Budget-Feasible Mechanism Design: Simpler, Better Mechanisms and General Payment Constraints

Rian Neogi
Dept. of Combinatorics and Optimization, University of Waterloo, Canada

Kanstantsin Pashkovich
Dept. of Combinatorics and Optimization, University of Waterloo, Canada

Chaitanya Swamy
Dept. of Combinatorics and Optimization, University of Waterloo, Canada

Abstract

In *budget-feasible mechanism design*, a buyer wishes to procure a set of items of maximum value from self-interested rational players. We are given an item-set U and a nonnegative valuation function $v : 2^U \mapsto \mathbb{R}_+$. Each item e is held by a player who incurs a *private cost* c_e for supplying item e. The goal is to devise a *truthful mechanism* such that the total payment made to the players is at most some given budget B, and the value of the set returned is a good approximation to $OPT := \max\{v(S) : c(S) \leq B,\ S \subseteq U\}$. We call such a mechanism a *budget-feasible mechanism*. More generally, there may be additional side constraints requiring that the set returned lies in some *downwards-monotone family* $\mathcal{I} \subseteq 2^U$. Budget-feasible mechanisms have been widely studied, but there are still significant gaps in our understanding of these mechanisms, both in terms of what kind of oracle access to the valuation is required to obtain good approximation ratios, and the best approximation ratio that can be achieved.

We substantially advance the state of the art of budget-feasible mechanisms by devising mechanisms that are *simpler*, and also *better*, both in terms of requiring weaker oracle access *and* the approximation factors they obtain. For XOS valuations, we devise the *first polytime* $O(1)$*-approximation budget-feasible mechanism using only demand oracles*, and also significantly improve the approximation factor. For subadditive valuations, we give the first explicit construction of an $O(1)$-approximation mechanism, where previously only an existential result was known.

We also introduce a fairly rich class of mechanism-design problems that we dub using the umbrella term *generalized budget-feasible mechanism design*, which allow one to capture payment constraints that are much-more nuanced than a single constraint on the total payment doled out. We demonstrate the versatility of our ideas by showing that our constructions can be adapted to yield approximation guarantees in such general settings as well.

A prominent insight to emerge from our work is the usefulness of a property called *nobossiness*, which allows us to nicely *decouple* the truthfulness + approximation, and budget-feasibility requirements. Some of our constructions can be viewed as *reductions* showing that an $O(1)$-approximation budget-feasible mechanism can be obtained provided we have a (randomized) truthful mechanism satisfying nobossiness that returns a (random) feasible set having (expected) value $\Omega(OPT)$.

2012 ACM Subject Classification Theory of computation → Algorithmic mechanism design; Theory of computation → Approximation algorithms analysis; Theory of computation → Discrete optimization

Keywords and phrases Algorithmic mechanism design, Approximation algorithms, Budget-feasible mechanisms

Digital Object Identifier 10.4230/LIPIcs.ITCS.2024.84

Funding *Chaitanya Swamy*: Supported in part by NSERC grant 327620-09.

© Rian Neogi, Kanstantsin Pashkovich, and Chaitanya Swamy;
licensed under Creative Commons License CC-BY 4.0

1 Introduction

The typical setup in *budget-feasible mechanism design* involves a ground set U of elements or items, a *valuation function* $v : 2^U \mapsto \mathbb{R}_+$ satisfying $v(\emptyset) = 0$, where $v(S)$ specifies the value obtained from set $S \subseteq U$, and a budget B. The mechanism designer (or buyer) seeks to procure a maximum-value set of elements, where each element e is held by a strategic player who incurs a *private cost* $c_e \geq 0$ for supplying item e. We use the terms player, element, and item interchangeably. In order to incentivize players to reveal their true costs, one needs to make suitable *payments* to the players. The utility of a player is then equal to the (payment received by it) − (cost incurred by it). A *truthful mechanism* is one where each player maximizes her utility by revealing her true cost and thereby has no incentive to misrepresent her true cost; an *individually-rational* mechanism is one where the utility of every truthful player is nonnegative.

The goal is to devise a truthful, individually rational mechanism that maximizes $v(S)$ subject to the *budget-feasibility* constraint that the total payment made by the mechanism is at most a given budget B. We call a truthful mechanism satisfying budget feasibility, a *budget-feasible mechanism*. We often refer to players in the output set as "winners". We say that a budget-feasible mechanism achieves approximation ratio α if it always returns a set S with $v(S) \geq OPT/\alpha$, where the benchmark $OPT := \max\{v(S) : S \subseteq U,\ c(S) \leq B\}$ is the optimal value that can be obtained if the costs were public information. (Note that $\alpha \geq 1$.) We say that a randomized mechanism is budget-feasible if it satisfies the truthfulness, individual-rationality, and budget-feasibility conditions with probability 1, i.e., it is a distribution over deterministic budget-feasible mechanisms; it achieves approximation ratio α if the expected value of the set returned is always at least OPT/α. (A distribution over truthful mechanisms is called a *universally-truthful mechanism*.)

Budget feasible mechanisms were introduced by Singer [24] and have since been extensively studied (see, e.g., [10, 26, 22, 7] and the references therein). Unlike the situation with the algorithmic problem of computing a good approximation to OPT, due to the budget-feasibility condition on payments, even the *existence* of "good" (i.e., $O(1)$-approximation) budget-feasible mechanisms (bereft of computational concerns) is not guaranteed; therefore even the development of good budget-feasible mechanisms setting aside computational considerations is of interest, and has been the focus of a significant body of work.

Despite their extensive study, our understanding of budget-feasible mechanisms is still sketchy in various respects, especially for valuation classes that are more general than submodular valuations, such as XOS and subadditive valuations. (v is subadditive if $v(S \cup T) \leq v(S) + v(T)$ for all $S, T \subseteq U$; it is XOS if it is the maximum of a collection of additive valuations.) In particular, for XOS and subadditive valuations, the state-of-the-art for budget-feasible mechanisms lags significantly behind that of the *algorithmic* problem of computing a good approximation to OPT, both *qualitatively*, in terms of what kind of oracle access to the valuation is required to obtain a good approximation ratio, and *quantitatively*, in terms of the best approximation ratio that can be achieved. The algorithmic problem admits a (polytime) $(2+\epsilon)$-approximation, even for subadditive valuations, using demand oracles [6], while value oracles are insufficient to achieve anything better than $\sqrt{n}$-approximation in polytime even for XOS valuations [24]. However, *all* existing $O(1)$-approximation budget-feasible mechanisms [10, 1, 22] for XOS valuations require *both* a demand oracle and a so-called *XOS oracle*, and the approximation ratio achieved, even ignoring computational concerns, is rather large (244 [1, 3]); in particular, it was not known whether one can obtain a polytime $O(1)$-approximation using only demand oracles, as with the algorithmic problem.

More broadly speaking, we have limited understanding of how truthful mechanism design interfaces with conditions on payments. While budget-feasible mechanism design considers a single constraint on the total payment doled out by the mechanism, it is not hard to envision settings involving more nuanced or complex payment constraints. For instance, players may be divided into certain groups (based on geographical, political, or other factors), and one may require that the total payment made to (the winners in) each group should fall within a budget. Another scenario involves economic aid: the government may offer economic relief to the buyer by deeming that it be responsible for only the ℓ largest payments (for some parameter ℓ), so that the budget constraint now translates to the Top-ℓ *constraint* that the sum of the ℓ largest payments should be at most the budget. To our knowledge, such *generalized budget-feasible mechanism design* problems have not been considered; moreover, as illustrated by our partial knowledge of budget-feasible mechanisms, we have quite limited knowledge of techniques for approaching generalized budget-feasible mechanism design.

Our contributions and results. Our contributions are twofold. We make substantial progress towards remedying the gap between the budget-feasible mechanism-design problem and the algorithmic problem (of approximating OPT), and in doing so obtain illuminating insights into the relationship between these two problems. We devise mechanisms that are fairly *simple* (to describe and analyze), and also *better* than the current state-of-the-art, both in terms of requiring weaker oracle access to the valuation, and the significantly better approximation factors they obtain. For XOS valuations, our results establish that polytime $O(1)$-approximation can be achieved given only demand oracles, while for subadditive valuations, we give an explicit construction of an $O(1)$-approximation mechanism. Second, we initiate a study of generalized budget-feasible mechanism design, and take important strides towards the development of techniques for truthful mechanism design in the presence of a broader set of payment constraints. We showcase the broad applicability of our ideas by demonstrating that our budget-feasible mechanisms can be adapted to yield guarantees for both group budget constraints, and Top-ℓ budget constraints.

We consider a more general setup wherein we may have additional side-constraints imposing that the set returned lie in some publicly-known *downwards-monotone family* $\mathcal{I} \subseteq 2^U$ of sets: in such a family, also called an *independence system*, if $S \in \mathcal{I}$ and $T \subseteq S$ then $T \in \mathcal{I}$. (Of course, our benchmark OPT is now $\max\{v(S) : S \in \mathcal{I},\ c(S) \leq B\}$.) Our mechanisms chiefly utilize two types of oracles.

- *Generalized demand oracle*: given prices $q \in \mathbb{R}^U_+$, return $\operatorname{argmax}\{v(S) - \sum_{e\in S} q_e : S \in \mathcal{I}\}$.
- *Knapsack-cover oracle*: given prices $q \in \mathbb{R}^U_+$ and a target Val, return $\operatorname{argmin}\{\sum_{e\in S} q_e : v(S) \geq Val,\ S \in \mathcal{I}\}$ (or determine infeasibility); we use *generalized knapsack-cover* to refer to the underlying optimization problem. (The terminology stems from the fact that for additive v, $\mathcal{I} = 2^U$, this optimization problem is called the knapsack-cover problem [12], or sometimes the covering knapsack problem.)

Both oracles have similar economic motivation: a generalized demand oracle is a demand oracle restricted to the sets one is *allowed* to procure, and represents the best set one would procure under the given prices; a knapsack cover oracle also captures this, but under the constraint that the set procured achieves a target value.

Side constraints modeled by a downwards-monotone family $\mathcal{I}$ (of varying generality) have been explicitly considered in some prior work; see, e.g., [2, 4, 22, 20]. Observe that it is possible to fold the downwards-monotone family $\mathcal{I}$ into the valuation function, by defining the new valuation function v', where $v'(S) = \max_{T\subseteq S: T\in\mathcal{I}} v(T)$. It is not hard to see that: (1) a {generalized demand, knapsack-cover} oracle for v is equivalent to having the same

oracle for v' but without any side constraints (i.e., with $\mathcal{I} = 2^U$); and (2) if v is XOS or subadditive, then so is v'. Therefore, existing results on XOS and subadditive functions also apply to the above setup with side constraints (given suitable access to v'). We explicitly list the side constraints in order to decouple, and better understand, the complexity arising from the valuation function v, and the complexity due to the side constraints.

We now state our main results. (Recall that a randomized budget-feasible mechanism satisfies truthfulness, individual rationality, and budget feasibility with probability 1.) For XOS-valuations, we devise a variety of *polynomial-time mechanisms* whose guarantees depend on the type of oracle access available.

▶ **Theorem 1.1.** *For any downwards-monotone family and any XOS valuation function v, we obtain polytime randomized budget-feasible mechanisms with the following guarantees, for any $\epsilon > 0$:*

(a) 28*-approximation given a generalized demand oracle for v (Theorem 3.1);*

(b) $(41 + \epsilon)$*-approximation given a knapsack-cover oracle for v (Theorem 3.4);*

(c) $(33 + \epsilon)$*-approximation given knapsack-cover and XOS oracles for v (Section 3.2).*

Part (a) of Theorem 1.1 yields the *first polytime $O(1)$-approximation budget-feasible mechanism using only demand oracles*, thereby placing the mechanism-design problem qualitatively on par with the algorithmic problem in terms of access required to the valuation function. The approximation ratios we achieve are all much better than the previous-best 244-approximation for XOS valuations [1], which is obtained by changing some parameters of a mechanism in [10]. But perhaps more significantly, our mechanisms are also simpler to describe and analyze, and this simplicity enables us to extract some illuminating insights that paint a clearer picture of the complexity of budget-feasible mechanism-design versus that of the algorithmic problem of approximating OPT.

Knapsack-cover oracle is a new oracle that we introduce in this work. Other than the economic motivation behind this oracle, as we discuss below, in the context of budget-feasible mechanism design, the use of a knapsack-cover oracle is quite well-motivated from a computational perspective, and it also affords us a good deal of flexibility in mechanism design. As noted earlier, due to the budget-feasibility condition on payments, even the *existence* of good budget-feasible mechanisms is a priori unclear. A natural direction, and a standard one followed in complexity theory, for understanding how much more complex one class of problems is relative to another class of problems Π is *oracle complexity*: what can be accomplished assuming we have an oracle for solving problems in Π? In budget-feasible mechanism-design, this amounts to having an oracle for the algorithmic problem of computing OPT. It is easy to see that an algorithm for computing OPT can always be used to supply a knapsack-cover oracle via a binary search on the budget. Thus, part (b) of Theorem 1.1 demonstrates that *budget-feasibility can always be enforced given an oracle for the underlying algorithmic problem*, which is an illuminating insight that delineates the complexity of budget-feasible mechanism design versus that of the algorithmic problem.

Note that generalized knapsack-cover (i.e., the optimization problem solved by knapsack-cover oracle) is a social-cost minimization problem, and part (b) can also be viewed as showing that the VCG mechanism, which is a truthful, optimal mechanism for generalized knapsack-cover, can be transformed in a black-box fashion to satisfy budget feasibility with an $O(1)$-factor loss in approximation (with respect to OPT). This transformation (see Algorithm XOS-Alg-KC in Section 3.2, which leads to Theorem 1.1 (b)) is in fact *approximation-friendly* (see Remark 3.7): we do not actually need to solve generalized knapsack-cover exactly, and can instead work with an *approximate knapsack-cover oracle*

(and even a weaker approximate feasibility oracle) provided that this approximate oracle satisfies: (i) *monotonicity*, the property that a winner remains a winner upon decreasing her cost, which characterizes truthfulness in single-dimensional domains; and (ii) a property called *nobossiness*, which means that fixing other players' costs c_{-e}, if a player e is a winner under inputs (c_e, c_{-e}) and (c'_e, c_{-e}), then the winner set stays the same for both inputs. Thus, an interesting insight to emerge from Algorithm XOS-Alg-KC is that we obtain *a reduction from budget-feasible mechanism design to the design of truthful, nobossy, approximation-mechanisms for generalized knapsack-cover*. This serves to nicely decouple the approximation + truthfulness, and budget-feasibility goals for XOS valuations (while introducing a nobossiness requirement).

For subadditive valuations, we obtain the following result, which yields the first *explicit* construction of an $O(1)$-approximation budget-feasible mechanism.

▶ **Theorem 1.2** (Proved in Section 4). *For any downwards-monotone family and any subadditive valuation function, we can obtain a randomized* 33*-approximation budget-feasible mechanism.*

We obtain Theorem 1.2 by distilling and refining a key insight from the existential 512-approximation for subadditive valuations in [10] that allows us to reduce the budget-feasible mechanism-design task to that of computing a suitable distribution over cost vectors (which can always be computed in exponential time), which leads to a very simple mechanism. For *cardinality-based subadditive functions* (i.e., $v(S)$ depends only on $|S|$), we obtain a *polytime* 69-approximation budget-feasible mechanism.

In Section 5, we consider generalized budget-feasible mechanism design, wherein we may have more complex constraints on the payments handed out by the mechanism. We consider two types of payment constraints: *group budget constraints*, wherein players may be divided into (potentially overlapping) groups, and we have a budget constraint on the total payment made to each group; (2) Top-ℓ *budget constraints*, wherein we have budget constraints on the sum of the ℓ largest payments doled out by the mechanism, for one or more indices $\ell \in [n]$. We highlight the versatility of our ideas by showing that they can be leveraged to obtain $O(k)$-approximation with k group budget constraints, or k Top-ℓ budget constraints, thereby nicely generalizing our results for budget-feasible mechanisms (where $k = 1$).

▶ **Theorem 1.3.** *For any downwards-monotone family, we obtain randomized truthful mechanisms for generalized budget-feasible mechanism design with the following guarantees.*

(a) *$O(k)$-approximation for subadditive valuations with k group budget constraints; for XOS valuations, we obtain a polytime mechanism with this guarantee given a generalized demand oracle (Theorem 5.1)*

(b) *$O(k)$-approximation for subadditive valuations with k Top-ℓ budget constraints, via a polytime reduction to budget-feasible mechanism design (Theorem 5.3).*

Technical overview. We may assume that $\{e\} \in \mathcal{I}$ and $c_e \leq B$ for all $e \in U$. Our mechanism for XOS valuations in part (a) of Theorem 1.1 using a generalized demand oracle is particularly simple to state and analyze. It consists of essentially three steps (see also Algorithm XOS-Alg-GD in Section 3.1). We first obtain an estimate of OPT by randomly partitioning the elements into two sets U_1, U_2, and computing an estimate V_1 of the optimum when we are restricted to only use items from U_1. We let V_1 be the optimal value of an LP-relaxation (BudgP) of the problem (see Section 2) that is restricted to only use items in U_1, which can be solved efficiently using a generalized demand oracle. Next, we obtain

a good candidate set $S^* \subseteq U_2$ by (roughly speaking) running a generalized demand oracle with prices $q_e = \frac{\lambda \cdot V_1}{B} \cdot c_e$, for a suitable value $\lambda \in [0,1]$. Finally, considering elements of S^* in some fixed order, we compute a maximal prefix $T \subseteq S^*$ having value at most λV_1. (In the actual algorithm, we replace λV_1 by a slightly different quantity in both steps, to fine-tune the approximation factor.) We return T with some probability p and $e^* := \operatorname{argmax}_{e \in U} v(e)$ with probability $1-p$. (This random choice is for technical reasons; for intuition, it suffices to focus on the outcome where we return T.)

The analysis is equally simple. For single-dimensional mechanism design, it is well known [23] that an algorithm f can be combined with suitable payments to obtain a truthful mechanism iff f is monotone, and in this case, the payment of a winning player is the threshold value at which it transitions from a winner to a non-winner (see Theorem 2.4). The key observation is that *the generalized-demand-oracle computation already yields suitable thresholds*: we obtain that $c_e \leq \frac{B}{\lambda V_1} \cdot \big(v(S^*) - v(S^* - e)\big)$ for all $e \in S^*$, which yields $\frac{B}{\lambda V_1} \cdot \big(v(S^*) - v(S^* - e)\big)$ as an upper bound on the threshold value of a winner e. For XOS valuations, $\sum_{e \in T} \big(v(S^*) - v(S^* - e)\big) \leq v(T)$ (Claim 2.5), and so by our choice of T, we obtain budget-feasibility and truthfulness. For the approximation guarantee, Bei et al. [10] show the intuitive result that random partitioning can be used to well-estimate OPT as also the optimum from U_2 (see Lemma 2.8). We generalize this result to show that the LP-optimum is also well-estimated (Lemma 2.9). We can assume therefore that V_1 is roughly $\Omega(\mathsf{LP}^*)$, where $\mathsf{LP}^* \geq OPT$ is the LP optimum considering the entire ground set U. The LP-relaxation for item-set U_2 yields a distribution over sets having large value (i.e., $\Omega(\mathsf{LP}^*)$) and cost at most B, and its optimal solution can be used to infer that $v(S^*) - q(S^*) = \Omega(\mathsf{LP}^*)$. Since both V_1 and $v(S^*)$ are large, this immediately implies that $v(T) = \Omega(\mathsf{LP}^*)$. (More precisely, V_1 and $v(S^*)$ are $\Omega\big(\mathsf{LP}^* - v(e^*)\big)$ and this is why we return e^* with some probability.)

We are substantially aided in our analysis by the fact that we can focus on a single winner-set T when considering a winner $e \in T$ and changes to c_e under which e remains a winner. This is the *nobossiness* property mentioned earlier, and it follows because one may assume that the generalized-demand oracle breaks ties consistently, so that we obtain the same sets S^* and T if any winner $e \in T$ decreases her cost.

Algorithm XOS-Alg-KC for XOS valuations (Section 3.2), which uses a knapsack-cover oracle and leads to part (b) of Theorem 1.1, utilizes similar ideas. We now let V_1 be a $(1+\epsilon)$-approximate estimate of the integer optimum from U_1, which is easy to obtain using a knapsack-cover oracle; again, this well-approximates OPT as also the integer optimum from U_2. We now use a knapsack-cover oracle with price-vector c and target value V_1 to obtain S^*, and one can assume that $c(S^*) \leq B$. The knapsack-cover oracle does not directly yield suitable threshold values (but it does yield some thresholds), so we now postprocess S^* by dropping elements as needed to ensure that every e in the remainder set A satisfies $c_e \leq \frac{B}{\lambda V_1} \cdot \big(v(A) - v(A-e)\big)$. This computation preserves monotonicity, and since $c(S^*) \leq B$, one can argue that this does not drop the value by much. We now return a maximal prefix $T \subseteq A$ with $v(T) \leq \lambda V_1$, and given the above thresholds, budget-feasibility follows as before.

Observe that it is not so important for the above analysis that we have an exact knapsack oracle: approximation factors in $v(S^*)$ and $c(S^*)$ are easily accommodated by choosing the parameter λ suitably. However, what is *crucial* is that the knapsack-cover oracle satisfies monotonicity and nobossiness. But as long as we have these two properties, even an approximate knapsack-cover oracle would suffice. Furthermore, we only care about obtaining some set S^* with $c(S^*) \leq B$ and $v(S^*) \geq V_1$, and so we can work with an approximate, monotone, nobossy feasibility oracle for this decision version of generalized knapsack-cover. (There are settings where such an approximate feasibility oracle is easy to obtain in polytime, whereas computing even an approximate generalized demand oracle is quite intractable.)

For subadditive valuations, we utilize the following powerful tool (Lemma 4.1): for any set $S \subseteq U$ and any finite set $K \subseteq \mathbb{R}_+^S$, there exists a distribution $\mathcal{D}$ over K such that, $\mathbb{E}_{\tau\sim\mathcal{D}}\big[v(\{e \in S : c_e \leq \tau_e\})\big] \geq 0.5 \cdot v(S)$ for any $c \in K$. Given this, we start off as in the algorithm for XOS valuations with a knapsack-cover oracle, to obtain a set S^* (in a monotone, nobossy fashion) with $c(S^*) \leq B$, and where $v(S^*) = \Omega(OPT)$. Now, we simply consider the set $K = K_{S^*} := \big\{x \in \mathbb{Z}_+^{S^*} : \sum_{e\in S^*} x_e \leq B\big\}$, sample a random vector τ from the distribution guaranteed by the above statement for K, and return $\{e \in S^* : c_e \leq \tau_e\}$. Since $\tau \in K$, this immediately yields truthfulness and budget feasibility, and the approximation guarantee follows because $c \in K$, and so the expected value obtained is at least $v(S^*)/2$.

In fact, this distribution-based template is quite powerful. Even with general payment constraints, it shows that if one can obtain a set S^* of large value via a monotone, nobossy algorithm, whose induced cost vector $c|_{S^*} := (c_e)_{e\in S^*}$ satisfies the payment constraints, then one can prune S^* using a random threshold vector sampled from a suitable distribution and obtain a truthful mechanism satisfying the payment constraints (see Theorem 5.2). Thus, we are again able to decouple the truthfulness + approximation, and budget-feasibility requirements, now for subadditive valuations, with the addition of nobossiness. This leads to the $O(k)$-approximation for subadditive valuations with group budget constraints.

An important takeaway from our simple and general constructions is that the underlying insights – e.g., computing a set that maximizes $v(S) - \frac{\lambda OPT}{B} \cdot c(S)$ followed by a pruning operation suffices for budget feasibility – may potentially be exported to other problem domains and allow us to tackle budget-feasibility constraints in other settings.

Other related work. Following the work of Singer [24], which introduced budget-feasible mechanism design and gave a randomized $O(1)$-approximation budget-feasible mechanism for monotone submodular valuations, various improvements in the approximation factor, as also deterministic mechanisms, were obtained for monotone submodular valuations by [15, 21]. The deterministic mechanism in [15] was not efficient, but has been modified to yield efficient mechanisms for various structured submodular functions such as coverage functions [2, 25], and information gain functions [19]. More recently, various works have developed $O(1)$-approximation mechanisms for nonmonotone submodular valuations [1, 4, 20]. The current-best approximation factors for monotone and non-monotone submodular valuations are due to [7]. All of these results utilize only a value oracle for accessing the submodular valuation.

With XOS valuations, value oracles cannot yield any bounded approximation [24], and so work on XOS and more general valuations has considered demand oracles, possibly in combination with other types of oracles. Dobzinski et al. [16] seem to have been the first to consider more general valuations, and they gave an $O(\log^2 n)$-approximation for subadditive valuations using demand oracles. Bei et al. [10, 11] made an important advance by providing the first $O(1)$-approximation mechanism for XOS valuations. Their mechanism has essentially remained the only mechanism for XOS valuations (its analysis has been improved by [3, 22]), and they require both a demand oracle and a so-called XOS oracle. Thus, prior to our work, there was no known $O(1)$-approximation budget-feasible mechanism for XOS valuations using only demand oracles. Bei et al. also considered a very general Bayesian setting, and devised an $O(1)$-approximation mechanism for subadditive valuations. In the journal version of their paper [11], they observed that their result for the Bayesian setting also implies the *existence* of a worst-case $O(1)$-approximation mechanism for subadditive valuations, using Yao's minimax principle. They also devised a polytime randomized $O\big(\frac{\log n}{\log\log n}\big)$-approximation for subadditive valuations; more recently, this guarantee has also been obtained deterministically [7].

Side constraints modeled by a downwards-monotone family $\mathcal{I}$ were explicitly considered by [2, 22, 4, 20], and the most general setting that has been considered is that of (non-monotone) submodular functions where $\mathcal{I}$ is a *p-system* (any two bases of a set differ in size by a multiplicative factor of at most p), for which there is a polytime $O(p)$-approximation budget-feasible mechanism [4].

Budget-feasible mechanism design has also been investigated under the large market assumption, where no single player has a large value (see, e.g., [5, 8, 21]), and in online settings [4, 9].

Future directions. One prominent insight to emerge from our work is the usefulness of the nobossiness property, which as we have seen allows us to nicely *decouple* the truthfulness + approximation, and budget-feasibility requirements. (Nobossines was also exploited by [17] to obtain a similar decoupling result in the context of cost-sharing mechanisms.) As already noted, some of our constructions can be viewed as reductions showing that an $O(1)$-approximation budget-feasible mechanism can be obtained provided we have a (randomized) truthful mechanism satisfying nobossiness that returns a (possibly random) feasible set having (expected) value $\Omega(OPT)$; in other words, budget-feasibility can always be injected into such a mechanism losing a small factor in the approximation. We believe that understanding the power and limitations of this intriguing property is a pertinent research question, and leave this for future work.

Generalized budget-feasible mechanism design is a research direction that is teeming with open questions. Even the algorithmic questions here are not well-understood, even in structured settings such as group budget constraints with disjoint groups. Obtaining a toolkit of techniques capable of handling nuanced payment constraints would constitute an important advance in mechanism design, and is an important direction for future work.

2 Preliminaries

We use $\mathbb{R}_+$ and $\mathbb{Z}_+$ to denote the set of nonnegative reals, and nonnegative integers respectively. For an integer $k \geq 1$, let $[k]$ denote $\{1, \ldots, k\}$. Throughout, $OPT := \max\{v(S) : S \in \mathcal{I},\ c(S) \leq B\}$ denotes the optimal value, and we use O^* to denote some fixed optimal set. Also, n denotes $|U|$. For notational convenience, we assume that items in U are labeled $1, 2, \ldots, n$; when we say item e, we mean the item labeled e, and so U is essentially $[n]$. Given $w \in \mathbb{R}^U$ and a set $S \subseteq U$, we use $w(S)$ to denote $\sum_{e \in S} w_e$.

We may assume that every element $e \in U$ satisfies $c_e \leq B$ and $\{e\} \in \mathcal{I}$. Otherwise, we can preprocess the input by discarding elements that do not satisfy this property. Note that this does not degrade the approximation ratio, and does not impact truthfulness, individual rationality, and budget feasibility. Throughout, we use e^* to denote $\operatorname{argmax}_{e \in U} v(e)$. Note that by the above preprocessing, we have $OPT \geq v(e^*)$.

All omitted proofs appear in the full version of the paper.

Valuation functions and oracles. Let $v : 2^U \mapsto \mathbb{R}_+$ be a valuation function. We will always assume that v is normalized, i.e., $v(\emptyset) = 0$. For an element $e \in U$ and set $S \subseteq U$, we use $v(e)$ as a shorthand for $v(\{e\})$, and $S + e$ and $S - e$ to denote $S \cup \{e\}$ and $S \setminus \{e\}$ respectively. We say that v is *monotone*, if $v(S) \leq v(T)$ whenever $S \subseteq T \subseteq U$. We consider various classes of valuation functions. We say that v is:

- *additive* (or *modular*), if there exists some $a \in \mathbb{R}^U$ such that $v(S) = a(S)$ for all $S \subseteq U$. Note that an additive valuation is monotone iff it is nonnegative.

- *submodular*, if $v(S) + v(T) \geq v(S \cap T) + v(S \cup T)$ for all $S, T \subseteq U$.
- *XOS*, if v is the maximum of a finite collection of additive functions, i.e., there exist $a^1, \ldots, a^k \in \mathbb{R}^U$ such that $v(S) = \max_{i \in [k]} a^i(S)$ for all $S \subseteq U$. Note that we allow the additive functions to be negative, and this allows us to capture non-monotone XOS functions. The above definition is equivalent to saying that for every $S \subseteq U$, there exists some $w \in \mathbb{R}^U$ such that $v(S) = w(S)$ and $v(T) \geq w(T)$ for all $T \subseteq S$; we say that w *(or the corresponding additive valuation) supports* S. XOS valuations are also called *fractionally subadditive* valuations, and can be equivalently defined in terms of fractional covers. A *fractional cover* of a set $S \subseteq U$ is a collection $\{\mu_T\}_{T \subseteq S}$ such that $\sum_{T \subseteq S: e \in T} \mu_T = 1$ for all $e \in S$, and its value is defined as $\sum_{T \subseteq S} v(T)\mu_T$. We say that v is fractionally subadditive, if for every $S \subseteq U$, every fractional cover of S has value at least $v(S)$. Using LP duality, it is not hard to infer that this is equivalent to stating that every $S \subseteq U$ has a supporting additive valuation. When v is monotone and XOS, we can assume that v is the maximum of a collection of nonnegative additive valuations, and we can relax the fractional-cover condition to the inequality $\sum_{T \subseteq U: e \in T} \mu_T \geq 1$ for all $e \in S$. This is because we can always ensure that equality holds here by dropping elements from sets if needed, and with a monotone valuation, this does not increase the value of the fractional cover.
- *subadditive*, if $v(S \cup T) \leq v(S) + v(T)$ for all $S, T \subseteq U$.

It is well known that additive valuations are a strict subclass of submodular valuations, which in turn form a strict subclass of both XOS and subadditive valuations. Also, *monotone* XOS valuations form a strict subclass of monotone subadditive valuations.

Our mechanisms chiefly utilize two types of oracles for accessing the valuation v: a generalized demand oracle and a knapsack-cover oracle. Both oracles take item prices $q \in \mathbb{R}^U_+$; a generalized demand oracle returns an optimal solution to $\max\{v(S) - q(S) : S \in \mathcal{I}\}$, and a knapsack-cover oracle takes also a target value *Val* and returns an optimal solution to $\min\{q(S) : v(S) \geq Val,\ S \in \mathcal{I}\}$.

While we have mentioned non-monotone valuations above, we observe that the following simple trick lets us reduce to the setting with monotone valuations. Given a valuation v, the *monotonized version of* v, denoted v^{mon}, is the function given by $v^{\mathsf{mon}}(S) = \max_{T \subseteq S} v(T)$ for all $S \subseteq U$. It is not hard to infer (see Lemma 2.1) that if v is {subadditive, XOS} then v^{mon} belongs to the same class; moreover, since the input prices to a generalized demand oracle or knapsack-cover oracle are always nonnegative, an oracle for v also yields an oracle for v^{mon}. In the sequel, we therefore focus on monotone valuation functions.

▶ **Lemma 2.1.** *Let $v : 2^U \mapsto \mathbb{R}_+$ and v^{mon} be its monotonized version. (a) If v is subadditive, then so is v^{mon}; if v is XOS, then so is v^{mon}. (b) A {generalized demand, knapsack-cover} oracle for v yields the same type of oracle for v^{mon}.*

We will often need to restrict our attention to a specific set $A \subseteq U$ when solving the optimization problems underlying our oracles (i.e., optimize among subsets of A that are in $\mathcal{I}$). For subadditive valuations, this can be easily achieved by setting q_e to be a suitably large value M for all $e \notin A$.

- For a generalized demand oracle, we can take $M = \max_{e \in U} v(e) + 1$; then for any set S with $S - A \neq \emptyset$, we have $v(S \cap A) - q(S \cap A) > v(S) - q(S)$, and so the oracle must return a subset of A as the maximizer.
- For a knapsack-cover oracle, we can take $M = |A| \cdot \max_{e \in A} q_e + 1$, which ensures that $q(S) > q(T)$ for any S, T such that $S - A \neq \emptyset$, $T \subseteq A$. So if the output S^* of the knapsack-cover oracle with these prices is not a subset of A, then we can infer that there is no feasible solution that is a subset of A.

We will also naturally require a value oracle for the valuation v. It is well known that a demand oracle can be used to supply a value oracle, and the same holds for sets $S \in \mathcal{I}$ given a generalized demand oracle. Given a knapsack-cover oracle and a set $S \in \mathcal{I}$, for a monotone valuation v, we can simply set the prices to be 0 for $e \in S$ and 1 for $e \notin S$, and keep increasing the target *Val* until the knapsack-cover oracle returns a set containing some element not in S. (For non-monotone valuations, we first move to the monotonized version of the valuation and simulate a value oracle for this monotonic function.)

Both oracles can be used to compute a good approximation to OPT, and one can show that one oracle can be used to supply an approximate version of the other oracle.

▶ **Lemma 2.2.** *Let $v : 2^U \mapsto \mathbb{R}_+$ be subadditive, $\mathcal{I} \subseteq 2^U$ be a downwards-monotone family. For any $\epsilon > 0$: (a) a knapsack-cover oracle for v can be used to compute a $(1+\epsilon)$-approximation to OPT in polytime; (b) a generalized demand oracle for v can be used to compute a $(2+\epsilon)$-approximation to OPT in polytime.*

Proof. Let $e^* = \operatorname{argmax}_{e \in U} v(e)$. Note that $OPT \geq v(e^*)$ (due to the assumptions stated at the start of this section). For part (a), we simply try all target values $Val = v(e^*)(1+\epsilon)^i$, where i is a nonnegative integer, and return the largest value (and the corresponding set) for which the knapsack cover oracle returns a set S with $c(S) \leq B$. It is clear that $v(S)$ is at least $OPT/(1+\epsilon)$. Part (b) follows from [6], by folding the family $\mathcal{I}$ into the valuation function: define v' by setting $v'(S) := \max_{T \subseteq S : T \in \mathcal{I}} v(T)$. Then v' is subadditive, a generalized demand oracle for $(v, \mathcal{I})$ yields a demand oracle for v', and $OPT = \max\{v'(S) : S \subseteq U, c(S) \leq B\}$. So using [6], we can obtain a set S with $v'(S) \geq OPT/(2+\epsilon)$. To obtain the set in $\mathcal{I}$ determining $v'(S)$, we can use a generalized demand oracle by setting 0 prices for elements in S, and very large prices (e.g., $v(e^*)+1$) for elements not in S. ◀

It will be important that our oracles use a consistent, price-independent, tie-breaking rule to choose an optimal solution in case there are multiple optimal solutions. Lemma 2.3 shows that one can suitably perturb the element prices to obtain a *lexicographically-smallest* set among all optimal solutions. Recall that item e means the item labeled e, where $e \in [n]$. Given two distinct sets $S, T \subseteq U$, we say that S is *lexicographically smaller* than T, denoted $S \prec_{\text{lex}} T$, if $\sum_{e \in S} \frac{1}{2^e} < \sum_{e \in T} \frac{1}{2^e}$.

▶ **Lemma 2.3.** *Let $q \in \mathbb{R}^U$ be some item prices, and Val be some target value. Suppose we have an integer M such that every q_e, and all $v(S)$ values are integer multiples of $\frac{1}{M}$. Define $q'_e := q_e + \frac{1}{2^e \cdot M}$ for all $e \in U$. For both generalized demand oracle and knapsack-cover oracle, the underlying optimization problem with prices q' has a unique optimal solution, which is the lexicographically-smallest set among all optimal solutions to the problem with prices q.*

Mechanism design. A (direct-revelation) *mechanism* consists of an *algorithm* or allocation rule f, and a *payment scheme* p_e for each player e. In budget-feasible mechanism design, f and the p_es receive as input the publicly-known information $(v : 2^U \mapsto \mathbb{R}_+, \mathcal{I}, B)$ and the players' reported costs $\{c_e \geq 0\}_{e \in U}$; the algorithm f outputs a set $S \in \mathcal{I}$, that we call the *winner set*, and each p_e outputs a number, which is the payment made to player e. Notationally, we will often view f and the p_es as functions of only the reported cost-vector c, treating $(v, \mathcal{I}, B)$ as implicitly fixed. The *utility* of player e, when her true private cost is $\overline{c}_e \geq 0$, she reports c_e, and the others report c_{-e}, is $u_e(\overline{c}_e; c_e, c_{-e}) := p_e(c_e, c_{-e}) - \overline{c}_e$ if $e \in f(c_e, c_{-e})$ and $p_e(c_e, c_{-e})$ otherwise, and each player aims to maximize her utility. We seek a mechanism $\mathcal{M} = \big(f, \{p_e\}_{e \in U}\big)$ satisfying the following properties.

- $\mathcal{M}$ is *truthful*: each player e maximizes her utility by reporting her true private cost: for every $\overline{c}_e, c_e, c_{-e}$, we have $u_e(\overline{c}_e; \overline{c}_e, c_{-e}) \geq u_e(\overline{c}_e; c_e, c_{-e})$.
- $\mathcal{M}$ is *individually rational* (IR): $u_e(\overline{c}_e; \overline{c}_e, c_{-e}) \geq 0$ for every e, and every $\overline{c}_e, c_{-e}$; note that this implies that $p_e(c) \geq 0$ for all c. We say that $\mathcal{M}$ makes *no positive transfers* (NPT) if $p_e(c) = 0$ if $e \notin f(c)$. In the sequel, we will always also implicitly require NPT.
- $\mathcal{M}$ is *budget feasible*: we have $\sum_e p_e(c) \leq B$ for every cost-vector c. (Assuming NPT, this is equivalent to $\sum_{e \in f(c)} p_e(c) \leq B$.) Note that if $\mathcal{M}$ is individually rational, this implies that $c\big(f(c)\big) \leq B$.

We say that $\mathcal{M}$ is an *α-approximation mechanism*, if f is an α-approximation algorithm, i.e., we have $v\big(f(c)\big) \geq OPT(c)/\alpha$ for all c, where $\alpha \geq 1$.

In budget-feasible mechanism design, where players are single-dimensional (i.e., each player's cost is a known linear function of a private real number), it follows from Myerson [23] that a mechanism $\mathcal{M} = (f, p)$ in this setting is truthful iff the algorithm f is *monotone*: for every player $e \in U$, every $c_e, c'_e \in \mathbb{R}_+$ with $c'_e \leq c_e$, and every $c_{-e} \in \mathbb{R}_+^{U-e}$, if $e \in f(c_e, c_{-e})$ then $e \in f(c'_e, c_{-e})$ (i.e., a winner remains a winner upon decreasing her reported cost).

▶ **Theorem 2.4** (Truthful mechanisms in single-dimensional domains [23]). *Given an algorithm f for budget-feasible mechanism design, there exist payment functions $\{p_e\}_{e \in U}$ such that (f, p) is a truthful mechanism iff f is monotone. Suppose that f is monotone, and $\tau_e = \tau_e(c_{-e}) := \sup\{c_e \geq 0 : e \in f(c_e, c_{-e})\}$ is finite for every $e \in U$ and $c_{-e} \in \mathbb{R}_+^{U-e}$. Then setting $p_e(c) = \tau_e(c_{-e})$ if e is a winner, and 0 otherwise, is the unique payment scheme that yields a truthful, individually-rational mechanism (satisfying NPT).*

Given this characterization of truthful mechanisms and payments, we focus on designing a monotone algorithm where the threshold values can be computed efficiently and the sum of the winners' thresholds is at most B, thereby obtaining a budget-feasible mechanism. We abuse notation slightly, and say that a monotone algorithm is budget-feasible, if the resulting mechanism is budget feasible (i.e., the sum of the winners' thresholds is at most B). Due to the preprocessing mentioned at the beginning of this section, where we discard elements e with $c_e > B$, the threshold value of every player is well-defined and is at most B.

Our mechanisms crucially exploit, and in turn satisfy, a key property called *nobossiness*: we say that an algorithm f satisfies nobossiness (or is nobossy) if for every player e and every c_{-e}, and c_e, c'_e, if $e \in f(c_e, c_{-e})$ and $e \in f(c'_e, c_{-e})$, then $f(c_e, c_{-e}) = f(c'_e, c_{-e})$. We say that a mechanism $(f, \{p_e\})$ satisfies nobossiness (or is nobossy), if f is nobossy. Monotonicity coupled with nobossiness ensures that the winner set remains unchanged when a winner decreases her cost. This is quite useful in: (a) the computation of threshold values, since we can focus on a fixed winner set, and (b) settings where the mechanism prunes the output of another algorithm g to obtain its output; monotonicity and nobossiness of g make it convenient to reason about this composition, since they ensure that the pruning procedure receives the same set as input when a winner decreases her cost.

Properties of XOS and subadditive valuations. We will utilize some well-known properties about XOS and subadditive valuations, which we collect below.

▷ Claim 2.5. Let $v : 2^U \mapsto \mathbb{R}_+$ be an XOS valuation. Then, for any subsets $T \subseteq S \subseteq U$, we have $\sum_{e \in T}\big(v(S) - v(S - e)\big) \leq v(T)$.

▷ Claim 2.6. Let $v : 2^U \mapsto \mathbb{R}_+$ be subadditive, $q \in \mathbb{R}_+^U$ be item prices, $\mathcal{I} \subseteq 2^U$ be a downwards-monotone family, and $S^* = \arg\max\{v(S) - q(S) : S \in \mathcal{I}\}$. Then, we have $v(T) \geq q(T)$ for all $T \subseteq S^*$.

Our randomized mechanisms all use a random-sampling step to compute a good approximation of OPT. Let U_1, U_2 be a random partition of U obtained by placing each element of U independently with probability $\frac{1}{2}$ in U_1 or U_2. Throughout, for $i = 1, 2$, we use $V_i^* := \max\{v(S) : S \in \mathcal{I},\ S \subseteq U_i,\ c(S) \leq B\}$ to denote the optimal value that can be achieved using only elements from U_i.

▶ **Lemma 2.7** (Random partitioning lemma [10]). *Let $g : 2^U \mapsto \mathbb{R}_+$ be a subadditive function. Consider any $S \subseteq U$. Then, $\Pr\left[g(S \cap U_1), g(S \cap U_2) \geq \frac{g(S) - \max_{e \in S} g(e)}{4}\right] \geq \frac{1}{2}$.*

▶ **Corollary 2.8.** *Let $v : 2^U \mapsto \mathbb{R}_+$ be subadditive. We have $\Pr\left[V_2^* \geq \frac{OPT}{2},\ V_2^* \geq V_1^* \geq \frac{OPT - v(e^*)}{4}\right] \geq \frac{1}{4}$.*

Proof. Applying Lemma 2.7 to the set $S = O^*$, we obtain that $\Pr\left[v(U_1 \cap O^*), v(U_2 \cap O^*) \geq \frac{OPT - v(e^*)}{4}\right] \geq \frac{1}{2}$, which implies that $\Pr\left[V_1^*, V_2^* \geq \frac{OPT - v(e^*)}{4}\right] \geq \frac{1}{2}$. Let Ω be the event that V_1^* and V_2^* are both at least $\frac{OPT - v(e^*)}{4}$. Note that V_1^* and V_2^* are identically distributed, and this remains true even when we condition on the event Ω. It follows that $\Pr[\Omega] \leq \Pr[\{V_1^* \geq V_2^*\} \wedge \Omega] + \Pr[\{V_2^* \geq V_1^*\} \wedge \Omega] = 2 \cdot \Pr[\{V_2^* \geq V_1^*\} \wedge \Omega]$. It follows that $\Pr[\{V_2^* \geq V_1^*\} \wedge \Omega] \geq \frac{1}{4}$. Observe that $V_1^* + V_2^* \geq OPT$. So the event $\{V_2^* \geq V_1^*\}$ also implies that $V_2^* \geq \frac{OPT}{2}$. ◀

For XOS valuations, we prove an analogous result for the optimal value of the LP-relaxation (BudgP) for the algorithmic problem of computing OPT. This will be more useful to us, since (BudgP) can be solved in polytime given a generalized demand oracle. Let LP^* be the optimal value of the following LP:

$$\max \sum_{S \in \mathcal{I}} v(S) x_S \qquad \text{s.t.} \qquad \sum_{S \in \mathcal{I}} c(S) x_S \leq B, \qquad \sum_{S \in \mathcal{I}} x_S \leq 1, \qquad x \geq 0. \tag{BudgP}$$

For $A \subseteq U$, let (BudgP(A)) denote the optimal value of (BudgP) when we are restricted to use only items in A (i.e., $x_S > 0$ only if $S \in \mathcal{I},\ S \subseteq A$). We note that (BudgP($A$)) can be solved in polytime using a generalized demand oracle, since the separation problem for the dual corresponds to a generalized-demand-oracle computation. Throughout, for $i = 1, 2$, let LP_i^* denote the optimal value of (BudgP(U_i)).

▶ **Lemma 2.9.** *Let $v : 2^U \mapsto \mathbb{R}_+$ be XOS. With probability at least $1/4$, we have $\mathsf{LP}_2^* \geq \mathsf{LP}_1^* \geq \left(\mathsf{LP}^* - v(e^*)\right)/4$ and $\mathsf{LP}_2^* \geq \mathsf{LP}^*/2$.*

Proof. We utilize Lemma 2.7 in a clever fashion. Let x^* be an optimal solution to (BudgP). For each $S \subseteq U$, let $q^S \in \mathbb{R}_+^U$ be the additive valuation supporting S. Defining $w_e = \sum_{S \in \mathcal{I} : e \in S} x_S^* q_e^S$ for each $e \in U$, we can write the value of x^* as $w(U)$. Moreover, $w(U_i) \leq \mathsf{LP}_i^*$ for $i = 1, 2$: this is because x^* can be mapped to the solution $\{x_A := \sum_{S \in \mathcal{I} : S \cap U_i = A} x_S^*\}_{A \in \mathcal{I}, A \subseteq U_i}$ that is feasible for (BudgP(U_i)) and whose value is at least $w(U_i)$. Also, $\max_{e \in U} w_e \leq v(e^*)$. So considering the additive valuation given by w, by Lemma 2.7, we have that $\Pr\left[w(U_1), w(U_2) \geq \frac{w(U) - \max_{e \in U} w_e}{4}\right] \geq \frac{1}{2}$. As in Corollary 2.8, this also implies that $\Pr\left[\mathsf{LP}_2^* \geq \mathsf{LP}_1^* \geq \frac{\mathsf{LP}^* - v(e^*)}{4}\right] \geq \frac{1}{4}$. We also have $\mathsf{LP}_1^* + \mathsf{LP}_2^* \geq \mathsf{LP}^*$, so if the event $\{\mathsf{LP}_2^* \geq \mathsf{LP}_1^*\}$ happens, then $\mathsf{LP}_2^* \geq \mathsf{LP}^*/2$. ◀

3 XOS Valuations

We begin by describing the mechanisms utilizing a generalized demand oracle, which leads to part (a) of Theorem 1.1. As mentioned earlier, we describe the underlying algorithm, which we prove is monotone and budget feasible. In Section 3.2, we describe the polytime mechanisms utilizing a knapsack-cover oracle (but not a generalized demand oracle), which will prove parts (b) and (c) of Theorem 1.1.

3.1 Mechanisms using a generalized demand oracle

We deviate slightly from the description in the Introduction in that, to slightly improve the approximation factor, after randomly partitioning the elements and computing an estimate V_1 of OPT from the first part, we find a good candidate set S^* from the second part by running a generalized demand oracle with prices $q_e = \frac{\lambda \cdot V_1 + v(e^*)/2}{B} \cdot c_e$. Recall that $e^* = \operatorname{argmax}_{e \in U} v(e)$, LP^* is the optimal value of (BudgP), and given a partition U_1, U_2 of U, we define LP_i^* as the optimal value of (BudgP(U_i)) for $i = 1, 2$. Clearly, $\mathsf{LP}^* \geq OPT$.

Algorithm XOS-ALG-GD. // algorithm(s) using generalized demand oracle

Input: Budget-feasible MD instance $\big(U, v : 2^U \mapsto \mathbb{R}_+, \mathcal{I}, \{c_e\}, B\big)$; parameters $p, \lambda \in [0, 1]$
Output: subset of U; payments are threshold values

1 Partition U into two sets U_1, U_2, by placing each element of U independently with probability $\frac{1}{2}$ in U_1 or U_2. Compute LP_1^*, the LP-optimum achievable from U_1, using a generalized demand oracle. Set $V_1 = \mathsf{LP}_1^*$.
2 Use a generalized demand oracle to obtain $S^* \leftarrow \operatorname{argmax} \big\{ v(S) - \frac{\lambda V_1 + v(e^*)/2}{B} \cdot c(S) : \ S \in \mathcal{I}, \ S \subseteq U_2 \big\}$.
3 Considering elements of S^* in increasing order, return a maximal prefix T of S^* with $v(T) \leq \lambda V_1 + v(e^*)/2$.
4 **return** T with probability p, and e^* with probability $1 - p$.

▶ **Theorem 3.1** (Part (a) of Theorem 1.1). *Taking $\lambda = \frac{2}{3}$ and $p = \frac{6}{7}$ in Algorithm XOS-Alg-GD, we obtain a polytime randomized 28-approximation budget-feasible mechanism.*

Proof. Lemmas 3.2 and 3.3 prove that truthfulness, individual rationality, and budget feasibility hold with probability 1, and that payments can be computed in polytime. We focus here on proving the approximation guarantee. Note that the set output is in $\mathcal{I}$ since $S^* \in \mathcal{I}$ and $\mathcal{I}$ is a downwards-monotone family. By Lemma 2.9, we have $\Pr\big[\mathsf{LP}_2^* \geq \frac{\mathsf{LP}^*}{2}, \ \mathsf{LP}_2^* \geq \mathsf{LP}_1^* \geq \frac{\mathsf{LP}^* - v(e^*)}{4}\big] \geq \frac{1}{4}$. Assume that this event happens. We claim that $v(S^*) \geq \mathsf{LP}_2^* - \lambda V_1 - 0.5 v(e^*)$, and since $V_1 = \mathsf{LP}_1^* \leq \mathsf{LP}_2^*$, this yields $v(S^*) \geq \frac{1-\lambda}{2} \cdot \mathsf{LP}^* - v(e^*)/2$. To see the claim, let $\theta = \lambda V_1 + v(e^*)/2$ and let x^* be an optimal solution to (BudgP(U_2)). Then $v(S^*) - \frac{\theta}{B} \cdot c(S^*) \geq \sum_{S \in \mathcal{I}: S \subseteq U_2} x_S^* \Big(v(S) - \frac{\theta}{B} \cdot c(S) \Big) \geq \mathsf{LP}_2^* - \theta$; the first inequality is because $\sum_{S \in \mathcal{I}, S \subseteq U_2} x_S^* \leq 1$, and the second is because $\sum_{S \in \mathcal{I}, S \subseteq U_2} c(S) x_S^* \leq B$.

The set T obtained in step 3 is either S^*, and if not, has value at least $\lambda V_1 - v(e^*)/2$. It follows that

$$v(T) \geq \min\Big\{ \tfrac{1-\lambda}{2} \cdot \mathsf{LP}^*, \tfrac{\lambda(\mathsf{LP}^* - v(e^*))}{4} \Big\} - \tfrac{v(e^*)}{2} = \min\Big\{ \tfrac{\mathsf{LP}^*}{6}, \tfrac{\mathsf{LP}^* - v(e^*)}{6} \Big\} - \tfrac{v(e^*)}{2}, = \tfrac{\mathsf{LP}^*}{6} - \tfrac{2}{3} \cdot v(e^*).$$

Hence, the expected value returned is $(1-p) v(e^*) + \frac{p}{4} \cdot v(T) \geq \frac{1}{7} \cdot v(e^*) + \frac{1}{28} \cdot \mathsf{LP}^* - \frac{1}{7} \cdot v(e^*) = \frac{\mathsf{LP}^*}{28}$. ◀

▶ **Lemma 3.2.** *Algorithm XOS-Alg-GD is a distribution over monotone, nobossy algorithms. For each monotone algorithm in the support, the threshold values can be computed efficiently.*

Proof. We consider each possible outcome of the random choices made in Algorithm XOS-Alg-GD, and show that the resulting algorithm is monotone. If the algorithm returns e^*, then it is trivially monotone. Also, the threshold value here is B, due to our initial preprocessing.

Now consider the outcome where we obtain partition U_1, U_2 by our random partitioning in step 1 and return the set T in step 4. Suppose that $S^* \subseteq U_2$ is the set computed in step 2 under input c. Consider any $e \in S^*$. We argue that for any $c'_e < c_e$, we still obtain the set

S^* in step 2 under the input $c' = (c'_e, c_{-e})$. Since the mapping from S^* to the final set T that is returned is independent of the costs, it follows that the same set T is returned under the inputs c and c'. This establishes monotonicity *and nobossiness*: the set returned does not change if a winner decreases her cost.

For $d \in \mathbb{R}^U_+$ and $S \subseteq U_2$, let $\mathrm{demd}(S; d) := v(S) - \frac{\lambda V_1 + v(e^*)/2}{B} \cdot d(S)$. We claim that the set of optimal solutions to $\max\{\mathrm{demd}(S; c') : S \in \mathcal{I},\ S \subseteq U_2\}$ contains S^*, and is a subset of the collection of optimal solutions to $\max\{\mathrm{demd}(S; c) : S \in \mathcal{I},\ S \subseteq U_2\}$. Given this, since the output S^* of the generalized demand oracle on input c is the lexicographically-smallest set among all optimal solutions to $\max\{\mathrm{demd}(S; c) : S \in \mathcal{I},\ S \subseteq U_2\}$, it follows that S^* is also the lexicographically-smallest set among all optimal solutions to $\max\{\mathrm{demd}(S; c') : S \in \mathcal{I},\ S \subseteq U_2\}$. To prove the claim, for any $A \subseteq U_2$, we have $\mathrm{demd}(A; c') - \mathrm{demd}(A, c) = \frac{\lambda V_1 + v(e^*)/2}{B} \cdot (c_e - c'_e)$ if $e \in A$, and equal to 0 otherwise. This implies that every optimal solution A to $\max\{\mathrm{demd}(S; c') : S \in \mathcal{I},\ S \subseteq U_2\}$ must satisfy: (i) $\mathrm{demd}(A; c') \geq \mathrm{demd}(S^*, c') > \mathrm{demd}(S^*, c) \geq \mathrm{demd}(A; c)$, and hence $e \in A$; and (ii) consequently, $0 \leq \mathrm{demd}(A; c') - \mathrm{demd}(S^*; c') = \mathrm{demd}(A; c) - \mathrm{demd}(S^*; c) \leq 0$, thus we must have equality everywhere. Due to (ii) S^* is an optimal solution to $\max\{\mathrm{demd}(S; c') : S \in \mathcal{I},\ S \subseteq U_2\}$, and A is an optimal solution to $\max\{\mathrm{demd}(S; c) : S \in \mathcal{I},\ S \subseteq U_2\}$.

Computation of threshold values. Consider some input c. We may again focus on the outcome where we do not return e^*. Suppose that U_1, U_2 is the partition obtained in step 1, and $T \subseteq S^* \subseteq U_2$ are the sets computed in steps 3 and 2. By the nobossiness property established earlier, in order for some $e \in T$ to be a winner under input $c' = (c'_e, c_{-e})$, it must be that, under input c', the same sets S^* and T are returned in steps 2, 3. Therefore, the threshold value for e is simply the largest value τ_e under which S^* is an optimal solution to $\max\{\mathrm{demd}\big(S; (\tau_e, c_{-e})\big) : S \in \mathcal{I},\ S \subseteq U_2\}$. It is not hard to see that this can be calculated in polytime using a generalized demand oracle. In particular, a generalized demand oracle is an affine minimizer, and this threshold is simply the payment in the VCG mechanism. ◀

▶ **Lemma 3.3.** *Algorithm XOS-Alg-GD is a distribution over budget-feasible algorithms.*

Proof. If the outcome of the random choices is to return e^*, then the total payment is B. So suppose otherwise. Let U_1, U_2 be the partition obtained in step 1, and $T \subseteq S^* \subseteq U_2$ be the sets computed in steps 2, 3. From the threshold-value computation in the proof of Lemma 3.2, we can infer that the threshold value of player $e \in T$ is at most $\frac{B}{\lambda V_1 + v(e^*)/2} \cdot \big(v(S^*) - v(S^* - e)\big)$, since for any larger value τ, we would have $\mathrm{demd}\big(S^* - e, (\tau, c_{-e})\big) > \mathrm{demd}\big(S^*, (\tau, c_{-e})\big)$. Therefore, the total payment is at most

$$\frac{B}{\lambda V_1 + v(e^*)/2} \cdot \sum_{e \in T} \Big(v(S^*) - v(S^* - e)\Big) \leq \frac{B}{\lambda V_1 + v(e^*)/2} \cdot v(T) \leq B.$$

The first inequality is due to Claim 2.5; the second is because $v(T) \leq \lambda V_1 + v(e^*)/2$. ◀

3.2 Mechanisms using a knapsack-cover oracle

We now describe the polytime mechanisms leading to parts (b) and (c) of Theorem 1.1. These mechanisms utilize a knapsack-cover oracle, but do not require a generalized demand oracle. The basic idea is similar to that of Algorithm XOS-Alg-GD, but one key difference is that after we randomly partition U, we obtain an estimate V_1 of V_1^*, the integer optimum achievable from U_1, and run a knapsack-cover oracle on the second part to obtain a good candidate set S^*. The postprocessing of S^* is also different: earlier, we could directly infer a

threshold value for an agent e to be included in S^* and thereby argue budget feasibility; now, we prune S^* to explicitly enforce this, so that, as before, we can ensure budget feasibility by returning a suitable prefix of the pruned set.

The mechanisms we present here demonstrate the effectiveness of knapsack-cover oracles, and are interesting for various reasons. First, they show that budget-feasibility can be enforced given an oracle for the underlying algorithmic problem. Moreover, these mechanisms work with even an *approximate* knapsack-cover oracle satisfying monotonicity and nobossiness (and in fact, even with a weaker feasibility oracle); see Remark 3.7. Thus, Algorithm XOS-Alg-KC yields a reduction from budget-feasible mechanism design to the design of truthful, nobossy, approximation mechanisms for generalized knapsack-cover. Second, the friendliness of Algorithm XOS-Alg-KC to approximate knapsack-cover oracles makes it more amenable than Algorithm XOS-Alg-GD for yielding efficient mechanisms, when the valuation function is specified implicitly (e.g., via a combinatorial-optimization problem) or in terms of a weaker oracle. The analysis of budget feasibility in Section 3.1 relied crucially on having an *exact* generalized demand oracle: specifically, the upper bound obtained on the payment made to a winner e relies on this exactness; the bound obtained from an approximate generalized demand oracle is not strong enough for the arguments to go through (unless the approximation factor is quite small). Often, computing even a good approximation for a mixed-sign objective problem, as in the case of generalized-demand oracle, is quite difficult. Algorithm XOS-Alg-KC shows that this difficulty does not pose an impediment provided we have a (approximate, nobossy) knapsack-cover (feasibility) oracle.

Algorithm XOS-ALG-KC. // algorithm(s) using knapsack-cover oracle

Input: Budget-feasible MD instance $\big(U, v : 2^U \mapsto \mathbb{R}_+, \mathcal{I}, \{c_e\}, B\big)$; parameters $p, \lambda \in [0,1]$
Output: subset of U; payments are threshold values

1 Partition U into two sets U_1, U_2, by placing each element of U independently with probability $\frac{1}{2}$ in U_1 or U_2. Compute estimate V_1 of the integer optimum achievable from U_1, using a knapsack-cover oracle.
2 Use a knapsack-cover oracle to obtain $S^* \leftarrow \operatorname{argmin}\big\{c(S) :\ v(S) \geq V_1,\ S \in \mathcal{I},\ S \subseteq U_2\big\}$. If the problem is infeasible, **return** $\emptyset$.
3 Let $A \leftarrow S^*$. While there exists $e \in A$ such that $c_e > \frac{B}{\lambda \cdot V_1} \cdot \big(v(A) - v(A-e)\big)$, considering elements in increasing order, find the first such element $f \in A$ and update $A \leftarrow A - f$.
4 Considering elements of A in increasing order, return a maximal prefix T of A with $v(T) \leq \lambda V_1$.
5 **return** T with probability p, and e^* with probability $1-p$.

▶ **Theorem 3.4** (Part (b) of Theorem 1.1). *For any $\epsilon > 0$, taking $\lambda = 0.5$ and $p = \frac{32+0.8\epsilon}{41+\epsilon}$ in Algorithm XOS-Alg-KC, we obtain a polytime randomized $(41+\epsilon)$-approximation budget-feasible mechanism.*

Proof. Lemmas 3.5 and 3.6 prove that truthfulness, individual rationality, and budget feasibility hold with probability 1, and that payments can be computed in polytime. Let $\beta = 1+\epsilon/40$. So $p = \frac{32\beta}{40\beta+1}$. Recall that $V_i^* := \max\{v(S) :\ S \in \mathcal{I},\ S \subseteq U_i,\ c(S) \leq B\}$, for $i = 1, 2$. We can assume that $V_1 \geq V_1^*/\beta$. By Lemma 2.8, we have that $\Pr\big[V_2^* \geq V_1^* \geq \frac{OPT - v(e^*)}{4}\big] \geq \frac{1}{4}$. Assume that this event happens. Then the optimization problem in step 2 is feasible, and we have $c(S^*) \leq B$. Let $A_0 = S^*, A_1, \ldots, A_k$ be the sequence sets obtained in step 3, where $A_i = A_{i-1} - f_i$ for all $i = 1, \ldots, k$. Then we have $v(A_{i-1}) - v(A_i) < \frac{\lambda V_1}{B} \cdot c_{f_i}$ for all $i = 1, \ldots, k$.

Adding these inequalities gives $v(S^*) - v(A_k) \leq \frac{\lambda V_1}{B} \cdot c(\{f_1, \ldots, f_k\}) \leq \frac{\lambda V_1}{B} \cdot c(S^*) \leq \lambda V_1$. Thus, the set A obtained at the end of step 3 satisfies $v(A) \geq (1-\lambda)V_1$. The set T obtained in step 3 is either A, and if not, has value at least $\lambda V_1 - v(e^*)$. It follows that

$$v(T) \geq \min\big\{(1-\lambda)V_1, \lambda V_1 - v(e^*)\big\} = 0.5 \cdot V_1 - v(e^*) \geq \tfrac{1}{8\beta} \cdot OPT - \tfrac{8\beta+1}{8\beta} \cdot v(e^*).$$

Hence, the expected value returned is

$$(1-p)v(e^*) + \frac{p}{4} \cdot v(T) \geq \frac{8\beta+1}{40\beta+1} \cdot v(e^*) + \frac{8\beta}{40\beta+1} \cdot \Big(\tfrac{1}{8\beta} \cdot OPT - \tfrac{8\beta+1}{8\beta} \cdot v(e^*)\Big) = \frac{OPT}{40\beta+1}. \quad \blacktriangleleft$$

▶ **Lemma 3.5.** *Algorithm XOS-Alg-KC is a distribution over monotone, nobossy algorithms. for each monotone algorithm in the support, the threshold values can be computed efficiently.*

Proof. As before, we consider each possible outcome of the random choices made in Algorithm XOS-Alg-KC and show that the resulting algorithm is monotone. If it returns e^*, then it is trivially monotone. Also, the threshold value here is B, due to our initial preprocessing.

Now suppose we obtain partition U_1, U_2 by our random partitioning in step 1 and return the set T in step 5. Suppose that $S^* \subseteq U_2$ is the set computed in step 2 under input c. Consider any $e \in S^*$. We argue that for any $c'_e < c_e$, we still obtain the set S^* in step 2 under the input (c'_e, c_{-e}). It follows then that for the set A obtained at the end of step 3, for any $e \in A$ and any $c'_e < c_e$, we still obtain the same set (S^* and) A at the end of step 3 under input (c'_e, c_{-e}). Since the mapping from A to the final set T that is returned is independent of the costs, it follows that the same set is returned under the inputs c and c'. This yields monotonicity and nobossiness.

We now prove the claim. So consider $e \in S^*$, and $c'_e < c_e$. Let $c' = (c'_e, c_{-e})$. We argue that the set of optimal solutions to $\min\{c'(S) : v(S) \geq Val,\ S \in \mathcal{I},\ S \subseteq U_2\}$ is precisely the set of optimal solutions to $\min\{c(S) : v(S) \geq Val,\ S \in \mathcal{I},\ S \subseteq U_2\}$ that contain e. The optimal value under c' is at least (optimal value under c) $-(c_e - c'_e)$. Thus, any optimal solution to the problem with costs c containing e (in particular, S^*) must be an optimal solution to $\min\{c'(S) : v(S) \geq Val,\ S \in \mathcal{I},\ S \subseteq U_2\}$. Now suppose A is an optimal solution to $\min\{c'(S) : v(S) \geq Val,\ S \in \mathcal{I},\ S \subseteq U_2\}$. Then $c'(A) = c'(S^*) < c(S^*) \leq c(A)$, so $e \in A$, and then $0 = c'(A) - c'(S^*) = c(A) - c(S^*)$, so A is an optimal solution to the problem with costs c. Since the knapsack-cover oracle always returns the lexicographically-smallest set among all optimal solutions, it follows that S^* is the lexicographically-smallest set also among all optimal solutions to $\min\{c'(S) : v(S) \geq Val,\ S \in \mathcal{I},\ S \subseteq U_2\}$, and hence will be returned by the knapsack-cover oracle under input c'.

Computation of threshold values. Consider some input c. We may again focus on the outcome where we do not return e^*. Suppose that U_1, U_2 is the partition obtained in step 1, and $T \subseteq A \subseteq S^* \subseteq U_2$ are the sets computed in steps 4, 3 (at the end of the step), 2 respectively. By the nobossiness property established earlier, in order for some $e \in T$ to be a winner under input $c' = (c'_e, c_{-e})$, it must be that, under input c', we obtain the same set S^* in step 2, the same sequence of sets in step 3, and the same set T in step 4. For $e \in T$, the largest value c_e for which we obtain S^* in step 2 can be computed efficiently using a knapsack-cover oracle; thus, the threshold value for $e \in T$ can be computed efficiently. Due to our rule for dropping elements in step 3, this threshold is at most $\frac{B}{\lambda V_1} \cdot \big(v(A) - v(A-e)\big)$. $\blacktriangleleft$

▶ **Lemma 3.6.** *Algorithm XOS-Alg-KC is a distribution over budget-feasible algorithms.*

Proof. If the outcome of the random choices is to return e^*, then the total payment is B. So suppose otherwise. Suppose that U_1, U_2 is the partition obtained in step 1, and $T \subseteq A \subseteq S^* \subseteq U_2$ are the sets computed in steps 4, 3 and 2 respectively. By the argument in

the proof of Lemma 3.5, due to our rule for dropping elements in step 3, the threshold value of a winner $e \in T$ is at most $\frac{B}{\lambda V_1} \cdot \big(v(A) - v(A-e)\big)$. So the sum of threshold values of elements in T is at most $\frac{B}{\lambda V_1} \cdot v(T)$, due to Claim 2.5, and hence, at most B since $v(T) \leq \lambda V_1$. ◀

Part (c) of Theorem 1.1. If, in addition to a knapsack-cover oracle, we are given an XOS oracle, then we can obtain an improved $(33+\epsilon)$-approximation as follows. An XOS oracle for v receives a set $S \subseteq U$ as input and returns an additive valuation supporting S. Instead of steps 3 and 4 in Algorithm XOS-Alg-KC, we use an XOS oracle for v to obtain $q \in \mathbb{R}^U_+$ supporting S^* and then run the randomized 2-approximation budget-feasible mechanism for additive valuations due to [18] on this additive valuation to obtain the set T. (As before, we return T with some probability p and e^* with probability $1-p$, where $1-p$ is now $\frac{1}{33+\epsilon}$.)

▶ Remark 3.7. We do not actually need an oracle that solves the generalized knapsack-cover problem *exactly*. Besides the fact that the oracle returns a low-cost set, what was crucial is that the oracle satisfies monotonicity and nobossiness. We can in fact work with any algorithm Alg for generalized knapsack-cover satisfying these properties. More precisely, suppose Alg satisfies the following properties.

- **(i)** **(α, β)-approximate knapsack-cover oracle**: for some $\alpha, \beta \geq 1$, on any input $q \in \mathbb{R}^U_+$, Val, it returns $S \in \mathcal{I}$ such that $q(S) \leq \alpha \cdot \big(\min\{q(S) : v(S) \geq Val,\ S \in \mathcal{I}\}\big)$ and $v(S) \geq Val/\beta$ if the problem is feasible;
- **(ii)** **Monotonicity and nobossiness**: if e belongs to the output set S for some $q \in \mathbb{R}^U_+$, then the same set S is output for any $q' = (q'_e, q_{-e})$ with $q'_e \leq q_e$.

Then, we can utilize algorithm Alg in place of a knapsack-cover oracle in step 2 of Algorithm XOS-Alg-KC, and choose parameters p, λ suitably, to obtain an $O(\alpha\beta)$-approximation budget-feasible mechanism. Thus, our construction can be seen as a reduction from budget-feasible mechanism design to the design of truthful, nobossy approximation mechanisms for the generalized knapsack-cover problem. In fact, since we only seek some S^* with $v(S^*) \geq Val$, $c(S^*) \leq B$, we can work with an even weaker *feasibility oracle* satisfying monotonicity and nobossiness. A feasibility oracle is also given a budget B and returns a set in $\{S \in \mathcal{I} : q(S) \leq B,\ v(S) \geq Val\}$ or declares infeasibility; again, we can work with a monotone, nobossy, approximate feasibility oracle, which returns S with $q(S) \leq \alpha B$, $v(S) \geq Val/\beta$ if the problem is feasible.

4 Subadditive valuations

In this section, we develop various budget-feasible mechanisms for subadditive valuations. We begin by presenting a 33-approximation budget-feasible mechanism for subadditive valuations, proving Theorem 1.2. We obtain this by distilling and refining a key insight from the existential $O(1)$-approximation for subadditive valuations in [10], which we state below.

▶ **Lemma 4.1.** *Let $v : 2^U \mapsto \mathbb{R}_+$ be a subadditive function, $S \subseteq U$, and $K \subseteq \mathbb{R}^S_+$ be a finite set. There exists a distribution $\mathcal{D}$ over K such that, for any $c \in K$, we have $\mathbb{E}_{\tau \sim \mathcal{D}}\big[v(\{e \in S : c_e \leq \tau_e\})\big] \geq 0.5 \cdot v(S)$.*

We show that, applied to a suitable set K, the distribution given by Lemma 4.1 immediately yields an (explicit) $O(1)$-approximation budget-feasible mechanism. For the set K that arises in the application to budget-feasible mechanism design, we do not know how to obtain such a distribution in polytime, and we leave this as an enticing open question. However, such a distribution can be computed in exponential time, thereby yielding an explicit exponential-time 33-approximation budget-feasible mechanism.

In the full version, we also present a polytime mechanism obtained via the same template that yields an $O(1)$-approximation for any *cardinality-based* subadditive valuation v, i.e., where $v(S)$ depends only on $|S|$ for every $S \subseteq U$. To our knowledge, this is the first such guarantee for such subadditive functions. Singer [24] obtained a 2-approximation for cardinality-based submodular functions (called symmetric submodular functions in [24]), and showed that this is best possible in this setting. This can be seen as providing some evidence that the above distribution-based template holds promise and may lead to stronger guarantees for subadditive valuations.

As we show in Section 5, Lemma 4.1 is quite powerful and can be used to decouple the truthfulness + approximation and budget-feasibility requirements, and thus obtain a reduction from budget-feasible mechanism design to the task of designing a truthful, nobossy mechanism that well-approximates OPT (see Theorem 5.2).

Mechanism yielding Theorem 1.2. By scaling, we may assume that the input cost vector is integral. Our mechanism is quite simple. Recall that given a partition U_1, U_2 of U, we define $V_i^* := \max\{v(S) : \ S \in \mathcal{I},\ S \subseteq U_i,\ c(S) \leq B\}$, for $i = 1, 2$.

Algorithm SUBADD-ALG. // $O(1)$-approximation for subadditive valuations

Input: Budget-feasible MD instance $\big(U, v : 2^U \mapsto \mathbb{R}_+, \mathcal{I}, \{c_e\}, B\big)$, where c is an integral vector

Output: subset of U; payments are threshold values

1 As in Algorithm XOS-Alg-KC, obtain partition U_1, U_2, and compute $V_1 = V_1^*$ and $S^* \leftarrow \operatorname{argmin}\big\{c(S) : \ v(S) \geq V_1,\ S \in \mathcal{I},\ S \subseteq U_2\big\}$. If this problem is infeasible or $c(S^*) > B$, **return** $\emptyset$.

2 Let $K = K_{S^*} := \big\{x \in \mathbb{Z}_+^{S^*} : \sum_{e \in S^*} x_e \leq B\big\}$. Apply Lemma 4.1 to obtain a distribution $\mathcal{D}$ over K. Sample a random threshold vector $\tau \in K$ from $\mathcal{D}$, and let $T \leftarrow \{e \in S^* : c_e \leq \tau_e\}$.

3 **return** T with probability $p := \frac{32}{33}$, and e^* with probability $1 - p$.

Proof of Theorem 1.2. We defer the proof of Lemma 4.1, and show here that the above mechanism has the stated guarantees assuming this. We first argue that Algorithm Subadd-Alg is a distribution over monotone, budget-feasible algorithms. If the algorithm returns e, it is trivially monotone and budget feasible. So suppose otherwise. Once we fix the random choices determining the partition U_1, U_2, we can view the resulting algorithm as follows. For each $S \subseteq U_2$, we obtain the distribution given by Lemma 4.1 for the set K_S. Fixing the random bits then fixes a specific threshold vector in K_S sampled from this distribution, for each $S \subseteq U_2$. Thus, we obtain a distribution over deterministic algorithms.

Suppose S^* was computed in step 1 of Algorithm Subadd-Alg for input c. We know that for $e \in S^*$ and $c'_e < c_e$, under input (c'_e, c_{-e}), we again compute S^*. Since the threshold vector $\tau \in K_{S^*}$ is fixed by the random choices, we therefore return the same set T under input c and (c'_e, c_{-e}). This proves monotonicity and nobossiness. Moreover, the threshold value for a winner $e \in T$ is the minimum of the threshold for e to belong to S^*, and τ_e. Since $\sum_{e \in S^*} \tau_e \leq B$, we obtain budget feasibility.

We now analyze the approximation guarantee. We have $\Pr\big[V_2^* \geq V_1^* \geq \frac{OPT - v(e^*)}{4}\big] \geq \frac{1}{4}$. Assume this event happens. Then, $c(S^*) \leq B$, and so by Lemma 4.1, $\mathbb{E}\big[v(T)\big] \geq v(S^*)/2 \geq V_1/2$. So the expected value returned is at least $(1-p)v(e^*) + \frac{p}{4} \cdot \frac{OPT - v(e^*)}{8} = OPT/33$.

The running time is dominated by the time needed to compute $V_1 = V_1^*$, and the distribution given by Lemma 4.1. We can clearly compute V_1^* in $O\big(2^n\big)$ time. (The computation of

V_1 is however not really a bottleneck, since we can instead work with a polytime-computable estimate $V_1 \geq V_1^*/\beta$.) We can solve the LP (P) in the proof of Lemma 4.1 in $\mathsf{poly}(|K_{S^*}|)$ time, and since $|K_{S^*}| = \exp\big(O(\max\{n, B\})\big)$ this yields $\exp\big(O(\max\{n, B\})\big)$ running time. ◀

Proof of Lemma 4.1. Consider the following primal LP (P) for finding the desired distribution, with variables x_τ for every point $\tau \in K$ and $\lambda \in \mathbb{R}$, and its dual (D). This primal-dual pair can be viewed as encoding the problems of finding the optimal mixed strategies for two players in a two-player zero-sum game.

$$\max \quad \lambda \quad \text{s.t.} \quad \sum_{\tau \in K} x_\tau \cdot \tfrac{v(\{e \in S : c_e \leq \tau_e\})}{v(S)} \geq \lambda \quad \forall c \in K, \qquad \sum_{\tau \in K} x_\tau = 1, \qquad x \geq 0. \tag{P}$$

$$\min \quad \mu \quad \text{s.t.} \quad \sum_{c \in K} y_c \cdot \tfrac{v(\{e \in S : c_e \leq \tau_e\})}{v(S)} \leq \mu \quad \forall \tau \in K, \qquad \sum_{c \in K} y_c = 1, \qquad y \geq 0. \tag{D}$$

Let (μ, y) be a feasible solution to (D). We argue that $\mu \geq 0.5$. By duality, this implies that the optimal value of (P) is at least 0.5. The optimal solution to (P) yields the desired distribution.

Let $M \in \mathbb{R}_+^{K \times K}$ be the matrix with entries $M_{c,\tau} = \frac{v(\{e \in S : c_e \leq \tau_e\})}{v(S)}$ for all $c, \tau \in K$. Let $J \in \mathbb{R}_+^{K \times K}$ be the all 1s matrix, and $\vec{1}$ be the all 1s vector in $\mathbb{R}^K$. Then, the constraints of (D) can be written compactly as $M^T y \leq \mu \vec{1}$, $y^T \vec{1} = 1$. So we have $\mu \geq y^T M^T y = y^T M y = 0.5 \cdot y^T (M + M^T) y$. Note that $M + M^T \geq J$ as v is subadditive and therefore $y^T (M + M^T) y \geq y^T J y = 1$. ◀

Improving the running time. We can improve the running time to $\exp\big(O(n)\big)$ while worsening the approximation by a factor of roughly 2, by suitably sparsifying K_{S^*}. After obtaining S^*, we simply scale and round $c|_{S^*} = (c_e)_{e \in S^*}$ so that its entries lie in $[n]$ and they sum to $2n$. This reduces the space of vectors to $2^{O(n)}$ yielding the improved running time for solving (P), and hence the algorithm. We need to take some care to ensure budget feasibility as the rounding introduces some error, leading to a violation of the budget. To compensate for this, we reduce the budget by factor of 2, which translates to an $O(1)$-factor loss in the approximation. We thus obtain a $2^{O(n)}$-time 69-approximation budget-feasible mechanism.

5 Budget-feasible mechanism design with general payment constraints

We now consider *generalized budget-feasible mechanism design*, wherein we have constraints on the payments doled out by the mechanism that are more general than just a bound on the total payment, as is the case in standard budget-feasible mechanism design. We focus on two types of payment constraints: (1) *group budget constraints*, wherein players may be divided into (potentially overlapping) groups, and we have a budget constraint on the total payment made to the winners in each group; (2) Top-ℓ *budget constraints*, wherein we have budget constraints on the sum of the ℓ largest payments doled out by the mechanism, for one or more indices $\ell \in [n]$.

5.1 Group budget constraints

We first define the problem precisely: we are given k subsets $G_1, \ldots, G_k$ of players, along with budgets $B_1, \ldots, B_k$. As before, each player $e \in U$ has a private cost $c_e \geq 0$, we have a valuation function $v : 2^U \mapsto \mathbb{R}_+$, and a downwards-monotone family $\mathcal{I} \subseteq 2^U$. The algorithmic problem is to compute (a good approximation to) $OPT := \max\{v(S) : S \in \mathcal{I},\ c(S \cap G_i) \leq B_i\ \forall i \in [k]\}$; we call a set $S \in \mathcal{I}$ with $c(S \cap G_i) \leq B_i$ for all $i \in [k]$, a feasible set. The mechanism-design problem is to devise a *group-budget-feasible mechanism*: a truthful, individually-rational

mechanism that returns a set $T \in \mathcal{I}$ such that $v(T)$ is a good approximation to OPT, and where the payments $\{p_e\}$ made to the players satisfy the group budget constraints $p(T \cap G_i) \leq B_i$ for all $i = 1, \ldots, k$. We may assume that $c_e \leq \min_{i \in [k]: e \in G_i} B_i$ (and $\{e\} \in \mathcal{I}$) for all $e \in U$, as we can discard any element e not satisfying this.

Our main result here is an $O(k)$-approximation mechanism (Theorem 5.1) for subadditive valuations, and such a polytime mechanism for XOS valuations, generalizing the state-of-the-art for budget-feasible mechanisms, which corresponds to the case $k = 1$. Moreover, we obtain this via a simple adaptation of our mechanisms for subadditive and XOS valuations, illustrating the versatility of our underlying ideas. We also obtain a novel and powerful reduction demonstrating that the requirements of truthfulness and group-budget-feasibility can be *completely decoupled*: we show that *a truthful, nobossy mechanism that returns an α-approximation to OPT can be transformed in a black-box way to an $O(\alpha)$-approximation group-budget-feasible mechanism* (Theorem 5.2). For XOS valuations, we also obtain a *polytime* reduction that yields an $O(k\alpha)$-approximation group-budget-feasible mechanism. These reductions are of independent interest, and we believe will prove to be useful in (generalized) budget-feasible mechanism design, since they allow one to ignore payment constraints and focus on only the truthfulness component (albeit with the additional nobossiness requirement), which is usually an easier task, especially in single-dimensional domains.

▶ **Theorem 5.1** (Part (a) of Theorem 1.3)**.** *For any downwards-monotone family and any subadditive valuation, we obtain a randomized $O(k)$-approximation group-budget-feasible mechanism. For XOS valuations, we obtain a polytime mechanism given a generalized demand oracle.*

Proof sketch. The mechanisms for both types of valuations follow the same template. We obtain a random partition U_1, U_2, and use U_1 to obtain an estimate V_1 of OPT, or the LP-optimum. We compute $S^* \leftarrow \operatorname{argmax}\left\{v(S) - \frac{V_1}{2k} \cdot \sum_{i=1}^{k} \frac{c(S \cap G_i)}{B_i} : S \in \mathcal{I},\ S \subseteq U_2\right\}$ via a generalized demand oracle. For subadditive valuations (where we are ignoring computational concerns), we take V_1 to be the integer optimum for U_1, and for XOS valuations, we take V_1 to be the LP-optimum for U_1, which can be computed in polytime given a generalized demand oracle. We take R to be a maximal prefix of S^* with $v(R) \leq V_1/2k$.

One can then argue that $c|_R$ satisfies the group budget constraints, and for XOS valuations, the threshold values for $e \in R$ (i.e., the payments) satisfy the group budget constraints. Moreover, assuming that V_1 is a good estimate of OPT, we can argue that $v(R) = \Omega(OPT/k)$. This finishes things up for XOS valuations. For subadditive valuations, we can now utilize Theorem 5.2 (a) to return a suitable subset of R, since R is computed via a monotone, nobossy algorithm. ◀

▶ **Theorem 5.2.** *Let $\mathcal{M} = (f, p)$ be a (possibly randomized) truthful, nobossy mechanism for a class $\mathcal{C}$ of subadditive valuations (and downwards-monotone family $\mathcal{I}$), achieving an α-approximation to OPT. (a) We can utilize $\mathcal{M}$ to obtain a 2α-approximation group-budget-feasible mechanism for class $\mathcal{C}$. (b) If $\mathcal{C}$ consists of XOS valuations and M runs in polytime, then we can obtain a polytime $O(k\alpha)$-approximation group-budget-feasible mechanisms for class $\mathcal{C}$.*

5.2 Top-ℓ budget constraints

The Top-ℓ norm of a vector $x \geq 0$ is the sum of the ℓ largest entries of x. Given $w \in \mathbb{R}^U$ and $S \subseteq U$, define $w|_S := (w_e)_{e \in S}$. The underlying algorithm problem that we now consider is to (approximately) compute $OPT := \max\left\{v(S) : S \in \mathcal{I},\ \text{Top-}\ell(c|_S) \leq B_\ell \quad \forall \ell \in F\right\}$. The index-set F and budgets $\{B_\ell\}_{\ell \in F}$ are part of the input (along with $v, c, \mathcal{I}$). In the

mechanism-design setting, the c_es are private costs, and we seek to devise a truthful, individually rational mechanism that achieves a good approximation to OPT, and where the payment vector $p = (p_e)$ satisfies the *Top-ℓ-budget constraints* Top-$\ell(p) \leq B_\ell$ for all $\ell \in F$. (Since we only make payments to winners, if T is the set returned by the mechanism, we have Top-$\ell(p) =$ Top-$\ell(p|_T)$.)

For $\ell, \ell' \in F$ with $\ell < \ell'$, we may assume that: (i) $B_\ell \leq B_{\ell'}$, since Top-$\ell(x) \leq$ Top-$\ell'(x)$ for any vector x; and (ii) $\frac{B_\ell}{\ell} \geq \frac{B_{\ell'}}{\ell'}$, since for any vector $x \geq 0$, we have Top-$\ell'(x) \leq$ Top-$\ell(x) + (\ell' - \ell) \cdot \frac{\text{Top-}\ell(x)}{\ell}$ as the $(\ell+1)$-th largest entry of x is at most $\frac{\text{Top-}\ell(x)}{\ell}$. (We can always satisfy (i) and (ii) by equivalently (re)defining the budget for index $\ell \in F$ to be $\ell \cdot \min\left\{\frac{B_{\ell''}}{\ell''} : \ell', \ell'' \in F,\ \ell' \leq \min\{\ell, \ell''\}\right\}$.)

Let $k = |F|$. Our main result here is a reduction to (standard) budget-feasible mechanism design while incurring a factor-$(k+1)$ loss in approximation, which applies to any class of subadditive valuations. Thus, we obtain $O(k)$-approximation for subadditive valuations, and polytime mechanisms with this guarantee for XOS valuations and cardinality-based subadditive valuations. Also, observe that the reduction in Theorem 5.2 (a) applies here as well, since we can specify a Top-ℓ-budget constraint Top-$\ell(c|_S) \leq B_\ell$ equivalently via the group-budget constraints $c(S \cap A) \leq B_\ell$ for all $A \subseteq U$ with $|A| = \ell$.

▶ **Theorem 5.3** (Part (b) of Theorem 1.3). *Let $\mathcal{M} = (f, p)$ be a (possibly randomized) α-approximation budget-feasible mechanism for a class $\mathcal{C}$ of subadditive valuations and downwards-monotone family $\mathcal{I}$. We can obtain a randomized $(k+1)\alpha$-approximation truthful mechanism for generalized budget-feasible mechanism design with k Top-ℓ-budget constraints, valuation class $\mathcal{C}$ and downwards-monotone family $\mathcal{I}$, which makes $k+1$ calls to $\mathcal{M}$.*

Proof sketch. We illustrate the main underlying idea. From prior algorithmic work on Top-ℓ-norm and minimum-norm optimization [13, 14], we infer that we can ensure that Top-$\ell(x) \leq B_\ell$ holds with a factor-2 violation by ensuring that $\sum_{i:x_i > B_\ell/\ell} x_i \leq B_\ell$. In our setting, for the algorithmic problem, $x = c|_S$ for the set S that is output. But we cannot violate the Top-ℓ budget constraint, and we cannot consider only the large-cost entries as this would violate monotonicity. The insight is that if Top-$\ell(c|_S) \leq B_\ell$, then we can consider $T = \{e \in S : c_e > B_\ell/\ell\}$. We then have $|T| \leq \ell$, and so the *total cost* of T is at most B_ℓ. Note also, that any $A \subseteq \{e : c_e \leq B_\ell/\ell\}$ trivially satisfies Top-$\ell(c|_A) \leq B_\ell$. Thus, we can take the better of two solutions, one satisfying a total-cost constraint of B_ℓ, and the other using only elements e with $c_e \leq B_\ell/\ell$, and this leads to a factor-2 loss in approximation. But note that we have reduced the Top-ℓ-budget constraint to a standard total-budget constraint. The extension to handle k Top-ℓ budget constraints considers $k+1$ solutions, and due to truthfulness considerations, we return each of these solutions with equal probability (instead of the best solution). ◀

▶ **Corollary 5.4.** *We can obtain an randomized $O(k)$-approximation truthful mechanism for generalized budget-feasible mechanism with k Top-ℓ-budget constraints for subadditive valuations. For XOS valuations, and cardinality-based subadditive functions, the mechanisms run in polytime.*

References

1 Georgios Amanatidis, Georgios Birmpas, and Evangelos Markakis. On budget-feasible mechanism design for symmetric submodular objectives. In *Proceedings of 13th WINE*, pages 1–15, 2017.

2 Georgios Amanatidis, Georgios Birmpas, and Evangelos K. Markakis. Coverage, matching, and beyond: New results on budgeted mechanism design. In *Proceedings of 12th WINE*, pages 414–428, 2016.

3 Georgios Amanatidis, Georgios Birmpas, and Evangelos K. Markakis. On budget-feasible mechanism design for symmetric submodular objectives. In *Proceedings of 13th WINE*, pages 1–15, 2017.

4 Georgios Amanatidis, Pieter Kleer, and Guido Schäfer. Budget-feasible mechanism design for non-monotone submodular objectives: Offline and online. In *Proceedings of 20th EC*, pages 901–919, 2019.

5 Nima Anari, Gagan Goel, and Afshin Nikzad. Mechanism design for crowdsourcing: An optimal 1-1/e competitive budget-feasible mechanism for large markets. In *Proceedings of 55th FOCS*, pages 266–275, 2014.

6 Ashwinkumar Badanidiyuru, Shahar Dobzinski, and Sigal Oren. Optimization with demand oracles. *Algorithmica*, 81:2244–2269, 2019.

7 Eric Balkanski, Pranav Garimidi, Vasilis Gkatzelis, Daniel Schoepflin, and Xizhi Tan. Deterministic budget-feasible clock auctions. In *Proceedings of 33rd SODA*, pages 2940–2963, 2022.

8 Eric Balkanski and Jason Hartline. Bayesian budget feasibility with posted pricing. In *Proceedings of 25th WWW*, pages 189–203, 2016.

9 MohammadHossein Bateni, Mohammad Taghi Hajiaghayi, and Morteza Zadimoghaddam. Submodular secretary problem and extensions. *ACM Trans. Algorithms*, 9:32:1–32:23, 2013.

10 Xiaohui Bei, Ning Chen, Nick Gravin, and Pinyan Lu. Budget feasible mechanism design: from prior-free to bayesian. In *Proceedings of 44th STOC*, pages 449–458, 2012.

11 Xiaohui Bei, Ning Chen, Nick Gravin, and Pinyan Lu. Worst-case mechanism design via bayesian analysis. *SIAM Journal on Computing*, 46(4):1428–1448, 2017.

12 Robert D Carr, Lisa K Fleischer, Vitus J Leung, and Cynthia A Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of 11th SODA*, pages 106–115, 2000.

13 Deeparnab Chakrabarty and Chaitanya Swamy. Interpolating between k-median and k-center: Approximation algorithms for ordered k-median. In *Proceedings of 45th ICALP*, pages 29:1–29:14, 2018.

14 Deeparnab Chakrabarty and Chaitanya Swamy. Approximation algorithms for minimum norm and ordered optimization problems. In *Proceedings of 51st STOC*, pages 126–137, 2019.

15 Ning Chen, Nick Gravin, and Pinyan Lu. On the approximability of budget feasible mechanisms. In *Proceedings of 22nd SODA*, pages 685–699, 2011.

16 Shahar Dobzinski, Christos Papadimitriou, and Yaron Singer. Mechanisms for complement-free procurement. In *Proceedings of 12th EC*, pages 273–282, 2011.

17 Konstantinos Georgiou and Chaitanya Swamy. Black-box reductions for cost-sharing mechanism design. *Games and Economic Behavior*, 113:17–37, 2019.

18 Nick Gravin, Yaonan Jin, Pinyan Lu, and Chenhao Zhang. Optimal budget-feasible mechanisms for additive valuations. *ACM Trans. Econ. Comput.*, 8(4), 2020. doi:10.1145/3417746.

19 Thibaut Horel, Stratis Ioannidis, and S. Muthukrishnan. Budget feasible mechanisms for experimental design. In *Proceedings of 11th LATIN*, pages 719–730, 2014.

20 He Huang, Kai Han, Shuang Cui, and Jing Tang. Randomized pricing with deferred acceptance for revenue maximization with submodular objectives. In *Proceedings of 32nd WWW*, pages 3530–3540, 2023. doi:10.1145/3543507.3583477.

21 Pooya Jalaly and Éva Tardos. Simple and efficient budget feasible mechanisms for monotone submodular valuations. In *Proceedings of 14th WINE*, pages 246–263, 2018.

22 Stefano Leonardi, Gianpiero Monaco, Piotr Sankowski, and Qiang Zhang. Budget feasible mechanisms on matroids. *Algorithmica*, 83(5):1222–1237, 2021.

23 Roger B. Myerson. Optimal auction design. *Math. Oper. Res.*, 6(1):58–73, 1981.

24 Yaron Singer. Budget feasible mechanisms. In *Proceedings of 51st FOCS*, pages 765–774, 2010.

25 Yaron Singer. How to win friends and influence people, truthfully: influence maximization mechanisms for social networks. In *Proceedings of 5th WSDM*, pages 733–742, 2012.

26 Yaron Singer. Budget feasible mechanism design. *ACM SIGecom Exchanges*, 12(2):24–31, 2013.