient information to verify that there exists a packing with the optimal profit p^*, while the second condition ensures that querying Q reveals sufficient information to verify that no packing can possibly have a larger profit than p^*, no matter what the profits of items $i \in \mathcal{I} \setminus Q$ actually are.

Since any packing P^* with $p(P^*) = p^*$ can only satisfy $U_{P^*}(Q) \leq p^*$ if $P^* \subseteq Q \cup \mathcal{I}_T$, the second condition of the definition actually implies the first one. In particular, this means that a query set is feasible if and only if it satisfies the constraints of the following ILP. Note that a similar covering point of view was first observed in [26] for the cheapest set problem.

$$\begin{array}{lll} \min & \sum_{i \in \mathcal{I}} x_i & \\ \text{s.t.} & \sum_{i \in P} x_i \cdot (U_i - p_i) \geq U_P - p^* & \forall P \subseteq \mathcal{I}\colon \sum_{i \in P} w_i \leq B \\ & x_i \in \{0,1\} & \forall i \in \mathcal{I} \end{array} \qquad \text{(K-ILP)}$$

The offline problem. In the offline problem OfflineKnapExp, we are given an instance $\mathcal{K} = (\mathcal{I}, B, w, p, \mathcal{A})$ with full knowledge of all parameters and our goal is to compute a feasible queryset of minimum cardinality, which is equivalent to solving (K-ILP) with full knowledge of all coefficients. We use Q^* to refer to an optimal solution of OfflineKnapExp.

The online problem. In the online problem KnapExp, we are also given an instance $\mathcal{K} = (\mathcal{I}, B, w, p, \mathcal{A})$ but the profits p_i are initially unknown. The goal is to iteratively and adaptively query items i to reveal their profit p_i until the set of queried items is a feasibile query set. This problem can be seen as solving (K-ILP) with uncertain coefficients $U_i - p_i$ and right-hand side values $U_P - p^*$: Querying an item i corresponds to irrevocably setting $x_i = 1$, incurs a cost that cannot be reverted, and reveals the coefficient $(U_i - p_i)$.

Relaxations. As OfflineKnapExp turns out to admit strong inapproximability results, we introduce the following relaxed notion of feasible query sets. We use (α, β)-OfflineKnapExp to refer to the offline problem of computing a (α, β)-feasible query set of minimum cardinality.

▶ **Definition 2** ((α, β)-feasibility)**.** *Let $\alpha, \beta \geq 1$. We say that a query set $Q \subseteq \mathcal{I}$ is (α, β)-feasible if the following two conditions hold:*
1. *There is a packing P such that $P \subseteq Q \cup \mathcal{I}_T$ and $p(P) \geq \frac{1}{\alpha} \cdot p^*$.*
2. *$U_P(Q) \leq \beta \cdot p^*$ for every packing P.*

The first condition of the definition ensures that we can find an α-approximation for the underlying knapsack instance by using only queried and trivial items. The second condition ensures that, after querying Q, no feasible packing can have profit greater than $\beta \cdot p^*$, no matter what the profits of the items in $\mathcal{I} \setminus Q$ are. Thus, querying Q reveals sufficient information to verify that the set P of the first condition is a $\frac{1}{\alpha\beta}$-approximation for the underlying knapsack instance. We use $Q^*_{\alpha,\beta}$ to refer to a minimum-cardinality (α, β)-feasible query set. Note that $Q^* = Q^*_{1,1}$, Q^* is (α, β)-feasible for every $\alpha, \beta \geq 1$ and $|Q^*| = \max_{\alpha \geq 1, \beta \geq 1} |Q^*_{\alpha,\beta}|$.

In contrast to Definition 1, the second condition of Definition 2 does not imply the first one. As a consequence, each (α, β)-feasible query set is a feasible solution for a variant of (K-ILP) in which we replace p^* with $\beta \cdot p^*$, but the inverse is not necessarily true.

1.2 Our Results and Outline

In Section 2, we give several hardness results for OfflineKnapExp. First, we show that deciding whether $Q = \emptyset$ is an (α, β)-feasible query set is weakly NP-hard for any $\alpha, \beta \geq 1$, which immediately implies that it is weakly NP-hard to approximate (α, β)-OfflineKnapExp within any bounded multiplicative factor. This hardness result mainly exploits that the optimal solution p^* to the weakly NP-hard [21] underlying knapsack problem appears on the right-hand sides of (K-ILP). Then, we move on to show that OfflineKnapExp is Σ_2^p-complete, which intuitively means that the problem remains hard even if we are given an oracle for deciding problems in NP. Since such an oracle allows us to compute p^*, the reason for the Σ_2^p-hardness is not the appearance of p^* in the right-hand sides but the exponential number of constraints in (K-ILP). In fact, we prove this hardness result via reduction from *succinct set cover* [31,32], which is a set cover variant where the elements and sets are only given implicitly, i.e., the number of set cover constraints is exponential in the encoding size of the problem. Exploiting a result by [30], our reduction also shows that there is no $n_0^{1-\epsilon}$-approximation for any $\epsilon > 0$ unless $\Sigma_2^p = \Delta_2^p$, where $n_0 := |\mathcal{I} \setminus \mathcal{I}_T|$.

In Section 3, we design algorithms for (α, β)-OfflineKnapExp for different values of α and β. Since the hardness results prevent any non-trivial results when comparing against $|Q^*_{\alpha,\beta}|$, we analyze our algorithms by comparing their solutions to $|Q^*|$ instead. This can be seen as a form of resource augmentation as the requirements on the algorithms's solution are less strict than the requirements on the optimal solution we compare against. To achieve our algorithmic results, we treat the two conditions for (α, β)-feasible query sets (cf. Definition 2) as separate subproblems: (i) Compute a query set Q_1 such that there exists a packing $P \subseteq Q_1 \cup \mathcal{I}_T$ with $p(P) \geq \frac{1}{\alpha} p^*$, and (ii) Compute a query set Q_2 such that $U_P(Q_2) \leq \beta p^*$ holds for all packings P. First, we show how to solve subproblem (i) in polynomial-time for $\alpha = \frac{1}{1-\epsilon}$ with a set Q_1 such that $|Q_1| \leq |Q^*|$. Our algorithm for this subproblem exploits existing results for the two-dimensional knapsack problem. For (ii), we first show how to solve the problem in pseudopolynomial time for $\beta = 2 + \epsilon$ with the guarantee $|Q_2| \leq |Q^*|$. The algorithm is based on solving an OfflineKnapExp variant that only considers packings that correspond to certain greedy solutions. We justify the pseudopolynomial running-time by showing weak NP-hardness for this OfflineKnapExp variant. By considering a relaxed version of that problem, we manage to solve problem (ii) for $\beta = 4 + \epsilon$ with $|Q_2| \leq |Q^*|$ in polynomial time. Combining the results for both subproblems yields a pseudopolynomial algorithm that computes a $(\frac{1}{1-\epsilon}, 2+2\epsilon)$-feasible query set Q and a polynomial time algorithm that computes a $(\frac{1}{1-\epsilon}, 4+4\epsilon)$-feasible query set Q. In both cases, $|Q| \leq 2 \cdot |Q^*|$.

1.3 Further Related Work

Meißner [27] gives an adversarial lower bound of n for KnapExp that holds even if the instance only has two different weights, preventing any non-trivial adversarial results. However, Megow and Schlöter [26] show that this lower bound does not hold in a stochastic setting where the profits p_i are drawn from their intervals I_i according to an unknown distribution that satisfies $\Pr[p_i \leq \frac{U_i + L_i}{2}] \leq \tau$ for a threshold parameter τ. However, their result only breaks that particular lower bound instance and does not imply general stochastic results for KnapExp.

Goerigk et al. [18] consider a knapsack problem under uncertainty in a different query setting. In their problem, the profits are known and the weights are uncertain. Furthermore, there is a budget on the queries that an algorithm is allowed to make. These differences lead to a problem fundamentally different from KnapExp.

Maehara and Yamaguchi [23] consider packing ILPs with uncertainty in the cost coefficients. The cost coefficients can be queried. The key difference to the setting of explorable uncertainty is that they are interested in bounding the absolute number of queries instead of comparing against the optimal feasible query set. We remark that this is an important distinction between explorable uncertainty and many other query models. For example, the same distinction applies to a line of research that studies queries that reveal the *existence* of entities instead of numeric values, e.g., the existence of edges in a graph, c.f. [4–6, 9, 17, 33]. For instance, Behnezhad et al. [4] considered vertex cover in a stochastic setting and showed that it can be approximated within a factor of $(2+\epsilon)$ with only a constant number of queried edges per vertex.

2 Hardness of Approximation

We start by showing our hardness results for (α, β)-OfflineKnapExp. Not surprisingly, the appearance of p^* in the right-hand sides of (K-ILP) suffices to render (α, β)-OfflineKnapExp weakly NP-hard. The following proposition shows that even deciding whether a given set Q is (α, β)-feasible is already weakly NP-hard (cf. the full version [29] for a formal proof).

▶ **Proposition 3.** *Deciding if $Q = \emptyset$ is (α, β)-feasible is weakly NP-hard for any $\alpha, \beta \geq 1$.*

This means that distinguishing between instances that can be solved without any query and instances that need at least one query is weakly NP-hard, which implies the following:

▶ **Corollary 4.** *It is weakly NP-hard to approximate (α, β)-*OfflineKnapExp *within any bounded multiplicative factor.*

Proposition 3 is also an indicator that (α, β)-OfflineKnapExp might not be in NP, as verifying whether a query set is (α, β)-feasible is already NP-hard. For $\alpha, \beta = 1$, we make this observation more formal by proving that OfflineKnapExp is complete for the second level of the polynomial hierarchy, i.e., Σ_2^p-complete. Intuitively, the class Σ_2^p contains problems that, given an oracle for deciding problems in NP, can be solved in non-deterministic polynomial time. Similarly, the class Δ_2^p contains problems that, given the same type of oracle, can be solved in deterministic polynomial time. Hardness and completeness for the class Σ_2^p are defined in the same way as for the class NP. For a more formal introduction, we refer to [1]. Under the conjecture that $\sum_2^p \neq$ NP, the $\sum_2^p$-completeness implies that OfflineKnapExp is not in NP, and under the conjecture $\sum_2^p \neq \Delta_2^p$ it cannot be solved optimally in polynomial time even when given an oracle for deciding problems in NP.

▶ **Theorem 5.** OfflineKnapExp *is Σ_2^p-complete.*

Proof. We show Σ_2^p-membership in the full version [29] and focus here on proving Σ_2^p-hardness. Our proof is via reduction from *succinct set cover*, which is known to be $\sum_2^p$-complete [31, 32]. In the same way as for NP-hardness proofs, we need to give a polynomial time reduction to the decision problem variant of OfflineKnapExp such that the constructed instance is a YES-instance if and only if the given succinct set cover instance is a YES-instance.

Succinct set cover. We are given n decision variables $x_1, \dots, x_n$, m propositional logic formulas $\phi_1, \dots, \phi_m$ over these variables and an integer parameter k. Each formula ϕ_j is in 3-DNF form[1] and we use S_j to denote the set of 0-1-vectors (variable assignments) that

[1] *Disjunctive normal form (DNF)* refers to a disjunction of conjunctions, i.e., $\phi_j = C_{j,1} \vee \ldots \vee C_{j,k_j}$, where each clause $C_{j,k}$ is a conjunction of literals. In 3-DNF, each $C_{j,k}$ contains exactly three literals. A formula in DNF is satisfied by a variable assignment if at least one clause is satisfied by the assignment.

satisfy ϕ_j. The formulas ϕ_j have to satisfy $\bigcup_{j\in\{1,\dots,m\}} S_j = \{0,1\}^n$, i.e., each variable assignment satisfies at least one formula ϕ_j. The goal is to find a subset $S \subseteq \{1,\dots,m\}$ such that $\bigcup_{j\in S} S_j = \{0,1\}^n$ and $|S| \leq k$. We assume that each variable occurs as a literal in at least one formula. If not, we can just remove the variable and obtain a smaller instance. Succinct set cover can be interpreted as a set cover variant where the elements and sets are only given *implicitly* and not as an explicit part of the input.

Main reduction idea. Before we give the technical details of the reduction, we first sketch the basic idea. In particular, we describe the properties that we want the constructed instance to have. In the technical part, we then describe how to actually achieve these properties. At its core, the reduction will use the knapsack weights to encode structural information of the input instance into the constructed instance. The idea to use numerical weights to encode constraints is quite natural in hardness proofs for weakly NP-hard problems, see e.g. the NP-hardness proof for the subset sum problem given in [22]. The usage of the knapsack weights in our reduction is on some level inspired by such classical reductions, but requires several new ideas to handle the implicit representation of the input problem and the covering nature of OfflineKnapExp.

First, we introduce a single trivial item i^* with $w_{i^*} = p_{i^*} = B$. This item alone fills up the complete knapsack capacity B, which we define later, and the instance will be constructed in such a way that $p^* = p_{i^*} = B$ is the maximum profit of any packing. Thus, only packings P with $U_P > p^*$ induce non-trivial constraints in (K-ILP). We design the instance such that $U_P \geq p^*$ only holds for packings that use the full capacity.

▶ **Property 1.** A packing P of the constructed instance satisfies $U_P \geq p^*$ only if $w(P) = B$.

Next, we want each packing P with $w(P) = B$ and $U_P > p^*$ to represent a distinct variable assignment in $\{0,1\}^n$. To this end, we introduce a set X of $2n$ items, two items v_i and $\bar{v}_i$ for each variable x_i with $i \in \{1,\dots,n\}$. Intuitively, v_i represents the positive literal x_i and $\bar{v}_i$ represents the negative literal $\neg x_i$. We say that a subset $X' \subseteq X$ *represents* a variable assignment if $|X' \cap \{v_i, \bar{v}_i\}| = 1$ for all $i \in \{1,\dots,n\}$. We design our instance such that the packings P with $w(P) = B$ and $U_P > p^*$ exactly correspond to the variable assignments in $\{0,1\}^n$. Note that this excludes the packing $P = \{i^*\}$ as this packing has $w(P) = B$ and $p(P) = p^*$.

▶ **Property 2.** If $w(P) = B$ and $U_P > p^*$, then $P \cap X$ represents a variable assignment. Each variable assignment is represented by at least one P with $w(P) = B$ and $U_P > p^*$.

If the first two properties hold, then all non-trivial constraints in the ILP (K-ILP) for the constructed instance correspond to a packing P with $w(P) = B$ and $U_P > p^*$ and, thus, to a variable assignment of the given succinct set cover instance. Furthermore, each variable assignment is represented by at least one active constraint. With the next property, we want to ensure that each possible query, i.e., each non-trivial item, corresponds to a succinct set cover formula ϕ_j. To this end, we introduce the set of items $Y = \{y_1,\dots,y_m\}$. These items will be the only non-trivial items in the constructed instance, so each possible query is to an element of Y. Next, we want to achieve that querying an item y_j suffices to satisfy all constraints of (K-ILP) for packings P with $w(P) = B$ and $U_P > p^*$ that represent a variable assignment which satisfies formula ϕ_j, and does not impact any other constraints. Formally, we would like to design our instance such that the following property holds.

▶ **Property 3.** For each packing P with $U_P > p^*$ and each $y_j \in Y$: $y_j \in P$ if and only if $X \cap P$ represents a variable assignment that satisfies ϕ_j. If $y_j \in P$, then $U_P - p^* \leq U_{y_j} - p_{y_j}$.

If we manage to define our reduction in such a way that the three properties are satisfied, it is not hard to show correctness (see the full version [29] for the second direction):

First Direction. If there is an index set S with $|S| \leq k$ that is a feasible solution to the succinct set cover instance, then each possible variable assignment must satisfy at least one formula ϕ_j with $j \in S$. We claim that $Q = \{y_j \mid j \in S\}$ is a feasible query set for the constructed OfflineKnapExp instance. To this end, consider an arbitrary packing P with $U_P > p^*$, which are the only packings that induce non-trivial constraints in the corresponding (K-ILP). By Property 1, we have $w(P) = B$. Property 2 implies that $X \cap P$ represents some variable assignment φ and Property 3 implies that $y_j \in P$ for all ϕ_j that are satisfied by φ. By assumption that S is a feasible succinct set cover solution, we get $Q \cap P \neq \emptyset$. Property 3 implies $U_P(Q) \leq U_P(\{y_j\}) \leq p^*$ for each $y_j \in Q \cap P$. Thus, Q satisfies the constraint of P in the (K-ILP) for the constructed instance. As this holds for all P with $U_P > p^*$, the set Q is a feasible solution for the constructed OfflineKnapExp instance.

Technical reduction. It remains to show how to actually construct an OfflineKnapExp instance that satisfies the three properties. Given an instance of succinct set cover, we construct an instance of OfflineKnapExp consisting of four sets X, Φ, A and L of items such that $\mathcal{I} = X \cup \Phi \cup A \cup L$. The set X is defined exactly as sketched above, the set Φ contains the set $Y = \{y_j, \ldots, y_m\}$ as introduced above, and $L := \{i^*\}$ for the item i^* with $w_{i^*} = p_{i^*} = B$. All further items will be used to ensure the three properties.

Conceptionally, we construct *several* partial weights for each item i that will later be combined into a single weight. For each item i, we construct two weights w_{i,ϕ_j} and w_{i,ρ_j} for each formula ϕ_j, and a single weight $w_{i,x}$. Similarly, we break down the knapsack capacity into multiple partial knapsack capacities B_x, and B_{ϕ_j}, B_{ρ_j} for each ϕ_j. Intuitively, the full weight w_i of an item i will be the concatenation of the decimal representations of the partial weights, i.e., $w_i = w_{i,x} w_{i,\rho_m} \cdots w_{i,\rho_1} w_{i,\phi_m} \cdots w_{i,\phi_1}$, and B will be the concatenation of the partial capacities. We make this more precise in the full version [29] after defining all partial weights in such a way that the following property holds.

▶ **Property 4.** For each packing P with $P \neq \{i^*\}$, it holds $\sum_{i \in P} w_i = B$ if and only if $\sum_{i \in P} w_{i,x} = B_x$, and $\sum_{i \in P} w_{i,\phi_j} = B_{\phi_j}$ and $\sum_{i \in P} w_{i,\rho_j} = B_{\rho_j}$ for all $j \in \{1, \ldots, m\}$.

For now, we operate under the assumption that Property 4 holds and focus on the partial weights and capacities, and proceed by defining the remaining parts of the construction.

Definition of the w_x-weights. As formulated in Property 2, we would like each packing P with $w(P) = B$ and $U_P > p^*$ to represent a variable assignment. To this end, we need such a packing to contain exactly one item of $\{v_i, \bar{v}_i\}$ for each $i \in \{1, \ldots, n\}$. To enforce this, we use the partial w_x-weights and the partial capacity B_x. In particular, we define $w_{v_i,x} = w_{\bar{v}_i,x} = 10^i$ for each $i \in \{1, \ldots, n\}$, and $B_x = \sum_{i=1}^{n} 10^i$. For all items $j \in \mathcal{I} \setminus X$, we define $w_{j,x} = 0$, which immediately implies the following property:

▶ **Property 5.** *P satisfies $\sum_{i \in P} w_{i,x} = B_x$ iff $P \cap X$ represents a variable assignment.*

Definition of the set A and the w_{ϕ_j}-weights. Define $A = \bigcup_{j \in \{1,\ldots,m\}} A_{\phi_j}$ for sets A_{ϕ_j} to be defined below. For formula ϕ_j, let k_j denote the number of clauses in ϕ_j and let $C_{j,1}, \ldots, C_{j,k_j}$ denote these clauses. For each $C_{j,k}$, we add four items $a_{j,k,0}, a_{j,k,1}, a_{j,k,2}, a_{j,k,3}$ to set A_{ϕ_j}. The idea is to define the partial w_{ϕ_j}-weights and the partial B_{ϕ_j} capacity in such a way that the following property holds.

▶ **Property 6.** For each ϕ_j, a packing P satisfies $\sum_{i\in P} w_{i,\phi_j} = B_{\phi_j}$ and $\sum_{i\in P} w_{i,x} = B_x$ iff $a_{j,k,0} \in P$ for each clause $C_{j,k}$ that is satisfied by the assignment represented by $X \cap P$.

To achieve the property, we first define the partial w_{ϕ_j}-weights for the items X. For a literal x_i, let $\mathcal{C}_{x_i,j} := \{k \mid x_i \text{ occurs in } C_{j,k}\}$ and define $\mathcal{C}_{\neg x_i,j}$ in the same way. We define the weights w_{v_i,ϕ_j} and $w_{\bar{v}_i,\phi_j}$ as $\sum_{k\in\mathcal{C}_{x_i,j}} 10^{k_j+k-1}$ and $\sum_{k\in\mathcal{C}_{\neg x_i,j}} 10^{k_j+k-1}$, respectively. Intuitively, digit $k_j + k$ of the sum $\sum_{h\in X\cap P} w_{h,\phi_j}$ of a packing P with $\sum_{i\in P} w_{i,x} = B_x$ indicates whether the assignment represented by $X \cap P$ satisfies clause $C_{j,k}$ or not: If the clause is satisfied, then $X \cap P$ contains the items that represent the three literals of $C_{j,k}$ and the digit has value 3. Otherwise, $X \cap P$ contains at most two of the items that represent the literals of $C_{j,k}$ and the digit has value at most 2.

Finally, for each for $i \in \{0, 1, 2, 3\}$, we define the w_{ϕ_j}-weight of item $a_{j,k,i}$ as $w_{\phi_j,a_{j,k,i}} = i\cdot 10^{k_j+k-1} + 10^{k-1}$. For all remaining items $i \in \mathcal{I}\setminus(X \cup A_{\phi_j})$, we define the partial w_{ϕ_j}-weight to be zero. Furthermore, we define the partial capacity $B_{\phi_j} = \sum_{k=0}^{k_j-1} 10^k + \sum_{k=k_j}^{2k_j-1} 3\cdot 10^k$. We claim that these definitions enforce Property 6.

Intuitively, the fact that the k_j decimal digits of lowest magnitude in B_{ϕ_j} have value 1 forces a packing P with $\sum_{i\in P} w_{i,\phi_j} = B_{\phi_j}$ to contain exactly one item of $\{a_{j,k,0}, \ldots, a_{j,k,3}\}$ for each $k \in \{1, \ldots, k_j\}$ as each such item increases the corresponding digit $k - 1$ by one. Similarly, the value of each digit $k_j + k - 1$ in B_{ϕ_j} is three. Since the elements of X that occur in clause $C_{j,k}$ increase the value of digit $k_j + k - 1$ in $w_{\phi_j}(P)$ by one and item $a_{j,k,i}$ increases the digit by i, a packing P with value three in digit $k_j + k - 1$ of $w_{\phi_j}(P)$ can contain item $a_{j,k,0}$ if and only if $X \cap P$ contains the three items that correspond to the literals in $C_{j,k}$. This implies Property 6. We give a more formal argumentation in the full version [29].

Definition of set Φ and the w_{ρ_j}-weight. Define $\Phi = \bigcup_{j\in\{1,\ldots,m\}} \Phi_j$ such that $\Phi_j = \{y_j, u_j, f_{j,0}, \ldots, f_{j,k_j-1}\}$. Note that y_j is the item that has already been introduced for Property 3. As a step toward enforcing this property, we define the w_{ρ_j}-weights such that:

▶ **Property 7.** For each ϕ_j, a packing P with $\sum_{i\in P} w_{i,\rho_j} = B_{\rho_j}$ has $y_j \in P$ if and only if $a_{j,k,0} \in P$ for some clause $C_{j,k}$ in ϕ_j.

To enforce this property, we define the following partial capacity $B_{\rho_j} = k_j + 10^{k_j^2} + 10^{k_j^2+1}$. Next, we define the w_{ρ_j}-weight of the elements $a_{j,k,0}$, $k \in \{1, \ldots, k_j\}$, as $w_{a_{j,k,0},\rho_j} = 1$. Furthermore, we define $w_{u_j,\rho_j} = 10^{k_j^2+1} + 10^{k_j^2} + k_j$, $w_{y_j,\rho_j} = 10^{k_j^2+1}$ and $w_{f_{j,k},\rho_j} = 10^{k_j^2} + k$, for all $k \in \{0, \ldots, k_j - 1\}$. For all other items, define the w_{ρ_j}-weight to be zero. We show in the full version [29] that these definitions imply Property 7.

Definition of the uncertainty intervals and precise profits. To finish the reduction, we define the profits and uncertainty intervals of all introduced items:

- For the items y_j, $j \in \{1, \ldots, m\}$, we define the uncertainty interval $I_{y_j} = (w_{y_j} - 2, w_{y_j} + \epsilon)$ for a fixed $0 < \epsilon < \frac{1}{m}$. We define the profits as $p_{y_j} = w_{y_j} - 1$.
- For all items $i \in \mathcal{I} \setminus \{y_j \mid j \in \{1, \ldots, m\}\}$, we use trivial uncertainty intervals $I_i = \{w_i\}$.

Proof of the three main properties. With the full construction in place, we are ready to prove the three main properties from the beginning of the proof:

1. **Property 1:** By definition of the profits, we have $p(P) \le w(P)$ for each packing P, which implies that the maximum profit is $p^* \le B$. Since the packing $P = L = \{i^*\}$ has a profit of exactly B, we get $p^* = B$. On the other hand, the upper limit U_P of a packing P is $w(P) + \epsilon|P \cap \{y_j \mid j \in \{1, \ldots, m\}\}|$ as only the items in $\{y_j \mid j \in \{1, \ldots, m\}\}$ have

a non-trivial uncertainty interval with upper limits of $w_{y_j} + \epsilon$. By choice of ϵ, this gives $U_P = w(P) + \epsilon|P \cap \{y_j \mid j \in \{1, \dots, m\}\}| < w(P) + 1$. Since all weights are integer, this implies that $U_P \geq p^* = B$ only holds if $w(P) = B$.

2. **Property 2:** The property, in particular the first part, is essentially implied by Property 4 and Property 5. We give the formal argumentation in the full version [29].
3. **Property 3:** By Property 1 and definition of the uncertainty intervals, a packing P has $U_P > p^*$ if and only if $w(P) = B$ and $P \neq \{i^*\}$. By Property 4, the latter holds if and only if $w_x(P) = B_x$, $w_{\phi_j}(P) = B_{\phi_j}$ and $w_{\rho_j}(P) = B_{\rho_j}$ for all $j \in \{1, \dots, m\}$. Fix a packing P with $y_j \in P$ for some $j \in \{1, \dots, m\}$. By Property 7, $y_j \in P$ holds if and only if $a_{j,k,0} \in P$ for some clause $C_{j,k}$ in ϕ_j. By Property 6, $a_{j,k,0} \in P$ if and only if $C_{j,k}$ is satisfied by the assignment represented by $X \cap P$. This gives the first part of Property 3. For the final part, observe that $U_{y_j} - p_{y_j} > 1$. On the other hand, $U_p - p^* < 1$. Hence, the second part of Property 3 holds.

To finish the proof of the reduction, it remains to argue about the running time and space complexity of the reduction. We do so in the full version [29]. The main argument is that, while the numerical values of the constructed weights are exponential, the number of digits in their decimal representations (and, thus, their encoding size) is polynomial. ◀

The previous theorem proves $\sum_2^p$-hardness for OFFLINEKNAPEXP. Exploiting the inapproximability result on the succinct set cover problem given in [30, Theorem 7.2], we can show the following stronger statement by using the same reduction (cf. the full version [29] for a formal proof).

▶ **Theorem 6.** *Unless $\Sigma_2^p = \Delta_2^p$, there exists no $n_0^{1-\epsilon}$-approximation (given access to an oracle for problems in NP) for* OFFLINEKNAPEXP *for any $\epsilon > 0$, where $n_0 := |\mathcal{I} \setminus \mathcal{I}_T|$.*

▶ Remark 7. We remark that all results given in this section require large numerical input parameters. Hence, they do not prohibit the existence of pseudopolynomial algorithms.

3 Algorithmic Results

In this section, we give algorithms for (α, β)-OFFLINEKNAPEXP for different values of α and β. Motivated by the hardness results of the previous section, we show bounds on the size of the computed query sets in comparison to $|Q^*|$ instead of $|Q^*_{\alpha,\beta}|$. All our algorithms treat the two conditions on (α, β)-feasible query sets (cf. Definition 2) as two separate subproblems:

1. Compute a query set Q_1 such that there exists a packing $P \subseteq Q_1 \cup \mathcal{I}_T$ with $p(P) \geq \frac{1}{\alpha} p^*$.
2. Compute a query set Q_2 such that $U_P(Q_2) \leq \beta \cdot p^*$ for all packings P.

In the following, we show how to solve these two problems and give bounds on $|Q_1 \cup Q_2|$ in terms of $|Q^*|$.

3.1 The First Subproblem

The following lemma solves the first subproblem for $\alpha = \frac{1}{1-\epsilon}$ by computing a packing P with $p(P) \geq (1 - \epsilon) \cdot p^*$ and $|P \setminus \mathcal{I}_T| \leq |Q^*|$. The set $Q_1 = P \setminus \mathcal{I}_T$ satisfies the requirement of the subproblem and has $|Q_1| \leq |Q^*|$. In the full version [29], we prove the lemma by exploiting existing algorithms for two-dimensional knapsack.

▶ **Lemma 8.** *Fix an $\epsilon > 0$. Given an instance of* OFFLINEKNAPEXP, *there exists a polynomial time algorithm that computes a packing P with $p(P) \geq (1 - \epsilon) \cdot p^*$ and $|P \setminus \mathcal{I}_T| \leq |Q^*|$.*

3.2 The Second Subprobem: Prefix Problems

For the second subproblem, we consider special packings that correspond to greedy solutions for the underlying knapsack problem. For an item $i \in \mathcal{I}$, define the *density* $d_i = \frac{p_i}{w_i}$ and the *optimistic density* $\bar{d}_i = \frac{U_i}{w_i}$. For a query set $Q \subseteq \mathcal{I}$, define the optimistic density of an item i *after* querying Q as $\bar{d}_i(Q) = \bar{d}_i$ if $i \notin Q$ and $\bar{d}_i(Q) = d_i$ if $i \in Q$. We use $\bar{\prec}_Q$ to denote the *optimistic density order* after querying Q, that is, for $i, j \in \mathcal{I}$ we use $i \bar{\prec}_Q j$ to denote that $\bar{d}_i(Q) \geq \bar{d}_j(Q)$. We assume an arbitrary but fixed tiebreaking rule between the items to treat $\bar{\prec}_Q$ as a total order. This allows us to define *optimistic prefixes* and the *prefix problem.*

▶ **Definition 9** (Optimistic prefixes). *For a set $Q \subseteq \mathcal{I}$ and a parameter $0 \leq C \leq B$, define the* optimistic prefix *$\bar{F}_C(Q)$ to be the maximal prefix S of order $\bar{\prec}_Q$ such that $w(S) \leq C$. To shorten the notation, we define $\bar{F}(Q) := \bar{F}_B(Q)$*

From the analysis of the knapsack greedy algorithm, it is well-known that the following holds for all packings P:

$$U_P(Q) \leq U_{\bar{F}(Q)}(Q) + \max_{i \in \mathcal{I}} U_i(Q). \tag{1}$$

If we compute a set Q, such that $U_{\bar{F}(Q)} \leq \beta' p^*$ and $\max_{i \in \mathcal{I}} U_i(Q) \leq \beta' \cdot p^*$, then Q solves the second subproblem for $\beta = 2 \cdot \beta'$, which motivates the following problem.

▶ **Definition 10** (Prefix problem). *Given an instance of* OfflineKnapExp *and a threshold parameter $D \geq p^*$, where p^* is the optimal profit of the underlying knapsack instance, the* prefix problem *asks for the set $Q \subseteq \mathcal{I}$ of minimum cardinality such that $U_{\bar{F}(Q)}(Q) \leq D$.*

Unfortunately, the prefix problem preserves the hardness of knapsack, even if the given threshold is larger than p^* by a constant factor. We show this in the full version [29].

▶ **Theorem 11.** *The prefix problem is weakly NP-hard for every $D = c \cdot p^*$ with $c \geq 1$.*

On the positive side, the problem can be solved to optimality in pseudopolynomial time.

▶ **Theorem 12.** *The prefix problem can be solved in pseudopolynomial time.*

We give the full proof of Theorem 12 in the full version [29], but highlight the main ideas here. In the following, we use Q_F^* to refer to an optimal solution for the prefix problem.

Assume for now, that the algorithm knows the last item i_1 in the prefix $\bar{F}(Q_F^*)$ and the first item i_2 outside of $\bar{F}(Q_F^*)$ in the order $\bar{\prec}_{Q_F^*}$ (Figure 1 (a)) and, for the sake of simplicity, assume that $i_1, i_2 \notin Q_F^*$. We design our algorithm to reconstruct the optimal solution $\bar{F}(Q_F^*)$ (or a similar solution) using just the knowledge of i_1 and i_2.

If we look at the same two items i_1, i_2 in the initial optimistic density order $\bar{\prec}_\emptyset$, then there can be a subset of items S before i_1, a subset of items A between i_1 and i_2 and a subset of items R after i_2 (Figure 1 (b)). Based on i_1 and i_2 being next to each other in order $\bar{\prec}_{Q_F^*}$, we can immediately deduce that $A \subseteq Q_F^*$ as items in $A \setminus Q_F^*$ would still be between i_1 and i_2 in $\bar{\prec}_{Q_F^*}$. Thus, the algorithm can safely add A to its solution as the optimal solution does the same. Similarly, from i_2 being the first item outside of $\bar{F}(Q_F^*)$, it is clear that $R \cap Q_F^* = \emptyset$ as the items in R stay outside the prefix $\bar{F}(Q_F^*)$ whether they are queried or not. Hence, the algorithm can safely ignore such items.

In the order $\bar{\prec}_A$, i.e., in the order after adding A to the solution, the items i_1 and i_2 must already be next to each other. However, we still can have $i_1 \notin \bar{F}(A)$, that is, i_1 might not yet be part of the prefix (Figure 1 (c)). To fix this, the algorithms needs to query items from the

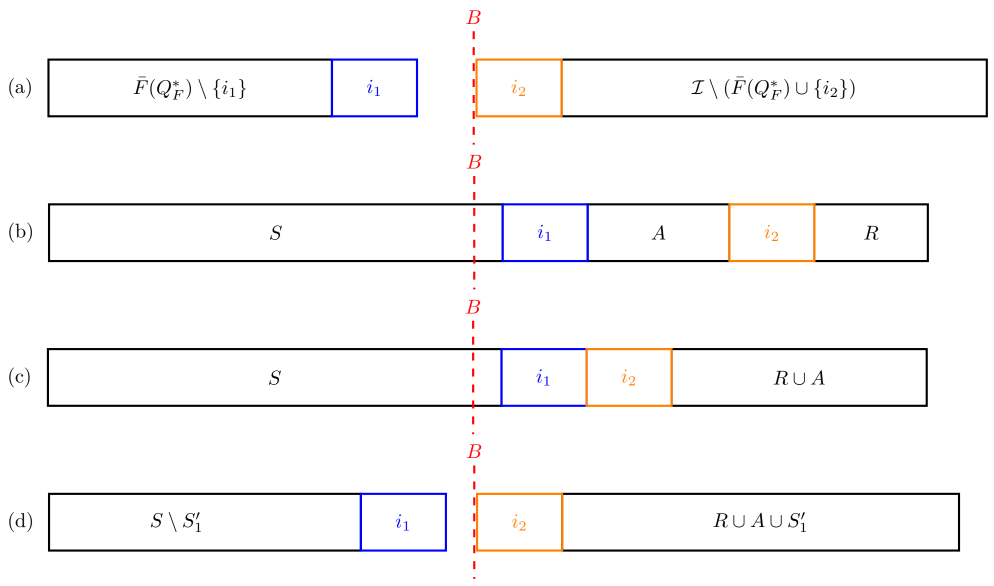

Figure 1 Illustration of the algorithmic ideas used to prove Theorem 12.

set $S_1 \subseteq S$, which contains items that are before i_1 in the order $\bar{\preceq}_A$ but would move behind i_2 if they are added to the solution. To reconstruct the optimal solution, the algorithm has to query these items in such a way that i_1 enters the prefix but i_2 does not. On the one hand, the algorithm should select items $i \in S_1$ with high U_i as the items will leave the prefix and the goal of the prefix problem is to decrease the upper limit of the prefix below the threshold D. On the other hand, the algorithm needs to make sure not to query too many items in S_1, so the cardinality of the solution does not grow too large. If K is the minimum amount of weight that needs to be queried for i_1 to enter the prefix, D is the maximum amount of weight that can be queried before i_2 enters the prefix, and n_1 is the number of queries that the algorithm can afford, then the algorithm should select its queries from S_1 according to the following ILP, which we show to be solvable in pseudopolynomial time:

$$\begin{array}{lll} \max & \sum_{i \in S_1} x_i \cdot U_i & \\ \text{s.t.} & \sum_{i \in S_1} x_i \cdot w_i \geq K & \\ & \sum_{i \in S_1} x_i \cdot w_i \leq H & \\ & \sum_{i \in S_1} x_i = n_1 & \\ & x_i \in \{0,1\} & \forall i \in S_1 \end{array} \tag{$\mathcal{P}_{S_1}$}$$

After adding the solution S_1' of $(\mathcal{P}_{S_1})$ to the solution, the prefix $\bar{F}(A \cup S_1')$ of the algorithm has reached roughly the same configuration as $\bar{F}(Q_F^*)$: i_1 is last in the prefix and i_2 is first outside the prefix (Figure 1 (d)). However, we still might have $U_{\bar{F}(A \cup S_1')}(A \cup S_1') > D$. To fix this, the algorithm has to query items of $S \setminus S_1'$ that stay in front of i_1 in the optimistic density order even after being queried. Since these items are part of the prefix $\bar{F}(A \cup S_1')$ *and* will stay part of the prefix even after being queried, the algorithm should greedily query such items i with large $U_i - p_i = U_{\bar{F}(A \cup S_1')}(A \cup S_1') - U_{\bar{F}(A \cup S_1' \cup \{i\})}(A \cup S_1' \cup \{i\})$ until the solution becomes feasible for the prefix problem instance. Our algorithm for Theorem 12 is based on exactly this approach. We show in the full version [29] how to formalize this and get rid of all assumptions that were used in the description above.

To achieve polynomial running time, we use the same approach but instead of $(\mathcal{P}_{S_1})$ we solve a certain LP-relaxation with the property that an optimal basic feasible solution has at most two fractional variables. Omitting the fractional items leads to the following corollary.

▶ **Corollary 13.** *Given an instance of the prefix problem with threshold parameter D, let Q^*_F denote an optimal solution to the instance. There exists a polynomial time algorithm that computes a set Q with $|Q| \leq |Q^*_F|$ such that $U_{\bar{F}(Q)}(Q) \leq D + 2 \cdot \max_{i \in \mathcal{I}} U_i$.*

3.3 Combining the Subproblems

By combining Lemma 8 and Theorem 12, we can show the following theorem. The idea is to use Lemma 8 to compute a packing P with $p(P) \geq (1 - \epsilon')p^*$ for a carefully chosen $\epsilon' > 0$ and then use Theorem 12 with threshold $D = \frac{p(P)}{(1-\epsilon')}$ to compute a solution Q' to the prefix problem. Exploiting (1), we return the solution $Q = (P \setminus \mathcal{I}_T) \cup Q' \cup \{i \in \mathcal{I} \mid U_i > D\}$ and observe that Q satisfies the theorem. A formal proof is given in the full version [29].

▶ **Theorem 14.** *Fix $\epsilon > 0$. There exists a pseudopolynomial algorithm that given an instance of* OfflineKnapExp *computes a $(\frac{1}{1-\epsilon}, (1+\epsilon) \cdot 2)$-feasible query set Q with $|Q| \leq 2|Q^*|$.*

Replacing the usage of Theorem 12 with Corollary 13 in the approach above yields:

▶ **Theorem 15.** *Fix $\epsilon > 0$. There exists a polynomial time algorithm that given an instance of* OfflineKnapExp *computes a $(\frac{1}{1-\epsilon}, (1+\epsilon) \cdot 4)$-feasible query set Q with $|Q| \leq 2|Q^*|$.*

4 Conclusion

We hope that our results on OfflineKnapExp improve the understanding of NP-hard problems under explorable uncertainty. In particular, our algorithmic insights on the resource augmentation setting give hope for tackling such problems even if the corresponding offline versions have strong impossibility results. For knapsack specifically, studying a stochastic version of the prefix problem of Section 3, for example in the stochastic setting of [26], seems like a logical next step towards algorithmic results for the non-offline KnapExp. For OfflineKnapExp, our results of Section 3 show that non-trivial algorithmic results with theoretical guarantees are possible, opening the door for more research on finding the best-possible guarantees.

References

1 Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach.* Cambridge University Press, 2009. URL: `http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264`.

2 Evripidis Bampis, Konstantinos Dogeas, Thomas Erlebach, Nicole Megow, Jens Schlöter, and Amitabh Trehan. Competitive query minimization for stable matching with one-sided uncertainty. In *APPROX/RANDOM*, volume 317 of *LIPIcs*, pages 17:1–17:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. `doi:10.4230/LIPIcs.APPROX/RANDOM.2024.17`.

3 Evripidis Bampis, Christoph Dürr, Thomas Erlebach, Murilo Santos de Lima, Nicole Megow, and Jens Schlöter. Orienting (hyper)graphs under explorable stochastic uncertainty. In *ESA*, volume 204 of *LIPIcs*, pages 10:1–10:18, 2021. `doi:10.4230/LIPIcs.ESA.2021.10`.

4 Soheil Behnezhad, Avrim Blum, and Mahsa Derakhshan. Stochastic vertex cover with few queries. In *SODA*, pages 1808–1846. SIAM, 2022. `doi:10.1137/1.9781611977073.73`.

5 Soheil Behnezhad, Mahsa Derakhshan, and MohammadTaghi Hajiaghayi. Stochastic matching with few queries: (1-ϵ) approximation. In *STOC*, pages 1111–1124. ACM, 2020. `doi:10.1145/3357713.3384340`.

6 Avrim Blum, John P. Dickerson, Nika Haghtalab, Ariel D. Procaccia, Tuomas Sandholm, and Ankit Sharma. Ignorance is almost bliss: Near-optimal stochastic matching with few queries. *Oper. Res.*, 68(1):16–34, 2020. `doi:10.1287/opre.2019.1856`.

7 R. Bruce, M. Hoffmann, D. Krizanc, and R. Raman. Efficient update strategies for geometric computing with uncertainty. *Theory of Computing Systems*, 38(4):411–423, 2005. doi:10.1007/s00224-004-1180-4.

8 George Charalambous and Michael Hoffmann. Verification problem of maximal points under uncertainty. In *IWOCA 2013*, volume 8288 of *Lecture Notes in Computer Science*, pages 94–105. Springer, 2013. doi:10.1007/978-3-642-45278-9_9.

9 Shaddin Dughmi, Yusuf Hakan Kalayci, and Neel Patel. On sparsification of stochastic packing problems. In *ICALP*, volume 261 of *LIPIcs*, pages 51:1–51:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.51.

10 T. Erlebach and M. Hoffmann. Minimum spanning tree verification under uncertainty. In D. Kratsch and I. Todinca, editors, *WG 2014: International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 8747 of *Lecture Notes in Computer Science*, pages 164–175. Springer Berlin Heidelberg, 2014. doi:10.1007/978-3-319-12340-0_14.

11 T. Erlebach, M. Hoffmann, and F. Kammer. Query-competitive algorithms for cheapest set problems under uncertainty. *Theoretical Computer Science*, 613:51–64, 2016. doi:10.1016/j.tcs.2015.11.025.

12 T. Erlebach, M. Hoffmann, D. Krizanc, M. Mihal'ák, and R. Raman. Computing minimum spanning trees with uncertainty. In *STACS'08: 25th International Symposium on Theoretical Aspects of Computer Science*, pages 277–288, 2008. doi:10.48550/arXiv.0802.2855.

13 Thomas Erlebach, Murilo S. de Lima, Nicole Megow, and Jens Schlöter. Sorting and hypergraph orientation under uncertainty with predictions. In *IJCAI*, pages 5577–5585. ijcai.org, 2023. doi:10.24963/ijcai.2023/619.

14 Thomas Erlebach, Murilo Santos de Lima, Nicole Megow, and Jens Schlöter. Learning-augmented query policies for minimum spanning tree with uncertainty. In *ESA*, volume 244 of *LIPIcs*, pages 49:1–49:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.49.

15 T. Feder, R. Motwani, L. O'Callaghan, C. Olston, and R. Panigrahy. Computing shortest paths with uncertainty. *Journal of Algorithms*, 62(1):1–18, 2007. doi:10.1016/j.jalgor.2004.07.005.

16 T. Feder, R. Motwani, R. Panigrahy, C. Olston, and J. Widom. Computing the median with uncertainty. *SIAM Journal on Computing*, 32(2):538–547, 2003. doi:10.1137/S0097539701395668.

17 Michel X. Goemans and Jan Vondrák. Covering minimum spanning trees of random subgraphs. *Random Struct. Algorithms*, 29(3):257–276, 2006. doi:10.1002/rsa.20115.

18 Marc Goerigk, Manoj Gupta, Jonas Ide, Anita Schöbel, and Sandeep Sen. The robust knapsack problem with queries. *Comput. Oper. Res.*, 55:12–22, 2015. doi:10.1016/j.cor.2014.09.010.

19 M. M. Halldórsson and M. S. de Lima. Query-competitive sorting with uncertainty. In *MFCS*, volume 138 of *LIPIcs*, pages 7:1–7:15, 2019. doi:10.4230/LIPIcs.MFCS.2019.7.

20 S. Kahan. A model for data in motion. In *STOC'91: 23rd Annual ACM Symposium on Theory of Computing*, pages 265–277, 1991. doi:10.1145/103418.103449.

21 Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

22 Jon Kleinberg and Eva Tardos. *Algorithm Design.* Addison-Wesley Longman Publishing Co., Inc., USA, 2005.

23 Takanori Maehara and Yutaro Yamaguchi. Stochastic packing integer programs with few queries. *Mathematical Programming*, 182(1):141–174, 2020. doi:10.1007/s10107-019-01388-x.

24 Corinna Mathwieser and Eranda Çela. Special cases of the minimum spanning tree problem under explorable edge and vertex uncertainty. *Networks*, 83(3):587–604, 2024. doi:10.1002/net.22204.

25 N. Megow, J. Meißner, and M. Skutella. Randomization helps computing a minimum spanning tree under uncertainty. *SIAM Journal on Computing*, 46(4):1217–1240, 2017. `doi:10.1137/16M1088375`.

26 Nicole Megow and Jens Schlöter. Set selection under explorable stochastic uncertainty via covering techniques. In *IPCO*, volume 13904 of *Lecture Notes in Computer Science*, pages 319–333. Springer, 2023. `doi:10.1007/978-3-031-32726-1_23`.

27 J. Meißner. *Uncertainty Exploration: Algorithms, Competitive Analysis, and Computational Experiments*. PhD thesis, Technischen Universität Berlin, 2018. `doi:10.14279/depositonce-7327`.

28 Arturo Merino and José A. Soto. The minimum cost query problem on matroids with uncertainty areas. In *ICALP*, volume 132 of *LIPIcs*, pages 83:1–83:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. `doi:10.4230/LIPIcs.ICALP.2019.83`.

29 Jens Schlöter. On the complexity of knapsack under explorable uncertainty: Hardness and algorithms, 2025. `doi:10.48550/arXiv.2507.02657`.

30 Amnon Ta-Shma, Christopher Umans, and David Zuckerman. Loss-less condensers, unbalanced expanders, and extractors. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 143–152, 2001. `doi:10.1145/380752.380790`.

31 Christopher Umans. Hardness of approximating sigma$_2{}^{\mathrm{P}}$ minimization problems. In *FOCS*, pages 465–474. IEEE Computer Society, 1999. `doi:10.1109/SFFCS.1999.814619`.

32 Christopher Umans. The minimum equivalent dnf problem and shortest implicants. *Journal of Computer and System Sciences*, 63(4):597–611, 2001. `doi:10.1006/jcss.2001.1775`.

33 Jan Vondrák. Shortest-path metric approximation for random subgraphs. *Random Struct. Algorithms*, 30(1-2):95–104, 2007. `doi:10.1002/rsa.20150`.