Optimal Mechanisms for Combinatorial Auctions and Combinatorial Public Projects via Convex Rounding*

Shaddin Dughmi† Tim Roughgarden‡ Qiqi Yan§

March 10, 2014

Abstract

We design the first truthful-in-expectation, constant-factor approximation mechanisms for NP-hard cases of the welfare maximization problem in combinatorial auctions with non-identical items and in combinatorial public projects. Our results apply to bidders with valuations that are nonnegative linear combinations of gross substitutes valuations, a class that encompasses many of the most well-studied subclasses of submodular functions, including coverage functions and weighted matroid rank functions. Our mechanisms have expected polynomial running time and achieve an approximation factor of $1 - 1/e$. This approximation factor is the best possible for both problems, even for known and explicitly given coverage valuations, assuming $P \neq NP$. Recent impossibility results suggest that our results cannot be extended to a significantly larger valuation class.

Both of our mechanisms are instantiations of a new framework for designing approximation mechanisms based on randomized rounding algorithms. The high-level idea of this framework is to optimize *directly over the (random) output of the rounding algorithm*, rather than the usual (and rarely truthful) approach of optimizing over the *input* to the rounding algorithm. This framework yields truthful-in-expectation mechanisms, and these mechanisms can be implemented efficiently when the corresponding objective function is concave. For bidders with valuations in the cone generated by gross substitutes valuations, we give novel randomized rounding algorithms that lead to both a concave objective function and a $(1 - 1/e)$-approximation of the optimal welfare.

*Preliminary versions of this work appeared in *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, June 2011 and *Proceedings of the 12th ACM Conference on Electronic Commerce (EC)*, June 2011.

†Department of Computer Science, University of Southern California, SAL 234, 941 Bloom Walk, Los Angeles, CA 90089. This work was done while the author was a PhD student at Stanford University, and was supported by NSF Grant CCF-0448664 and a Siebel Foundation Scholarship. Email: shaddin@usc.edu.

‡Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Supported in part by NSF CAREER Award CCF-0448664, an ONR Young Investigator Award, an ONR PECASE Award, an AFOSR MURI grant, and an Alfred P. Sloan Fellowship. Email: tim@cs.stanford.edu.

§Google Research, Mountain View, CA 94043. This work was done while the author was a PhD student at Stanford University, and was supported by a Stanford Graduate Fellowship. Email: qiqiyan@cs.stanford.edu.

1 Introduction

1.1 Combinatorial Auctions and Combinatorial Public Projects

The overarching goal of *algorithmic mechanism design* is to design computationally efficient algorithms that solve or approximate fundamental optimization problems in which the underlying data is a priori unknown to the algorithm. The first problem that we study, which is central in both theory and practice, is welfare maximization in *combinatorial auctions (CAs)*. In a CA, there are m items for sale and n bidders vying for them. Each bidder i has a private *valuation* $v_i(S)$ for each subset S of the items.[1] The *welfare* of an allocation $S_1, \ldots, S_n$ of the items to the bidders is $\sum_{i=1}^{n} v_i(S_i)$. The second problem is welfare maximization in *combinatorial public projects (CPPs)*. Here, a public planner chooses a subset of m projects. Each player i has a private valuation $v_i(S)$ for each subset S of projects. The welfare maximization problem is to compute a subset of at most k projects that maximizes $\sum_{i=1}^{n} v_i(S)$.[2]

We consider both problems with valuations that are initially unknown to the decision-maker, so computing a near-optimal allocation requires eliciting information from the (self-interested) players, for example via a bid. A *mechanism* is a protocol that extracts such information and computes an allocation of the items or projects. We additionally allow mechanisms to charge payments to players in order to incentivize the truthful reporting of valuations. We call a (possibly randomized) mechanism "incentive-compatible" or "truthful" if every participant maximizes its expected payoff by truthfully revealing its information to the mechanism, no matter how the other participants behave (see Section 2.2 for formal definitions). A simple example of a truthful mechanism is the second-price single-item auction, which sells a good to the highest bidder at a price equal to the second-highest bid.

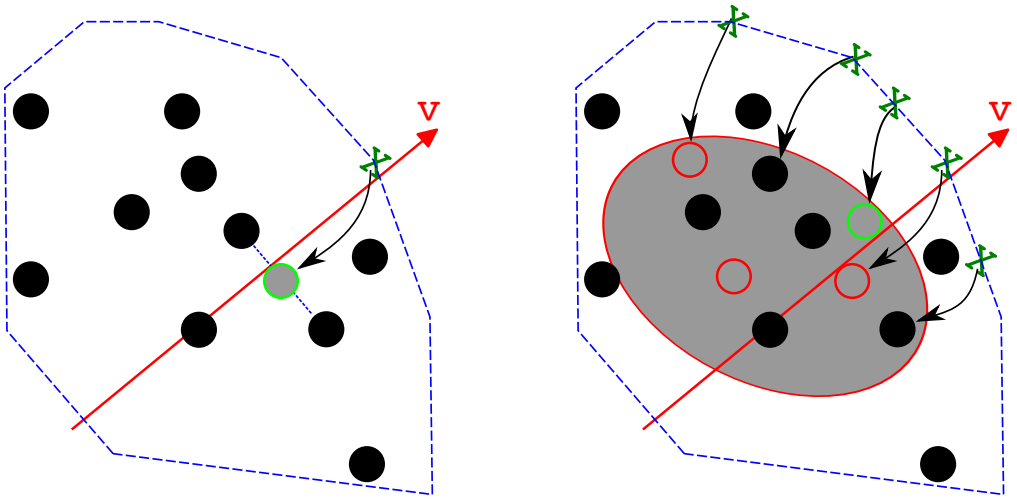
We seek mechanisms that run in time polynomial in n and m. If we drop the computational efficiency requirement, then we can solve exactly the two welfare maximization problems above using the incentive-compatible VCG mechanism (see e.g. [44]). They cannot be solved efficiently, however, except for highly restricted classes of valuations. These problems are also highly inapproximable for general valuations — even by non-truthful computationally efficient algorithms — so it is most interesting to study them for restricted classes of valuations that at least admit good approximation algorithms.

The "holy grail" of algorithmic mechanism design is to design polynomial-time truthful approximation mechanisms that match the approximation guarantee of the best (non-truthful) polynomial-time approximation algorithms. Unfortunately, several recent impossibility results have shed serious doubt on the possibility of this goal in general [15, 47, 7, 8, 12, 25, 20].[3] This paper provides such positive results for welfare maximization in CAs and CPPs for a fundamental class of valuations, via a novel randomized mechanism design framework based on convex optimization.

[1] Each bidder has an exponential number of private values; we ignore the attendant representation issues for the moment.

[2] This is sometimes called the *flexible* version of the CPP problem, as opposed to the *exact* version in which the planner must choose exactly k projects. This distinction is uninteresting from an approximation algorithms standpoint (assuming monotone valuations), but is quite important in algorithmic mechanism design; see [12] for a detailed discussion.

[3] The impressively general positive results for implementations in Bayes-Nash equilibria that were recently obtained in [32, 31, 2] do not apply to the stronger incentive-compatibility notion used here.



(a) Randomized rounding applied to an optimal solution to LP

(b) Optimizing over output of randomized rounding algorithm.

Figure 1: Optimizing over the input of a rounding scheme vs its output

1.2 The Challenge of Algorithmic Mechanism Design

Algorithmic mechanism design is difficult because incentive compatibility severely limits how an algorithm can compute an outcome, and this prohibits use of most of the ingenious approximation algorithms that have been developed for different optimization problems. The incentive-compatibility constraint necessitates the design of carefully crafted approximation algorithms, tailored specifically for truthfulness. Understanding the power of these truthful approximation mechanisms is the central goal of algorithmic mechanism design. This research agenda was first advocated by Nisan and Ronen [43]. Since then, combinatorial auctions and combinatorial public projects have emerged as the paradigmatic "challenge problems" of the field, with much work in recent years establishing upper and lower bounds on truthful polynomial-time mechanisms for these problems (e.g. [36, 16, 18, 17, 15, 11, 19, 47, 7, 8, 12, 24, 21, 25, 20]).

The most general approach known for designing (randomized) truthful mechanisms is via *maximal-in-distributional range (MIDR) algorithms* [13, 22]. An MIDR algorithm fixes a set of distributions over feasible solutions — the *distributional range* — independently of the valuations reported by the self-interested participants, and outputs a random sample from the distribution that maximizes expected (reported) welfare. An analog of the payments of the VCG mechanism can be used to extend an MIDR algorithm to a truthful (in expectation) mechanism.

Most approximation algorithms are not MIDR algorithms. Consider, as an example, a *randomized rounding* algorithm for welfare maximization in combinatorial auctions (e.g. [26, 18]). We can view such an algorithm as the composition of two algorithms, a *relaxation algorithm* and a *rounding algorithm* (see Figure 1(a)). The relaxation algorithm is deterministic and takes as input the problem data (players' valuations v), and outputs the (fractional) solution to a linear programming relaxation of the welfare-maximization problem that is optimal for the objective function defined by v. The rounding algorithm is randomized and takes as input this fractional solution and outputs a feasible allocation of the items to the players. Taken together, these algorithms assign

to each input v a probability distribution $D(v)$ over integral allocations. For almost all known randomized rounding algorithms, there is an input v such that the expected objective function value $\mathrm{E}_{y\sim D(v)}[v^T y]$ with the distribution $D(v)$ is inferior to that $\mathrm{E}_{y\sim D(w)}[v^T y]$ with a distribution $D(w)$ that the algorithm would produce for a different input w — and this is a violation of the MIDR property. Informally, such violations are inevitable unless a rounding algorithm is designed explicitly to avoid them, on top of the usual approximation requirements.

The exception that proves the rule is the important and well-known mechanism design framework of Lavi and Swamy [36]. Lavi and Swamy [36] begin with the foothold that the *fractional* welfare maximization problem — the relaxation algorithm above — can be made truthful by charging appropriate VCG payments. Further, they identify a very special type of rounding algorithm that preserves truthfulness: if the expected allocation produced by the rounding algorithm is *always identical to the input to the rounding algorithm*, component-wise, up to some universal scaling factor α, then composing the two algorithms easily yields an α-approximate truthful-in-expectation mechanism (after scaling the fractional VCG payments by α). Perhaps surprisingly, there are some interesting problems, such as welfare maximization in CAs with general valuations, that admit such a rounding algorithm with a best-possible approximation guarantee (assuming $P \neq NP$). However, most NP-hard welfare maximization problems, including CAs and CPPs with restricted valuation classes, do not seem to admit good randomized rounding algorithms of the rigid type required by this design framework.

1.3 Our Contributions

We introduce a new approach to designing truthful-in-expectation approximation mechanisms based on randomized rounding algorithms; we outline it here for the special case of welfare maximization in combinatorial auctions. The high-level idea is to optimize *directly on the outcome of the rounding algorithm*, rather than merely on the outcome of the relaxation algorithm (the *input* to the rounding algorithm). See also Figure 1(b). In other words, let $r(x)$ denote a randomized rounding algorithm, from fractional allocations to integer allocations. Given players' valuations v, we compute a fractional allocation x that maximizes the expected welfare $\mathrm{E}_{y\sim r(x)}[v^T y]$ over all fractional allocations x. This methodology evidently gives MIDR algorithms. This optimization problem is often intractable, but when the rounding algorithm r and the space of valuations v are such that the function $\mathrm{E}_{y\sim r(x)}[v^T y]$ is always concave in x — in which case we call r a *convex rounding algorithm* — it can be solved in polynomial time using convex programming (modulo numerical issues that we address later).

For our first main result, we use this design framework to give an expected polynomial-time, truthful-in-expectation, $(1 - 1/e)$-approximation mechanism for welfare maximization in CAs in which every bidder's valuation belongs to a rich subclass of submodular valuations.[4] This subclass consists of all nonnegative linear combinations of gross substitutes (GS) valuations, which we denote by CGS (where C stands for "cone"); see Section 2.5 for formal definitions. In addition to all GS valuations, this set encompasses many of the concrete examples of submodular functions that have been studied in the combinatorial auctions literature, including coverage functions and matroid weighted rank functions (see Appendix A.1 for formal definitions). Our mechanism treats valuations as "black boxes," and only assumes that they support a randomized analog of a "value oracle." We

[4] A set function $f : 2^X \to \mathbb{R}$ is *submodular* if it exhibits diminishing marginal returns; i.e., $f(T \cup \{j\}) - f(T) \leq f(S \cup \{j\}) - f(S)$ whenever $S \subseteq T$.

also give an explicit (non-oracle-based) implementation of the mechanism for explicitly represented coverage valuations.

Our approximation guarantee is optimal, assuming $P \neq NP$, even for the welfare maximization problem with known and and explicitly presented coverage valuations [35]. Our mechanism is the first truthful-in-expectation and polynomial-time mechanism to achieve a constant-factor approximation for an NP-hard class of CAs without assuming that there are a logarithmic (in m) number of copies of every type of item. Recent negative results for truthful approximation mechanisms for CAs with bidders with general submodular valuations [25, 20] suggest that our positive results cannot be extended to a significantly larger valuation class.

Our second result is a $(1 - 1/e)$-approximate truthful-in-expectation mechanism for welfare maximization for CPPs with CGS valuations that runs in expected polynomial-time. This is the best approximation possible for this problem, even with only GS valuations and without truthfulness, unless $P = NP$. Therefore, ours is the first truthful mechanism for an NP-hard variant of CPP that matches the approximation ratio of the best non-truthful algorithm. Again, recent impossibility results [25, 20] suggest that this guarantee cannot be extended to significantly more general valuations.

Both of our mechanisms are instantiations of our convex rounding framework. For CAs, we use a randomized rounding algorithm that allocates each item independently. We show that standard randomized rounding does not yield a convex rounding algorithm, but composing it with a suitable transformation of the optimal fractional solution does (for CGS valuations). We also show that at least a $(1 - 1/e)$ fraction of the expected welfare of a fractional solution is preserved by this transformation.

For CPPs, to respect the cardinality constraint of k on the set of chosen projects, our rounding scheme cannot round different items independently. While the expected value of a submodular function over a product distribution (i.e., with independent rounding) has been studied extensively, and is closely related to the now well-understood multi-linear extension (see e.g. [9, 50]), proving the necessary convexity and approximation results for our dependent distribution requires new ideas. We address this technical challenge by combining ideas from combinatorics and convex analysis.

1.4 Additional Discussion of Related Work

1.4.1 Combinatorial Auctions

For combinatorial auctions, we discuss only the results most pertinent to this work; see [10] for an introduction to the topic and [5] for a survey of truthful approximation mechanisms for combinatorial auctions.

For the welfare maximization problem in CAs with general valuations (assuming only that $v_i(\emptyset) = 0$ and that $v_i(S) \leq v_i(T)$ whenever $S \subseteq T$), the best approximation factor possible by a polynomial-time approximation algorithm is roughly $\min\{\sqrt{m}, n\}$, where n is the number of bidders and m is the number of items. There are comparable unconditional lower bounds for polynomial communication and unbounded computation [39] and hardness of approximation results for various classes of succinctly represented valuations [38].

These strong negative results for welfare maximization with general valuations motivate the study of important special cases. Numerous special cases have been considered (see [5, Fig 1.2]), and the most well-studied one is for bidders with submodular valuations. Without incentive-compatibility constraints, the welfare maximization problem with submodular bidder valuations is

completely solved. Vondrák [50] gave a $(1-\frac{1}{e})$-approximation algorithm for the problem, improving over the $\frac{1}{2}$-approximation given in [37]. The algorithm in [50] works in the *value oracle* model, where each valuation v is modeled as a "black box" that returns the value $v(S)$ of a queried set S in a single operation. The approximation factor of $1-\frac{1}{e}$ is unconditionally optimal in the value-oracle model (for polynomial communication) [39], and is also optimal (for polynomial time) for certain succinctly represented submodular valuations, assuming $P \neq NP$ [35]. The result of [35] implies that $1-1/e$ is the optimal approximation factor in our model as well, assuming $P \neq NP$.[5]

Despite intense study, prior to this work, there were no truthful-in-expectation and polynomial-time constant-factor approximation mechanisms for welfare maximization in CAs with any non-trivial subclass of submodular bidder valuations. The best previous results, which apply to all submodular valuations, are a truthful-in-expectation $O\left(\frac{\log m}{\log \log m}\right)$ approximation mechanism in the communication complexity model due to Dobzinski, Fu and Kleinberg [14], and a universally-truthful[6] $O(\log m \log \log m)$ approximation mechanism in the demand oracle model due to Dobzinski [11].

The aforementioned works [14, 36] are precursors to our general design framework that optimizes directly over the output of a randomized rounding algorithm. In the framework of Lavi and Swamy [36], the input to and output of the rounding algorithm are assumed to coincide up to a scaling factor, so optimizing over its input (as they do) is equivalent to optimizing over its output (as we do). In the result of Dobzinski et al. [14], optimizing with respect to their "proxy bidders" is equivalent to optimizing over the output of a particular randomized rounding algorithm.

1.4.2 Combinatorial Public Projects

Combinatorial Public Projects, in particular its *exact* variant, was first introduced by Papadimitriou, Schapira, and Singer [47]. They show that no deterministic truthful mechanism for exact CPP with submodular valuations can guarantee better than a $O(\sqrt{m})$ approximation to the optimal social welfare. The non-strategic version of the problem, on the other hand, is equivalent to maximizing a submodular function subject to a cardinality constraint, and admits a $(1-1/e)$-approximation algorithm [41]. This is optimal, assuming $P \neq NP$ [48].

Buchfuhrer, Schapira, and Singer [8] explored approximation algorithms and truthful mechanisms for CPP with various classes of valuations in the submodular hierarchy. The most relevant result of [8] to our paper is a lower-bound of $O(\sqrt{m})$ on *deterministic* truthful mechanisms for the exact variant of CPP with coverage valuations — a class of valuations for which our *randomized* mechanism for flexible CPP obtains a $(1-1/e)$ approximation.

Dobzinski [12] showed two lower bounds for CPP in the value oracle model: A lower bound of $O(\sqrt{m})$ on universally truthful mechanisms for flexible CPP with submodular valuations, and a lower bound of $O(\sqrt{m})$ on truthful-in-expectation mechanisms for *exact* CPP with submodular valuations.

[5] We show in Appendix B.1 that our oracle model is no more powerful than polynomial-time computation in the special case of explicitly represented coverage functions, for which $1-1/e$ is optimal assuming $P \neq NP$ [35]. In contrast, the work of [29] improves on the approximation factor of $1-1/e$ by using *demand oracles*, which for coverage functions can encode the NP-hard Set Cover problem.

[6] A mechanism is universally-truthful if, for every realization of the mechanism's coins, each player maximizes his payoff by bidding truthfully. Universally truthful mechanisms are defined formally in Section 2.2.

1.4.3 More Recent Work

Since the initial publication of the work in the paper, impossibility results in [25, 20] have ruled out polynomial-time constant-approximation truthful mechanisms for combinatorial auctions and public projects with general submodular valuations. The results of [25] hold in the value oracle model, and those of [20] hold in the computational complexity model assuming $NP \not\subseteq P/poly$. Moreover, the convex rounding framework has been employed in [33] to design truthful auctions for secondary spectrum usage in wireless networks.

2 Preliminaries

2.1 Optimization Problems

We consider optimization problems Π of the following general form. Each instance of Π consists of a *feasible set* $\mathcal{S}$, and an *objective function* $w : \mathcal{S} \to \mathbb{R}$. The solution to an instance of Π is given by the following optimization problem.

$$\begin{array}{ll} \text{maximize} & w(x) \\ \text{subject to} & x \in \mathcal{S}. \end{array} \tag{1}$$

2.2 Mechanism Design Basics

We consider mechanism design optimization problems of the form in (1). In such problems, there are n players, where each player i has a *valuation function* $v_i : \mathcal{S} \to \mathbb{R}$. We are concerned with *welfare maximization* problems, where the objective is $w(x) = \sum_{i=1}^{n} v_i(x)$.

We consider direct-revelation mechanisms for optimization mechanism design problems. Such a mechanism comprises an *allocation rule*, which is a function from (hopefully truthfully) reported valuation functions $v_1, \ldots, v_n$ to an outcome $x \in \mathcal{S}$, and a *payment rule*, which is a function from reported valuation functions to a required payment from each player. We allow the allocation and payment rules to be randomized.

A mechanism with allocation and payment rules $\mathcal{A}$ and p is *truthful-in-expectation* if every player always maximizes its expected payoff by truthfully reporting its valuation function, meaning that

$$\mathbf{E}[v_i(\mathcal{A}(v)) - p_i(v)] \geq \mathbf{E}[v_i(\mathcal{A}(v'_i, v_{-i})) - p_i(v'_i, v_{-i})] \tag{2}$$

for every player i, (true) valuation function v_i, (reported) valuation function v'_i, and (reported) valuation functions v_{-i} of the other players. The expectation in (2) is over the coin flips of the mechanism. If (2) holds for every flip of the coins, rather than merely in expectation, we call the mechanism *universally truthful*.

The mechanisms that we design can be thought of as randomized variations on the classical VCG mechanism, as we explain next. Recall that the *VCG mechanism* is defined by the (generally intractable) allocation rule that selects the welfare-maximizing outcome with respect to the reported valuation functions, and the payment rule that charges each player i a bid-independent "pivot term" minus the reported welfare earned by other players in the selected outcome. This (deterministic) mechanism is truthful; see e.g. [42].

Now let $dist(\mathcal{S})$ denote the probability distributions over a feasible set $\mathcal{S}$, and let $\mathcal{D} \subseteq dist(\mathcal{S})$ be a compact subset of them. The corresponding *Maximal in Distributional Range (MIDR)* allocation rule is defined as follows: given reported valuation functions $v_1, \ldots, v_n$, return an outcome that is sampled randomly from a distribution $D^* \in \mathcal{D}$ that maximizes the expected welfare $\mathbf{E}_{x \sim D}[\sum_i v_i(x)]$ over all distributions $D \in \mathcal{D}$. Analogous to the VCG mechanism, there is a (randomized) payment rule that can be coupled with this allocation rule to yield a truthful-in-expectation mechanism (see [13]).

2.3 Combinatorial Auctions

In *Combinatorial Auctions* there is a set $[m] = \{1, 2, \ldots, m\}$ of items, and a set $[n] = \{1, 2, \ldots, n\}$ of players. Each player i has a valuation function $v_i : 2^{[m]} \to \mathbb{R}^+$ that is normalized ($v_i(\emptyset) = 0$) and monotone ($v_i(A) \leq v_i(B)$ whenever $A \subseteq B$). A feasible solution is an *allocation* $(S_1, \ldots, S_n)$, where S_i denotes the items assigned to player i, and $\{S_i\}_i$ are mutually disjoint subsets of $[m]$. Player i's value for outcome $(S_1, \ldots, S_n)$ is equal to $v_i(S_i)$. The goal is to choose the allocation maximizing *social welfare*: $\sum_i v_i(S_i)$.

2.4 Combinatorial Public Projects

In *Combinatorial Public Projects* there is a set $[m] = \{1, \ldots, m\}$ of *projects*, a cardinality bound k such that $0 \leq k \leq m$, and a set $[n] = \{1, \ldots, n\}$ of *players*. Each player i has a valuation function $v_i : 2^{[m]} \to \mathbb{R}^+$ that is normalized ($v_i(\emptyset) = 0$) and monotone ($v_i(A) \leq v_i(B)$ whenever $A \subseteq B$). In this paper, we consider the *flexible* variant of combinatorial public projects: a feasible solution is a set $S \subseteq [m]$ of projects with $|S| \leq k$. Player i's value for outcome S is equal to $v_i(S)$. The goal is to choose the feasible set S maximizing *social welfare*: $\sum_i v_i(S)$.

2.5 Gross Substitutes and CGS Valuations

We now define the valuation class to which our results apply. We begin with the fundamental class of gross substitutes (GS) valuations. For a valuation $v : 2^{[m]} \to \mathbb{R}^+$ and a price vector $p : [m] \to \mathbb{R}$, the *demand set* $D(v, p)$ is the set $\arg\max_{S \subseteq [m]}\{v(S) - \sum_{j \in S} p(j)\}$ of utility-maximizing bundles at the prices p. The usual definition of gross substitutes (GS) valuations states that increasing the price of some goods can only increase the demand for other goods.

Definition 2.1 ([34])**.** *A valuation $v : 2^{[m]} \to \mathbb{R}^+$ satisfies* gross substitutes *if for every pair $p, q : [m] \to \mathbb{R}$ of price vectors with $q(j) \geq p(j)$ for all $j \in [m]$ and every bundle $S \in D(v, p)$, there exists a set $T \in D(v, q)$ containing $\{j \in S : q(j) = p(j)\}$.*

The following price-free characterization of GS valuations is more convenient for our purposes. It follows from combining results in [37] and [4]; see also the survey [46].

Proposition 2.2. *A valuation $v : 2^{[m]} \to \mathbb{R}^+$ is in GS if and only if*

$$\mathcal{F}^v_S(i, j) \geq \min\{\mathcal{F}^v_S(i, k), \mathcal{F}^v_S(k, j)\}$$

for every $S \subseteq [m]$ and $i, j, k \in S$, where $\mathcal{F}^v_S(i, j)$ is the deficit in additivity of i and j given the set S:

$$\mathcal{F}^v_S(i, j) = (v(S \cup \{i\}) - v(S)) + (v(S \cup \{j\}) - v(S)) - (v(S \cup \{i, j\}) - v(S)). \tag{3}$$

It was shown in [30] that GS valuations are submodular. One interesting subclass of GS valuations is the set of matroid weighted rank functions (see Appendix A.1 for a review of relevant concepts from matroid theory). Welfare-maximization in combinatorial auctions with GS bidder valuations can be solved in polynomial time [40].

GS valuations are not closed under addition; in fact, even the sum of matroid rank functions may not be in GS — see [46] for an example. We therefore obtain a more general valuation class by considering all nonnegative linear combinations of GS valuations — the cone generated by gross substitutes valuations (CGS). Since submodular valuations are closed under nonnegative linear combinations, CGS valuations are submodular. CGS valuations include most concrete examples of monotone submodular valuations that appear in the literature, including coverage functions.[7] Moreover, if $P \neq NP$ and without regard to incentive-compatibility, $1-1/e$ is the best approximation possible in polynomial time for combinatorial auctions [35] when players have CGS valuations and for combinatorial public projects [48] even when players have GS valuations.[8] That said, we note that some interesting submodular functions — such as some budget additive functions[9] — are not in the CGS family (see Appendix B.3).

2.6 Lotteries and Oracles

A *value oracle* for a valuation $v : 2^{[m]} \to \mathbb{R}$ takes as input a set $S \subseteq [m]$, and returns $v(S)$. We define two analogous oracles, one for use in our mechanism for combinatorial auctions and the other for combinatorial public projects, that each take as input a description of a simple lottery over subsets of $[m]$, and output the expectation of v over this lottery. While our "lottery oracles" can not be implemented efficiently for all explicitly represented CGS functions, they nevertheless can be approximated arbitrarily well with high probability using random sampling, and can efficiently be implemented exactly for explicitly represented coverage valuations — we discuss the details in Appendix B.1.

2.6.1 Product-lottery-value oracles

First, we define product-lottery-value oracles, which we will use in our mechanism for combinatorial auctions. Given a vector $x \in [0,1]^m$ of probabilities on the items, let D_x be the distribution over $S \subseteq [m]$ that includes each item j in S independently with probability x_j. We use $F_v(x)$ to denote the expected value of $v(S)$ over draws $S \sim D_x$ from this lottery.

Definition 2.3. *A* product-lottery-value oracle *for set function $v : 2^{[m]} \to \mathbb{R}$ takes as input a vector $x \in [0,1]^m$, and outputs*

$$F_v(x) = \mathop{\mathrm{E}}_{S \sim D_x}[v(S)] = \sum_{S \subseteq [m]} v(S) \prod_{j \in S} x_j \prod_{j \neq S} (1 - x_j). \tag{4}$$

[7] A *coverage function* f on ground set $[m]$ designates some set $\mathcal{L}$ of elements, and m subsets $A_1, \ldots, A_m \subseteq \mathcal{L}$, such that $f(S) = |\cup_{j \in S} A_j|$. We note that $\mathcal{L}$ may be an infinite, yet measurable, space. Coverage functions are arguably *the* canonical example of a submodular function, particularly for combinatorial auctions. For more background, see Appendix A.1.

[8] The result of [48] holds when the sum of players' valuations in CPP may be an arbitrary coverage function. Since a coverage function can be expressed as the sum of GS functions (see Appendix A.1), this implies hardness for CPP with GS valuations.

[9] A set function f on ground set $[m]$ is *budgeted additive* if there exists a constant $B \geq 0$ (the budget) such that $f(S) = \min(B, \sum_{j \in S} f(\{j\}))$.

We note that F_v is simply the well-studied *multi-linear extension* of v (see for example [9, 50]).

2.6.2 Bounded-lottery-value oracles

Our mechanism for combinatorial public projects requires evaluating a player's expected value for a different type of lotteries. Let $k \in [m]$, let $R \subseteq [m]$, and let $x \in [0,1]^m$ be a vector such that $\sum_j x_j \leq 1$. We interpret x as a probability distribution over $[m] \cup \{*\}$, where $*$ represents not choosing a project. Specifically, project $j \in [m]$ is chosen with probability x_j, and $*$ is chosen with probability $1 - \sum_j x_j$. We define a distribution $D_k^R(x)$ over $2^{[m]}$, and call this distribution the *k-bounded lottery with marginals x and promise R*. We sample $S \sim D_k^R(x)$ as follows: Let $j_1, \ldots, j_k$ be independent draws from x, and let $S = R \cup \{j_1, \ldots, j_k\} \setminus \{*\}$. Essentially, this lottery commits to choosing projects R, and adds an additional k projects chosen randomly with replacement from distribution x. When $R = \emptyset$, as will be the case through most of this paper, we omit mention of the promised set.

Definition 2.4. *A* bounded-lottery-value oracle *for set function $v : 2^{[m]} \to \mathbb{R}$ takes as input a vector $x \in [0,1]^m$ with $\sum_j x_j \leq 1$, a bound $k \in [m]$, and a set $R \subseteq [m]$, and outputs $\mathrm{E}_{S \sim D_k^R(x)}[v(S)]$.*

3 The Convex Rounding Framework

3.1 Relaxations and Rounding Schemes

Let Π be an optimization problem. A *relaxation* Π' of Π defines for every $(\mathcal{S}, w) \in \Pi$ a convex and compact *relaxed feasible set* $\mathcal{R} \subseteq \mathbb{R}^m$ that is independent of w (we suppress the dependence on $\mathcal{S}$); and an *extension* $w_{\mathcal{R}} : \mathcal{R} \to \mathbb{R}$ of the objective w to the relaxed feasible set $\mathcal{R}$. This gives the following *relaxed optimization problem*.

$$\begin{array}{ll} \text{maximize} & w_{\mathcal{R}}(x) \\ \text{subject to} & x \in \mathcal{R}. \end{array} \tag{5}$$

Generally, the extension is defined so that it is computationally tractable to find a point $x \in \mathcal{R}$ that maximizes $w_{\mathcal{R}}(x)$ (possibly approximately).

For example, $\mathcal{S}$ could be the allocations of m items to n bidders in a combinatorial auction, $w(x)$ the welfare of an allocation, $\mathcal{R}$ the feasible region of a linear programming relaxation, and $w_{\mathcal{R}}$ the natural linear extension of w to fractional allocations.

The solution $x \in \mathcal{R}$ to the relaxed problem need not be in $\mathcal{S}$. A *rounding scheme* for relaxation Π' of Π defines for each feasible set $\mathcal{S}$ of Π, and its corresponding relaxed set $\mathcal{R}$, a (possibly randomized) function $r : \mathcal{R} \to \mathcal{S}$. Since our rounding scheme will be randomized, we will frequently use $r(x)$ to denote the distribution over $\mathcal{S}$ resulting from rounding the point $x \in \mathcal{R}$. Commonly, the rounding scheme satisfies the following approximation guarantee: $\mathrm{E}_{y \sim r(x)}[w(y)] \geq \alpha \cdot w_{\mathcal{R}}(x)$ for every $x \in \mathcal{R}$. In this case, if x^* maximizes $w_{\mathcal{R}}$ over $\mathcal{R}$ and $w_{\mathcal{R}}$ agrees with w on $\mathcal{S}$, then $\mathrm{E}_{y \sim r(x^*)}[w(y)] \geq \alpha \cdot \max_{y \in \mathcal{S}} w(y)$.

3.2 Convex Rounding Schemes and MIDR

Our technique is motivated by the following observation: instead of solving the relaxed problem and subsequently rounding the solution, why not *optimize directly on the outcome of the rounding*

scheme? In particular, consider the following relaxation of Π that "absorbs" rounding scheme r into the objective.

$$\begin{array}{ll} \text{maximize} & \mathrm{E}_{y\sim r(x)}[w(y)] \\ \text{subject to} & x \in \mathcal{R}. \end{array} \tag{6}$$

The solution to this problem rounds to the best possible distribution in the range of the rounding scheme, over all possible fractional solutions in $\mathcal{R}$. While this problem is often intractable, it always leads to an MIDR allocation rule.

Algorithm 1 MIDR Allocation Rule via Optimizing over Output of Rounding Scheme

Parameter: Feasible set $\mathcal{S}$ of Π.
Parameter: Relaxed feasible set $\mathcal{R} \subseteq \mathbb{R}^m$.
Parameter: (Randomized) rounding scheme $r : \mathcal{R} \to \mathcal{S}$.
Input: Objective $w : \mathcal{S} \to \mathbb{R}$ satisfying $(\mathcal{S}, w) \in \Pi$.
Output: Feasible solution $z \in \mathcal{S}$.
1: Let x^* maximize $\mathrm{E}_{y\sim r(x)}[w(y)]$ over $x \in \mathcal{R}$.
2: Let $z \sim r(x^*)$

Lemma 3.1. *Algorithm 1 is an MIDR allocation rule.*

We say a rounding scheme $r : \mathcal{R} \to \mathcal{S}$ is *α-approximate* for $\alpha \leq 1$ if $w(x) \geq \mathrm{E}_{y\sim r(x)}[w(y)] \geq \alpha \cdot w(x)$ for every $x \in \mathcal{S}$. When r is α-approximate, so is the allocation rule of Algorithm 1.

Lemma 3.2. *If r is an α-approximate rounding scheme, then Algorithm 1 returns an α-approximate solution (in expectation) to the original optimization problem* (1)*.*

For most rounding schemes in the approximation algorithms literature, the optimization problem (6) cannot be solved in polynomial time (assuming $P \neq NP$). The reason is that for any rounding scheme that always rounds a feasible solution to itself – i.e., $r(x) = x$ for all $x \in \mathcal{S}$ — an optimal solution to (6) is also optimal for (1). Thus, in this case, hardness of the original problem (1) implies hardness of (6). We conclude that we need to design rounding schemes with the unusual property that $r(x) \neq x$ for some $x \in \mathcal{S}$.

We call a (randomized) rounding scheme $r : \mathcal{R} \to \mathcal{S}$ *convex* if $\mathrm{E}_{y\sim r(x)}[w(y)]$ is concave function of $x \in \mathcal{R}$. Convex rounding schemes induce convex optimization problems.

Lemma 3.3. *When r is a convex rounding scheme,* (6) *is a convex optimization problem.*

Under additional technical conditions, discussed in the context of combinatorial auctions in Appendix C.1, and in the context of combinatorial public projects in Appendix C.2, the convex program (6) can be solved efficiently (e.g., using the ellipsoid method). This reduces the design of a polynomial-time α-approximate MIDR algorithm to designing a polynomial-time α-approximate convex rounding scheme.

Summarizing, Lemmas 3.1, 3.2, and 3.3 give the following informal theorem.

Theorem 3.4. *(Informal) Let Π be a welfare-maximization optimization problem, and let Π' be a relaxation of Π. If there exists a polynomial-time, α-approximate, convex rounding scheme for Π', then there exists a truthful-in-expectation, polynomial-time, α-approximate mechanism for Π.*

Of course, there is no reason a priori to believe that useful convex rounding schemes – let alone ones computable in polynomial time – exist for any important problems. We show in Sections 4 and 5 that they do in fact exist for combinatorial auctions and public projects with CGS valuations.

4 Combinatorial Auctions

In this section, we use the framework of Section 3 to prove our result for combinatorial auctions.

Theorem 4.1. *There is a $(1-1/e)$-approximate, truthful-in-expectation mechanism for combinatorial auctions with CGS valuations in the product-lottery-value oracle model, running in expected* $\mathrm{poly}(n,m)$ *time.*

We formulate welfare maximization in combinatorial auctions as an optimization problem as follows. An instance $(\mathcal{S}, w)$ of combinatorial auctions is given by the following integer program with feasible set $\mathcal{S}$ contained in $\{0,1\}^{n\times m}$. Variable x_{ij} indicates whether item j is allocated to player i, and $w(x)$ denotes the social welfare of allocation x.

$$\begin{array}{lll} \text{maximize} & w(x) = \sum_i v_i(\{j : x_{ij} = 1\}) & \\ \text{subject to} & \sum_i x_{ij} \le 1, & \text{for } j \in [m]. \\ & x_{ij} \in \{0,1\}, & \text{for } i \in [n], j \in [m]. \end{array} \tag{7}$$

We let the relaxed feasible set $\mathcal{R} = \mathcal{R}(\mathcal{S})$ be the result of relaxing the constraints $x_{ij} \in \{0,1\}$ of (7) to $0 \le x_{ij} \le 1$.

We structure the proof of Theorem 4.1 as follows. We define the *Poisson rounding scheme*, which we denote by r_{poiss}, in Section 4.1. We prove that r_{poiss} is $(1-1/e)$-approximate (Lemma 4.3), and convex (Lemma 4.2). Lemmas 3.1, 3.2 and 4.3, taken together, imply that Algorithm 1 when instantiated for combinatorial auctions with $r = r_{\text{poiss}}$, is a $(1-1/e)$-approximate MIDR allocation rule. Lemma 4.2 reduces implementing this allocation rule to solving a convex program.

In Appendix C, we handle the technical and numerical issues related to solving convex programs. First, we prove that our instantiation of Algorithm 1 for combinatorial auctions can be implemented in expected polynomial-time using the ellipsoid method under a simplifying assumption on the numerical conditioning of our convex program (Lemma C.2). Then we show in Section C.1.3 that the previous assumption can be removed by slightly modifying our rounding scheme.

Finally, Proposition B.3 in Appendix B.2 shows that we truth-telling VCG payments for our mechanism can be computed efficiently. This completes the proof of Theorem 4.1.

4.1 The Poisson Rounding Scheme

In this section we define the *Poisson rounding scheme*, which we denote by r_{poiss}. When instantiated for combinatorial auctions with $r = r_{\text{poiss}}$, Algorithm 1 reduces to solving the following optimization problem.

$$\begin{array}{lll} \text{maximize} & f(x) = \mathrm{E}_{y\sim r_{\text{poiss}}(x)}[w(y)] & \\ \text{subject to} & \sum_i x_{ij} \le 1, & \text{for } j \in [m]. \\ & 0 \le x_{ij} \le 1, & \text{for } i \in [n], j \in [m]. \end{array} \tag{8}$$

We define the Poisson rounding scheme as follows. Given a fractional solution x to (8), do the following independently for each item j: assign j to player i with probability $1-e^{-x_{ij}}$. (This is well defined since $1-e^{-x_{ij}} \le x_{ij}$ for all players i and items j, and $\sum_i x_{ij} \le 1$ for all items j.) We make this more precise in Algorithm 2. For clarity, we represent an allocation as a function from items to players, with an additional null player $*$ reserved for items that are left unassigned. The Poisson rounding scheme is $(1-1/e)$-approximate and convex. The proof of Lemma 4.3 is not difficult, and is included below. We prove Lemma 4.2 in Section 4.3. As a warm-up, we first present a simplified proof of Lemma 4.2 for the special case of coverage valuations in Section 4.2.

Algorithm 2 The Poisson Rounding Scheme r_{poiss}

Input: Fractional allocation x with $\sum_i x_{ij} \leq 1$ for all j, and $0 \leq x_{ij} \leq 1$ for all i, j.
Output: Feasible allocation $a : [m] \to [n] \cup \{*\}$.

```
for j = 1, ..., m do
    Draw p_j uniformly at random from [0, 1].
    if Σ_i (1 − e^{−x_ij}) ≥ p_j then
        Let a(j) be the minimum index such that Σ_{i≤a(j)} (1 − e^{−x_ij}) ≥ p_j.
    else
        a(j) = *
    end if
end for
```

Lemma 4.2. *The Poisson rounding scheme is convex when player valuations are in CGS.*

Lemma 4.3. *The Poisson rounding scheme is* $(1 - 1/e)$*-approximate when player valuations are submodular.*

Proof. Let $S_1, \ldots, S_n$ be an allocation, and let x be an the integer point of (8) corresponding to $S_1, \ldots, S_n$. Let $(S'_1, \ldots, S'_n) \sim r_{\text{poiss}}(x)$. It suffices to show that $\mathrm{E}[\sum_i v_i(S'_i)] \geq (1-1/e) \cdot \sum_i v_i(S_i)$.

By definition of the Poisson rounding scheme, S'_i includes each $j \in S_i$ with probability $1 - 1/e$. Submodularity implies that $\mathrm{E}[v_i(S'_i)] \geq (1-1/e) \cdot v_i(S_i)$ — this can be proved by a simple induction on $|S_i|$, and has been previously shown in many contexts: see for example [28, Lemma 2.2], and the earlier related result in [26, Proposition 2.3]. This completes the proof. □

4.2 Warm-up: Convexity for Coverage Valuations

In this section, we prove the special case of Lemma 4.2 for coverage valuations, as defined in Section 2.5. Fix n, m, and coverage valuations $\{v_i\}_{i=1}^n$, and let $\mathcal{R}$ denote the feasible set of mathematical program (8). Let $(S_1, \ldots, S_n) \sim r_{\text{poiss}}(x)$ be the (random) allocation computed by the Poisson rounding scheme for point $x \in \mathcal{R}$. The expected welfare $\mathrm{E}[w(r_{\text{poiss}}(x))]$ can be written as $\mathrm{E}[\sum_{i=1}^n v_i(S_i)]$, where the expectation is taken over the internal random coins of the rounding scheme. By linearity of expectation, as well as the fact that the sum of concave functions is concave, it suffices to show that $\mathbf{E}[v_i(S_i)]$ is a concave function of x for an arbitrary player i with coverage valuation v_i.

Fix player i, and use x_j, v, and S as short-hand for x_{ij}, v_i, and S_i respectively. Recall that v is a coverage function; let $\mathcal{L}$ be a ground set and $A_1, \ldots, A_m \subseteq \mathcal{L}$ be such that $v_i(T) = |\cup_{j \in T} A_j|$ for each $T \subseteq [m]$. The Poisson rounding scheme includes each item j in S independently with probability $1 - e^{-x_j}$. The expected value of player i can be written as follows.

$$\begin{aligned} \mathrm{E}\left[v(S)\right] &= \mathrm{E}[|\cup_{j \in S} A_j|] \\ &= \sum_{\ell \in \mathcal{L}} \Pr[\ell \in \cup_{j \in S} A_j] \end{aligned}$$

Since the sum of concave functions is concave, it suffices to show that $\Pr[\ell \in \cup_{j \in S} A_j]$ is concave in x for each $\ell \in \mathcal{L}$. We can interpret $\Pr[\ell \in \cup_{j \in S} A_j]$ as the probability that element ℓ is *covered* by an item in S, where $j \in [m]$ covers $\ell \in \mathcal{L}$ if $\ell \in A_j$. For each $\ell \in \mathcal{L}$, let C_ℓ be the set of items that

cover ℓ. Element $\ell \in \mathcal{L}$ is covered by S precisely when $C_\ell \cap S \neq \emptyset$. Each item $j \in C_\ell$ is included in S independently with probability $1 - e^{-x_j}$. Therefore, the probability $\ell \in \mathcal{L}$ is covered by S can be re-written as follows:

$$\begin{aligned}\Pr[\ell \in \cup_{j \in S} A_j] &= 1 - \prod_{j \in C_\ell} e^{-x_j} \\ &= 1 - \exp\left(-\sum_{j \in C_\ell} x_j\right). \end{aligned} \tag{9}$$

Expression (9) is the composition of the concave function $g(y) = 1 - e^{-y}$ with the affine function $y \to \sum_{j \in C_\ell} x_j$. It is well known that composing a concave function with an affine function yields another concave function (see e.g. [6]). Therefore, $\Pr[\ell \in \cup_{j \in S} A_j]$ is concave in x for each $\ell \in \mathcal{L}$, as needed. This completes the proof.

4.3 Convexity for CGS Valuations

In this section, we will prove Lemma 4.2 in its full generality. First, we define a discrete analogue of a Hessian matrix for set functions, and show that these discrete Hessians are negative semi-definite for CGS functions.

Definition 4.4. *Let $v : 2^{[m]} \to \mathbb{R}$ be a set function. For $S \subseteq [m]$, we define the* discrete Hessian matrix $\mathcal{H}^v_S \in \mathbb{R}^{m \times m}$ *of v at S as follows:*

$$\mathcal{H}^v_S(j,k) = v(S \cup \{j,k\}) - v(S \cup \{j\}) - v(S \cup \{k\}) + v(S) \tag{10}$$

for $j, k \in [m]$.

Naturally, $\mathcal{H}^v_S$ is a symmetric matrix. Moreover, $-\mathcal{H}^v_S(j,k)$ is the deficit in additivity of j and k given the set S, as defined in Equation (3).

Claim 4.5. *If $v : 2^{[m]} \to \mathbb{R}^+$ is a CGS function, then $\mathcal{H}^v_S$ is negative semi-definite for each $S \subseteq [m]$.*

Proof. We observe that $\mathcal{H}^v_S$ is linear in v, and recall that a non-negative weighted sum of negative semi-definite matrices is negative semi-definite. Therefore, it is sufficient to prove this claim when v is a gross substitutes function.

We assume v is in GS, and let $\mathcal{F}^v_S \in \mathbb{R}^{m \times m}$ be as defined in Equation (3). Since $\mathcal{F}^v_S = -\mathcal{H}^v_S$, we prove that $\mathcal{F}^v_S$ is positive semi-definite, using only the characterization of GS functions from Proposition 2.2. We fix v and S, and use $\mathcal{F}$ as shorthand for $\mathcal{F}^v_S$. Let $a_1, \ldots, a_\tau$ be the (distinct) real numbers appearing in $\mathcal{F}$, and assume $a_1 < a_2 < \ldots < a_\tau$. We write $\mathcal{F}$ as a nonnegative weighted sum of $\{0,1\}$ matrices as follows.

$$\mathcal{F} = a_1 \mathcal{F}^1 + \sum_{t=2}^{\tau} (a_t - a_{t-1}) \mathcal{F}^t,$$

where $\mathcal{F}^t \in \{0,1\}^{m \times m}$, and $\mathcal{F}^t_{jk} = 1$ if and only if $\mathcal{F}_{jk} \geq a_t$.

Since a nonnegative weighted sum of positive semi-definite matrices is positive semi-definite, it suffices to show that each $\mathcal{F}^t$ is positive semi-definite. First, note that each matrix $\mathcal{F}^t$ is symmetric

by definition, since $\mathcal{F}$ is symmetric. Proposition 2.2 implies that whenever $\mathcal{F}_{jk} \geq a_t$ and $\mathcal{F}_{k\ell} \geq a_t$, it is the case that $\mathcal{F}_{j\ell} \geq a_t$. Therefore, by definition of $\mathcal{F}^t$, whenever $\mathcal{F}^t_{jk} = 1$ and $\mathcal{F}^t_{k\ell} = 1$, we have $\mathcal{F}_{j\ell} = 1$. Therefore, each $\mathcal{F}^t$ encodes a transitive relation on $[m]$. Moreover, the relation is symmetric since $\mathcal{F}^t$ is symmetric.

A binary matrix encoding a symmetric and transitive relation is a block diagonal matrix where each diagonal block is an all-ones or all-zeros sub-matrix. It is known, and easy to prove, that such a matrix is positive semi-definite. Therefore $\mathcal{F}^v_S$ is positive semi-definite, and $\mathcal{H}^v_S$ is negative semi-definite. □

We now return to Lemma 4.2. Fix n, m, and CGS valuations $\{v_i\}_{i=1}^n$, and let $\mathcal{R}$ denote the feasible set of mathematical program (8). Let $(S_1, \ldots, S_n) \sim r_{\text{poiss}}(x)$ be the (random) allocation computed by the Poisson rounding scheme for point $x \in \mathcal{R}$. The expected welfare $\mathrm{E}[w(r_{\text{poiss}}(x))]$ can be written as $\mathrm{E}[\sum_{i=1}^n v_i(S_i)]$, where the expectation is taken over the internal random coins of the rounding scheme. By linearity of expectation, as well as the fact that the sum of concave functions is concave, it suffices to show that $\mathbf{E}[v_i(S_i)]$ is a concave function of x for an arbitrary player i with CGS valuation v_i.

Fix player i, and use x_j, v, S as short-hand for x_{ij}, v_i, S_i respectively. The Poisson rounding scheme includes each item j in S independently with probability $1 - e^{-x_j}$. We can now write the expected value of player i as the following function $G_v : \mathbb{R}^m \to \mathbb{R}$:

$$G_v(x_1, \ldots, x_m) = \sum_{S \subseteq [m]} v(S) \prod_{j \in S} (1 - e^{-x_j}) \prod_{j \neq S} e^{-x_j} \tag{11}$$

The following claim, combined with Claim 4.5, completes the proof of Lemma 4.2.

Claim 4.6. *If all discrete Hessians of v are negative semi-definite, then G_v is concave.*

Proof. Assume $\mathcal{H}^v_S$ is negative semi-definite for each $S \subseteq [m]$. We work with G_v as expressed in Equation (11). We will show that the Hessian matrix of G_v at an arbitrary $x \in \mathbb{R}^m$ is negative semi-definite, which is a sufficient condition for concavity. We take the mixed-derivative of G_v with respect to x_j and x_k (possibly $j = k$).

$$\begin{aligned}
\frac{\partial^2 G_v(x)}{\partial x_j \partial x_k} &= \sum_{S \subseteq [m] \setminus \{j,k\}} \prod_{\ell \in S} (1 - e^{-x_\ell}) \prod_{\ell \in [m] \setminus S} e^{-x_\ell} \Big(v(S) - v(S \cup \{j\}) - v(S \cup \{k\}) + v(S \cup \{j,k\}) \Big) \\
&= \sum_{S \subseteq [m]} \prod_{\ell \in S} (1 - e^{-x_\ell}) \prod_{\ell \in [m] \setminus S} e^{-x_\ell} \Big(v(S) - v(S \cup \{j\}) - v(S \cup \{k\}) + v(S \cup \{j,k\}) \Big) \\
&= \sum_{S \subseteq [m]} \prod_{\ell \in S} (1 - e^{-x_\ell}) \prod_{\ell \in [m] \setminus S} e^{-x_\ell} \; \mathcal{H}^v_S(j,k)
\end{aligned}$$

The first equality follows by grouping the terms of Equation (11) by the projection of S onto $[m] \setminus \{i,j\}$, and then differentiating. The second equality follows from the fact that $v(S) - v(S \cup \{j\}) - v(S \cup \{k\}) + v(S \cup \{j,k\}) = 0$ when S includes either of j and k. The last equality follows by definition of $\mathcal{H}^v_S$.

The above derivation immediately implies that we can write the Hessian matrix of $G_v(x)$ as a non-negative weighted sum of discrete Hessian matrices.

$$\bigtriangledown^2 G_v(x) = \sum_{S \subseteq [m]} \prod_{\ell \in S} (1 - e^{-x_\ell}) \prod_{\ell \in [m] \setminus S} e^{-x_\ell} \; \mathcal{H}^v_S \tag{12}$$

A non-negative weighted sum of negative semi-definite matrices is negative semi-definite. This completes the proof of the claim. □

5 Combinatorial Public Projects

In this section, we prove our result for combinatorial public projects.

Theorem 5.1. *There is a $(1-1/e)$-approximate, truthful-in-expectation mechanism for combinatorial public projects with CGS valuations in the bounded-lottery-value oracle model, running in expected* $\mathrm{poly}(n,m)$ *time.*

We formulate welfare maximization in combinatorial public projects as an optimization problem as follows. An instance $(\mathcal{S}, w)$ is given by the following integer program with feasible set $\mathcal{S}$ contained in $\{0,1\}^m$. Variable x_j indicates whether project j is selected, and $w(x)$ denotes the social welfare of allocation x.

$$\begin{array}{lll} \text{maximize} & w(x) = \sum_i v_i(\{j : x_j = 1\}) & \\ \text{subject to} & \sum_{j=1}^m x_j \le k & \\ & x_j \in \{0,1\}, & \text{for } j \in [m]. \end{array} \tag{13}$$

We let the relaxed feasible set $\mathcal{R} = \mathcal{R}(\mathcal{S})$ be the result of relaxing the constraints $x_j \in \{0,1\}$ of (7) to $0 \le x_j \le 1$.

We structure the proof of Theorem 5.1 similarly to the proof of Theorem 4.1. We define the *k-bounded-lottery rounding scheme*, which we denote by r_k, in Section 5.1. We prove that r_k is $(1-1/e)$-approximate (Lemma 5.3), and convex (Lemma 5.2). Lemmas 3.1, 3.2 and 5.3, taken together, imply that Algorithm 1, when instantiated for combinatorial public projects with $r = r_k$, is a $(1-1/e)$-approximate MIDR allocation rule. Lemma 5.2 reduces implementing this allocation rule to solving a convex program.

The rest of the proof of Theorem 5.1 proceeds as for Theorem 4.1. Specifically, we handle technical and numerical issues related to solving convex programs in Appendix C, and Proposition B.3 shows that truth-telling VCG payments can be computed in efficiently.

5.1 The k-Bounded-Lottery Rounding Scheme

We now define the *k-bounded-lottery rounding scheme* for combinatorial public projects, which we denote by r_k. When instantiated for combinatorial public projects with $r = r_k$, Algorithm 1 reduces to solving the following optimization problem.

$$\begin{array}{lll} \text{maximize} & \mathrm{E}_{S \sim r_k(x)}[\sum_i v_i(S)] & \\ \text{subject to} & \sum_{j=1}^m x_j \le k & \\ & 0 \le x_j \le 1, & \text{for } j = 1, \ldots, m. \end{array} \tag{14}$$

Given a feasible solution x for mathematical program (14), we let distribution $r_k(x)$ be the k-bounded-lottery with marginals x/k (and promise $\emptyset$), as defined in Section 2.6. We make this more explicit in Algorithm 3.

The k-bounded-lottery rounding scheme is $(1-1/e)$ approximate and convex. We prove the approximation lemma below, and convexity in Section 5.2.

Lemma 5.2. *The k-bounded-lottery rounding scheme is convex for CPP with CGS valuations.*

Algorithm 3 The k-Bounded-Lottery Rounding Scheme r_k

Input: Fractional solution $x \in \mathbb{R}^m$ with $\sum_j x_j \leq k$, and $0 \leq x_j \leq 1$ for all j.
Output: $S \subseteq [m]$ with $|S| \leq k$

1: For each $j \in [m]$ designate the interval $I_j = [\frac{1}{k}\sum_{j'<j} x_{j'}, \frac{1}{k}\sum_{j' \leq j} x_{j'}]$ of length $\frac{x_j}{k}$
2: Draw $p_1, \ldots, p_k$ independently and uniformly from $[0, 1]$
3: Let $S = \{j \in [m] : \{p_1, \ldots, p_k\} \cap I_j \neq \emptyset\}$

Lemma 5.3. *The k-bounded-lottery rounding scheme is $(1-1/e)$-approximate when valuations are submodular.*

Proof. Fix n, m, k and $\{v_i\}_{i=1}^n$. Let $S \subseteq [m]$ be a feasible solution to CPP — i.e. $|S| \leq k$. Let 1_S be the vector with 1 in indices corresponding to S, and 0 otherwise. Let $T \sim r_k(1_S)$. We will first show that each element of $j \in S$ is included in T with probability at least $1-1/e$. Observe that T is the union of k independent draws from a distribution on $[m] \cup \{*\}$, where each time the probability of $j \in S$ is $1/k$. Therefore, the probability that j is included in T is $1-(1-1/k)^k \geq 1-1/e$.

As in the proof of Lemma 4.3, submodularity implies that $\mathrm{E}[v_i(T)] \geq (1-1/e) \cdot v_i(S)$ for each player i. This completes the proof. □

5.2 Convexity for CGS Valuations

In this section, we will prove Lemma 5.2. We note that the special case of Lemma 5.2 for coverage valuations admits a simpler proof similar to that in Section 4.2, with the the function $1-(1-\frac{y}{k})^k$ playing the role of the function $1-e^{-y}$.

Fix n and m. For each cardinality bound $k \in [m]$, let $\mathcal{P}_k$ denote the polytope of fractional solutions to CPP as given in (14). For a set of CGS valuations $v_1, \ldots, v_n$, we observe that the social welfare $v(S) = \sum_{i=1}^n v_i(S)$ is — by the (obvious) fact that the sum of CGS functions is a CGS function — also a CGS function. Therefore, we will prove Lemma 5.2 by showing that, for each $k \in [m]$ and CGS function $v : 2^{[m]} \to \mathbb{R}$, the following function of $x \in \mathcal{P}_k$ is concave in x.

$$\begin{aligned} G_k^v(x) &= \mathop{\mathrm{E}}_{S \sim r_k(x)}[v(S)] \\ &= \sum_{S \subseteq [m]} v(S) \Pr[r_k(x) = S] \end{aligned} \tag{15}$$

We use techniques from combinatorics to write $\Pr[r_k(x) = S]$ in a form that will be easier to work with. For $T \subseteq [m]$, we use x_T as short-hand for $\sum_{j \in T} x_j$, and $\overline{T}$ as short-hand for $[m] \setminus T$.

Claim 5.4. *For each $k \in [m]$, $x \in \mathcal{P}_k$, and $S \subseteq [m]$*

$$\Pr[r_k(x) = S] = -1^{|S|} \sum_{R \subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^k \tag{16}$$

Proof. First, we observe that $\Pr[r_k(x) \subseteq R]$ can be expressed as a simple closed form in x for every set $R \subseteq [m]$. Let $p_1, \ldots, p_k$ and $I_1, \ldots, I_m$ be as in Algorithm 3. The event $r_k(x) \subseteq R$ occurs exactly when none of $p_1, \ldots, p_k$ land in the intervals corresponding to projects $\overline{R}$. Recalling that

the interval I_j of project j has length x_j/k, we get that the probability of any particular p_t falling in $\cup_{j\in\overline{R}} I_j$ is exactly $x_{\overline{R}}/k$. Therefore, by the independence of the variables $p_1, \ldots, p_k$, we get that

$$\Pr[r_k(x) \subseteq R] = \left(1 - \frac{x_{\overline{R}}}{k}\right)^k \tag{17}$$

Next, we express $\Pr[r_k(x) = S]$ in terms of the expressions $\Pr[r_k(x) \subseteq R]$ for sets $R \subseteq S$. It is easy to see that $\Pr[r_k(x) = S]$ is equal to:

$$\Pr[r_k(x) \subseteq S] - \Pr[\bigvee_{j\in S} r_k(x) \subseteq S \setminus \{j\}] \tag{18}$$

Using the inclusion-exclusion principle, we can rewrite (18) as follows:

$$\Pr[r_k(x) \subseteq S] - \sum_{\emptyset \neq T \subseteq S} -1^{|T|-1} \Pr[r_k(x) \subseteq S \setminus T] \tag{19}$$

Letting $R = S \setminus T$ in (19), we get

$$\Pr[r_k(x) \subseteq S] - \sum_{R \subsetneq S} -1^{|S|-|R|-1} \Pr[r_k(x) \subseteq R] \tag{20}$$

We can easily simplify (20) to conclude that

$$\Pr[r_k(x) = S] = \sum_{R \subseteq S} -1^{|S|-|R|} \Pr[r_k(x) \subseteq R] \tag{21}$$

Combining (21) and (17) completes the proof. □

Recall the *discrete Hessian* matrices from Definition 4.4. Building on Claim 5.4, we now express the Hessian matrix of G_k^v as a non-negative weighted sum of discrete Hessian matrices of v. We note that when $x \in \mathcal{P}_k$, it is easy to verify that $\frac{k-2}{k} \cdot x \in \mathcal{P}_{k-2}$, and therefore (22) is well defined.

Claim 5.5. *For each $k \in [m]$, $x \in \mathcal{P}_k$, and $v : 2^{[m]} \to \mathbb{R}$, we have*

$$\bigtriangledown^2 G_k^v(x) = \frac{k-1}{k} \sum_{S\subseteq[m]} \Pr\left[r_{k-2}\left(\frac{k-2}{k} \cdot x\right) = S\right] \mathcal{H}_S^v \tag{22}$$

where $\mathcal{H}_S^v$ is as in Definition 4.4.

Proof. Fix $i, j \in [m]$, possibly with $i = j$. We work with G_k^v as defined in Equation (15), and plug in expression (16).

$$G_k^v(x) = \sum_{S\subseteq[m]} v(S) \cdot -1^{|S|} \sum_{R\subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^k$$

Differentiating with respect to x_i and x_j gives:

$$\frac{\partial^2 G_k^v(x)}{\partial x_i \partial x_j} = \frac{k-1}{k} \sum_{S\subseteq[m]} v(S) \cdot -1^{|S|} \sum_{R\subseteq S\setminus\{i,j\}} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-2}$$

We group the terms by projecting S onto $[m] \setminus \{i, j\}$, and then we simplify the resulting expression.

$$\begin{aligned}\frac{\partial^2 G_k^v(x)}{\partial x_i \partial x_j} =& \frac{k-1}{k} \sum_{S \subseteq [m] \setminus \{i,j\}} -1^{|S|} \sum_{R \subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-2} \left(v(S) - v(S \cup \{i\}) - v(S \cup \{j\}) + v(S \cup \{i,j\})\right) \\ =& \frac{k-1}{k} \sum_{S \subseteq [m]} -1^{|S|} \sum_{R \subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-2} \left(v(S) - v(S \cup \{i\}) - v(S \cup \{j\}) + v(S \cup \{i,j\})\right) \\ =& \frac{k-1}{k} \sum_{S \subseteq [m]} -1^{|S|} \sum_{R \subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-2} \mathcal{H}_S^v(i,j) \end{aligned} \tag{23}$$

The second equality follows from the fact that $v(S) - v(S \cup \{i\}) - v(S \cup \{j\}) + v(S \cup \{i,j\}) = 0$ when S includes either of i and j. The last equality follows by definition of $\mathcal{H}_S^v$.

Invoking Claim 5.4 with $k' = k-2$ and $x' = \frac{k-2}{k} \cdot x$, and plugging the resulting expression into into (23), we conclude that

$$\frac{\partial^2 G_k^v(x)}{\partial x_i \partial x_j} = \frac{k-1}{k} \sum_{S \subseteq [m]} \Pr\left[r_{k-2}\left(\frac{k-2}{k} \cdot x\right) = S\right] \mathcal{H}_S^v(i,j).$$

□

Claims 4.5 and 5.5 establish that, when v is in CGS and $k \in [m]$, $\bigtriangledown^2 G_k^v(x)$ is a non-negative weighted sum of negative semi-definite matrices for each $x \in \mathcal{P}_k$. A non-negative weighted sum of negative semi-definite matrices is negative semi-definite. Therefore, the Hessian matrix of G_k^v is negative semi-definite at each $x \in \mathcal{P}_k$, and we conclude that G_k^v is a concave function on $\mathcal{P}_k$. This completes the proof of Lemma 5.2.

6 Conclusions and Open Problems

In this paper we introduced the convex rounding mechanism design framework, and used it to design computationally tractable truthful-in-expectation approximation mechanisms for welfare maximization in combinatorial auctions and combinatorial public projects. Our guarantees apply to bidders with valuations that are in the cone generated by gross substitutes valuations, are the best possible assuming $P \neq NP$, and are the first such guarantees for natural NP-hard variants of these problems.

Since initial publication of the work in this paper, the convex rounding framework has been applied to other problems in mechanism design in at least one instance ([33]), and we anticipate more. In particular, an attractive potential application is to combinatorial auctions with valuations exhibiting limited *complementarity*, as defined in [1] and [27]. More generally, a characterization of the family of problems and linear programming relaxations which admit optimal (or near optimal) convex rounding algorithms would open the door to understanding the power and limitations of our technique.

Other open questions and potential extensions of our results remain. Can our mechanisms be "de-randomized", i.e. converted to deterministic or at least universally-truthful mechanisms attaining the same guarantees for CGS valuations? Can our lottery oracle models be relaxed to the traditional value oracle model, without sacrificing any of our guarantees? What about extensions to other natural classes of valuations, such as budgeted additive valuations?

Finally, our result for combinatorial auctions can be thought of as extending the tractability of truthful combinatorial auctions from gross substitutes valuations to CGS valuations at the cost of a constant factor loss in the approximation ratio. More generally, noting that gross substitutes valuations have many attractive economic properties (e.g. permitting Walrasian equilibria in markets with discrete goods), do these properties admit approximate generalizations to CGS valuations? A more thorough understanding the cone of gross substitutes valuations, and its economic and algorithmic implications, appears worthy of pursuit.

Acknowledgments

We thank Ittai Abraham, Moshe Babaioff, Bobby Kleinberg, and Jan Vondrák for helpful discussions and comments. Specifically, we thank Bobby Kleinberg for insights into the algebraic properties of set functions that were useful in guiding the early stages of this work, and we thank Jan Vondrák for pointing out that the proof of Lemma 4.2 can be simplified in the special case of coverage functions (Section 4.2). Finally, we thank the anonymous STOC and EC referees for helpful suggestions.

References

[1] Ittai Abraham, Moshe Babaioff, Shaddin Dughmi, and Tim Roughgarden, *Combinatorial auctions with restricted complements*, Proceedings of the 13th ACM Conference on Electronic Commerce (EC), 2012, pp. 3–16.

[2] Xiaohui Bei and Zhiyi Huang, *Towards optimal Bayesian algorithmic mechanism design*, Proceedings of the 22nd ACM Symposium on Discrete Algorithms (SODA), 2011, pp. 720–733.

[3] Aharon Ben-Tal and Arkadi Nemirovski, *Lectures on modern convex optimization: Analysis, algorithms, and engineering applications*, SIAM, 2001.

[4] Meir Bing, Daniel J. Lehmann, and Paul Milgrom, *Presentation and structure of substitutes valuations*, Proceedings of the 5th ACM Conference on Electronic Commerce (EC), 2004, pp. 238–239.

[5] Liad Blumrosen and Noam Nisan, *Combinatorial auctions (a survey)*, Algorithmic Game Theory (Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay Vazirani, eds.), Cambridge University Press, 2007.

[6] Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge University Press, 2004.

[7] Dave Buchfuhrer, Shaddin Dughmi, Hu Fu, Robert Kleinberg, Elchanan Mossel, Christos Papadimitriou, Michael Schapira, Yaron Singer, and Chris Umans, *Inapproximability for VCG-based combinatorial auctions*, Proceedings of the 21st ACM Symposium on Discrete Algorithms (SODA), 2010, pp. 518–536.

[8] David Buchfuhrer, Michael Schapira, and Yaron Singer, *Computation and incentives in combinatorial public projects*, Proceedings of the 11th ACM Conference on Electronic Commerce (EC), 2010, pp. 33–42.

[9] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák, *Maximizing a submodular set function subject to a matroid constraint*, Proceedings of the 12th International Conference on Integer Programming and Combinatorial Optimization (IPCO), 2007, pp. 182–196.

[10] Peter Cramton, Yoav Shoham, and Richard Steinberg (eds.), *Combinatorial auctions*, MIT Press., 2006.

[11] Shahar Dobzinski, *Two randomized mechanisms for combinatorial auctions*, Proceedings of the 10th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX) (2007), 89–103.

[12] Shahar Dobzinski, *An impossibility result for truthful combinatorial auctions with submodular valuations*, Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC), 2011, pp. 139–148.

[13] Shahar Dobzinski and Shaddin Dughmi, *On the power of randomization in algorithmic mechanism design*, Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS), 2009, pp. 505–514.

[14] Shahar Dobzinski, Hu Fu, and Robert Kleinberg, *Truthfulness via proxies*, arXiv:1011.3232 (2010).

[15] Shahar Dobzinski and Noam Nisan, *Limitations of VCG-based mechanisms*, Proceedings of the 38th ACM Symposium on Theory of Computing (STOC), 2007, pp. 338–344.

[16] Shahar Dobzinski, Noam Nisan, and Michael Schapira, *Approximation algorithms for combinatorial auctions with complement-free bidders*, Proceedings of the 36th ACM Symposium on Theory of Computing (STOC), 2005, pp. 610–618.

[17] ———, *Truthful randomized mechanisms for combinatorial auctions*, Proceedings of the 37th ACM Symposium on Theory of Computing (STOC), 2006, pp. 644–652.

[18] Shahar Dobzinski and Michael Schapira, *An improved approximation algorithm for combinatorial auctions with submodular bidders*, Proceedings of the 17th ACM Symposium on Discrete Algorithms (SODA), 2006, pp. 1064–1073.

[19] Shahar Dobzinski and Mukund Sundararajan, *On characterizations of truthful mechanisms for combinatorial auctions and scheduling*, Proceedings of the 9th ACM Conference on Electronic Commerce (EC), 2008.

[20] Shahar Dobzinski and Jan Vondrák, *The computational complexity of truthfulness in combinatorial auctions*, Proceedings of the 13th ACM Conference on Electronic Commerce (EC), 2012, pp. 405–422.

[21] Shaddin Dughmi, *A truthful randomized mechanism for combinatorial public projects via convex optimization*, Proceedings of the 12th ACM Conference on Electronic Commerce (EC), 2011, pp. 263–272.

[22] Shaddin Dughmi and Tim Roughgarden, *Black-box randomized reductions in algorithmic mechanism design*, Proceedings of the 51st IEEE Symposium on Foundations of Computer Science (FOCS), 2010, pp. 775–784.

[23] Shaddin Dughmi, Tim Roughgarden, Jan Vondrák, and Qiqi Yan, *An approximately truthful-in-expectation mechanism for combinatorial auctions using value queries*, CoRR **abs/1109.1053** (2011).

[24] Shaddin Dughmi, Tim Roughgarden, and Qiqi Yan, *From convex optimization to randomized mechanisms: toward optimal combinatorial auctions*, Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC), 2011, pp. 149–158.

[25] Shaddin Dughmi and Jan Vondrák, *Limitations of randomized mechanisms for combinatorial auctions*, Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science (FOCS), 2011.

[26] Uriel Feige, *On maximizing welfare where the utility functions are subadditive*, Proceedings of the 37th ACM Symposium on Theory of Computing (STOC), 2006, pp. 122–142.

[27] Uriel Feige and Rani Izsak, *Welfare maximization and the supermodular degree*, Proceedings of the 4th Conference on Innovations in Theoretical Computer Science, 2013, pp. 247–256.

[28] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák, *Maximizing non-monotone submodular functions*, Proceedings of the 48th IEEE Symposium on Foundations of Computer Science (FOCS), 2007, pp. 461–471.

[29] Uriel Feige and Jan Vondrák, *Approximation algorithms for allocation problems: Improving the factor of 1-1/e*, Proceedings of the 47th IEEE Symposium on Foundations of Computer Science (FOCS), 2006.

[30] Faruk Gul and Ennio Stacchetti, *Walrasian equilibrium with gross substitutes*, Journal of Economic Theory **87** (1999), 95 – 124.

[31] Jason Hartline, Robert Kleinberg, and Azarakhsh Malekian, *Multi-parameter Bayesian algorithmic mechanism design*, Proceedings of the 22nd ACM Symposium on Discrete Algorithms (SODA), 2011, pp. 734–747.

[32] Jason D. Hartline and Brendan Lucier, *Bayesian algorithmic mechanism design*, Proceedings of the 41st ACM Symposium on Theory of Computing (STOC), 2010, pp. 301–310.

[33] Martin Hoefer and Thomas Kesselheim, *Secondary spectrum auctions for symmetric and submodular bidders*, Proceedings of the 13th ACM Conference on Electronic Commerce, ACM, 2012, pp. 657–671.

[34] A. S. Kelso, Jr. and V. P. Crawford, *Job matching, coalition formation, and gross substitutes*, Econometrica **50** (1982), no. 6.

[35] Subhash Khot, Richard J. Lipton, Evangelos Markakis, and Aranyak Mehta, *Inapproximability results for combinatorial auctions with submodular utility functions*, Algorithmica **52** (2008), no. 1, 3–18.

[36] Ron Lavi and Chaitanya Swamy, *Truthful and near-optimal mechanism design via linear programming*, Proceedings of the 46th IEEE Symposium on Foundations of Computer Science (FOCS), 2005, pp. 595–604.

[37] B. Lehmann, D. Lehmann, and N. Nisan, *Combinatorial auctions with decreasing marginal utilities*, Games and Economic Behavior **55** (2006), no. 2, 270–296.

[38] Daniel Lehmann, Liadan Ita Oćallaghan, and Yoav Shoham, *Truth revelation in approximately efficient combinatorial auctions*, Journal of the ACM (JACM) **49** (2002), no. 5, 577–602.

[39] Vahab Mirrokni, Michael Schapira, and Jan Vondrák, *Tight information-theoretic lower bounds for welfare maximization in combinatorial auctions*, Proceedings of the 9th ACM Conference on Electronic Commerce (EC), 2008.

[40] K. Murota, *Valuated matroid intersection II: Algorithms*, SIAM Journal on Discrete Mathematics **9** (1996), no. 4, 562–576.

[41] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, *An analysis of approximations for maximizing submodular set functions – I.*, Mathematical Programming **14** (1978), no. 3, 265–294.

[42] Noam Nisan, *Introduction to mechanism design (for computer scientists)*, Algorithmic Game Theory (Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay Vazirani, eds.), Cambridge University Press, 2007.

[43] Noam Nisan and Amir Ronen, *Algorithmic mechanism design*, Games and Economic Behaviour **35** (2001), 166 – 196.

[44] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani, *Algorithmic game theory*, Cambridge University Press, New York, NY, USA, 2007.

[45] J. G. Oxley, *Matroid theory*, Oxford University Press, 1992.

[46] R. Paes Leme, *Gross substitutability: an algorithmic survey*, Working paper, 2013.

[47] Christos Papadimitriou, Michael Schapira, and Yaron Singer, *On the hardness of being truthful*, Proceedings of the 49th IEEE Symposium on Foundations of Computer Science (FOCS), 2008, pp. 250–259.

[48] Ran Raz and Shmuel Safra, *A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP*, Proceedings of the 29th ACM Symposium on Theory of Computing (STOC), 1997, pp. 475–484.

[49] Alexander Schrijver, *Combinatorial optimization*, Springer, 2003.

[50] Jan Vondrák, *Optimal approximation for the submodular welfare problem in the value oracle model*, Proceedings of the 39th ACM Symposium on Theory of Computing (STOC), 2008, pp. 67–74.

A Additional Preliminaries

A.1 Matroids and Related Set Functions

In this section, we review some basics of matroid theory, and related CGS functions. For a more comprehensive reference on matroid theory we refer the reader to [45]. For their connection to gross substitutes, we refer the reader to the survey in [46].

A *matroid* M is a pair $(\mathcal{X}, \mathcal{I})$, where $\mathcal{X}$ is a finite *ground set*, and $\mathcal{I}$ is a non-empty family of subsets of $\mathcal{X}$ satisfying the following two properties. (1) *Downward closure:* If S belongs to $\mathcal{I}$, then so do all subsets of S. (2) *The Exchange Property:* Whenever $T, S \in \mathcal{I}$ with $|T| < |S|$, there is some $x \in S \setminus T$ such that $T \cup \{x\} \in \mathcal{I}$. Elements of $\mathcal{I}$ are often referred to as the *independent sets* of the matroid. Subsets of $\mathcal{X}$ that are not in $\mathcal{I}$ are often called *dependent*.

We associate with matroid M a set function $rank_M : 2^{\mathcal{X}} \to \mathbb{N}$, known as the *rank function of* M, defined as follows: $rank_M(A) = \max_{S \in \mathcal{I}} |S \cap A|$. Equivalently, the rank of set A in matroid M is the maximum size of an independent set contained in A. More generally, given a matroid $M = (\mathcal{X}, \mathcal{I})$ and a weight vector $w \in \mathbb{R}^{\mathcal{X}}$, the associated *weighted rank* function is defined as follows: $rank^w_M(A) = \max_{S \in \mathcal{I}} \sum_{i \in S \cap A} w_i$. A set function f on a ground set $\mathcal{X}$ is a *matroid rank function* if there exists a matroid M on the same ground set such that $f = rank_M$, and is a *matroid weighted rank function* if there exists M and w such that $f = rank^w_M$. It is known, and easy to check, that a matroid weighted rank function can be expressed as a nonnegative weighted sum of (unweighted) matroid rank functions. Both weighted and unweighted matroid rank functions are monotone ($f(S) \leq f(T)$ when $S \subseteq T$), normalized ($f(\emptyset) = 0$), and submodular ($f(S) + f(T) \geq f(S \cap T) + f(S \cup T)$ for all S and T). In fact, they are in GS (see [46]).

Recall that a a *coverage function* f on ground set $[m]$ designates some set $\mathcal{L}$ of elements, and m subsets $A_1, \ldots, A_m \subseteq \mathcal{L}$, such that $f(S) = |\cup_{j \in S} A_j|$. We note that coverage functions can be expressed as a sum of matroid rank functions. In particular, for each $\ell \in \mathcal{L}$, let $f_\ell(S) = 1$ if $\ell \in \cup_{j \in S} A_S$, and $f_\ell(S) = 0$ otherwise. It is easy to check that f_ℓ is a matroid rank function, and moreover $f(S) = \sum_{\ell \in L} f_\ell(S)$ for all $S \subseteq [m]$. As a consequence, coverage functions are in CGS.

A.2 Convex Optimization

In this section, we distill some basics of convex optimization. For more details, see [3].

Definition A.1. *A maximization problem is given by a set Π of instances $(\mathcal{P}, c)$, where $\mathcal{P}$ is a subset of some euclidean space, $c : \mathcal{P} \to \mathbb{R}$, and the goal is to maximize $c(x)$ over $x \in \mathcal{P}$. We say Π is a convex maximization problem if for every $(\mathcal{P}, c) \in \Pi$, $\mathcal{P}$ is a compact convex set, and $c : \mathcal{P} \to \mathbb{R}$ is concave. If $c : \mathcal{P} \to \mathbb{R}^+$ for every instance of Π, we say Π is non-negative.*

Definition A.2. *We say a non-negative maximization problem Π is* R-solvable *in polynomial time if there is an algorithm that takes as input the representation of an instance $\mathcal{I} = (\mathcal{P}, c) \in \Pi$ — where we use $|\mathcal{I}|$ to denote the number of bits in the representation — and an approximation parameter ϵ, and in time $\text{poly}(|\mathcal{I}|, \log(1/\epsilon))$ outputs $x \in \mathcal{P}$ such that $c(x) \geq (1 - \epsilon) \max_{y \in \mathcal{P}} c(y)$.*

Fact A.3. *Consider a non-negative convex maximization problem Π. If the following are satisfied, then Π is R-solvable in polynomial time using the ellipsoid method. We let $\mathcal{I} = (\mathcal{P}, c)$ denote an instance of Π, and let m denote the dimension of the ambient euclidean space.*

1. *Polynomial Dimension: m is polynomial in $|\mathcal{I}|$.*

2. *Starting ellipsoid: There is an algorithm that computes, in time* $\text{poly}(|\mathcal{I}|)$*, a point* $c \in \mathbb{R}^m$*, a matrix* $A \in \mathbb{R}^{m \times m}$*, and a number* $\mathcal{V} \in \mathbb{R}$ *such that the following hold. We use* $E(c, A)$ *to denote the ellipsoid given by center* c *and linear transformation* A*.*

   (a) $E(c, A) \supseteq \mathcal{P}$

   (b) $\mathcal{V} \leq volume(\mathcal{P})$

   (c) $\frac{volume(E(c,A))}{\mathcal{V}} \leq 2^{\text{poly}(|\mathcal{I}|)}$

3. *Separation oracle for* $\mathcal{P}$*: There is an algorithm that takes takes input* $\mathcal{I}$ *and* $x \in \mathbb{R}^m$*, and in time* $\text{poly}(|\mathcal{I}|, |x|)$ *where* $|x|$ *denotes the size of the representation of* x*, outputs "yes" if* $x \in \mathcal{P}$*, otherwise outputs* $h \in \mathbb{R}^m$ *such that* $h^T x < h^T y$ *for every* $y \in \mathcal{P}$*.*

4. *First order oracle for* c*: There is an algorithm that takes input* $\mathcal{I}$ *and* $x \in \mathbb{R}^m$*, and in time* $\text{poly}(|\mathcal{I}|, |x|)$ *outputs* $c(x) \in \mathbb{R}$ *and* $\bigtriangledown c(x) \in \mathbb{R}^m$*.*

B Additional Technical Details and Commentary

B.1 Relaxing the Oracle Model

Both of our mechanisms assumed the existence of a "lottery value" oracle for evaluating a player's expected value on a distribution. As argued in [23], these oracles can not always be implemented efficiently — in fact, doing so is #P-hard for some matroid rank functions. Nevertheless, we justify employing these oracles on two grounds: First, our lottery-value oracles are easily implemented for the most interesting example of CGS valuations, specifically explicit coverage functions. Second, our lottery-value oracles can be approximated arbitrarily well with high probability using a polynomial number of value oracle queries (see [50]). Even though we are not able to reconcile the incurred sampling errors — small as they may be — with the requirement that our mechanisms be *exactly* truthful, it was shown in in [23] that relaxing our solution concept to approximate truthfulness – also known as ϵ-truthfulness – removes this difficulty for combinatorial auctions, allowing us to relax our oracle model to the more traditional value oracles. Next, we show that our oracles can be implemented efficiently for explicit coverage functions.

Coverage Valuations

An *explicit coverage valuation* v is represented as follows. There is a finite set $\mathcal{L}$, and a family $A_1, \ldots, A_m$ of subsets of $\mathcal{L}$. The valuation v is defined by $v_i(S) = |\cup_{j \in S} A_j|$. The set system $\left(\mathcal{L}, \{A_j\}_{j=1}^m\right)$ is encoded explicitly as a bipartite graph. Next, we show that both our lottery oracles can be implemented in time polynomial in the size of the representation of such a valuation, implying that the mechanisms of Theorems 4.1 and 5.1 can be implemented in polynomial time when players have explicit coverage valuations.

Claim B.1. *Let* v *be a coverage valuation represented explicitly. Product-lottery-value queries on* v *can be answered in time polynomial in the size of the representation of* v*.*

Proof. Let $(\mathcal{L}, \{A_j\}_{j=1}^m)$ be the set system defining v, and let $x \in [0, 1]^m$. Let S be a random set that includes each $j \in [m]$ independently with probability x_j. The outcome of the product-lottery-value oracle of v evaluated at x is equal to the sum, over all $\ell \in \mathcal{L}$, of the probability that ℓ is

"covered" by S – specifically, $\sum_{\ell \in \mathcal{L}} \Pr[\ell \in \cup_{j \in S} A_j]$. It is easy to verify that a term of this sum can be expressed as the following closed form expression.

$$\Pr[\ell \in \cup_{j \in S} A_j] = 1 - \prod_{j : A_j \ni \ell} (1 - x_j)$$

It is simple to check this expression can be evaluated in time polynomial in the representation of the set system. This completes the proof. □

Claim B.2. *Let v be a coverage valuation represented explicitly. Bounded-lottery-value queries on v can be answered in time polynomial in the size of the representation of v.*

Proof. Let $(\mathcal{L}, \{A_j\}_{j=1}^m)$ be the set system defining v. Consider $R \subseteq [m]$, $k \in [m]$, and $x \in [0,1]^m$ with $\sum_{i=1}^m x_i \leq 1$. For $\ell \in \mathcal{L}$, let $T_\ell = \{j \in [m] : \ell \in A_j\}$ be the set of projects that "cover" ℓ. The output of the k-bounded-lottery-value oracle on marginals x and promise R can be written as follows:

$$v(R) + \sum_{\ell \in \mathcal{L} \setminus \cup_{j \in R} A_j} \left(1 - \left(1 - \sum_{j \in T_\ell} x_j \right)^k \right).$$

It is simple to check that this expression can be evaluated in time polynomial in the representation of the set system. This completes the proof. □

B.2 Computing Payments

In this section, we show how to efficiently compute truth-telling payments for our mechanisms. In fact, as shown below, this is possible for any maximal in distributional range allocation rule for combinatorial auctions or public projects, given as a black box.

Proposition B.3. *Let $\mathcal{A}$ be an MIDR allocation rule for either combinatorial auctions or combinatorial public projects. Let $\mathcal{R}$ denote the distributional range of $\mathcal{A}$. There exists a randomized payment rule p, implementable in* $\text{poly}(n)$ *time given black-box access to $\mathcal{A}$, such that the resulting mechanism $(\mathcal{A}, p)$ is truthful in expectation, individually rational in expectation, and its payments are non-negative in expectation.*

Proof. This theorem applies to any mechanism design problem where players have non-negative valuations, and where the zero valuation (i.e. the valuation assigning 0 to each outcome) is a feasible report for each player. Both properties hold for combinatorial auctions and combinatorial public projects.

Let p^{vcg} denote the (deterministic) VCG payment scheme for the problem of expected welfare maximization over $\mathcal{R}$. We can write p^{vcg} as follows.

$$p_i^{vcg}(v_1, \ldots, v_n) = \mathop{\mathrm{E}}_{T \sim \mathcal{A}(\mathbf{0}, v_{-i})} \left[\sum_{j \neq i} v_j(T) \right] - \mathop{\mathrm{E}}_{S \sim \mathcal{A}(v_1, \ldots, v_n)} \left[\sum_{j \neq i} v_j(S) \right]. \tag{24}$$

Combining $\mathcal{A}$ with p^{vcg} yields a mechanism that is in expectation truthful, individually rational, and has non-negative payments. However, payment scheme p^{vcg} may not be implementable efficiently, due to the difficulty in exactly evaluating the expectations in expression (24). Nevertheless,

it is an easy observation that any payment scheme p with $\mathbf{E}[p_i(v)] = p_i^{vcg}(v)$ for each i and v yields the same guarantees in expectation.

To complete the proof, we show how to compute such a payment rule p using only a polynomial number of invocations of $\mathcal{A}$ as a black box. We sample random variable $p_i(v)$ as follows: Let T be a sample from lottery $\mathcal{A}(v)$, let T_{-i} be a sample from lottery $\mathcal{A}(v_{-i}, 0)$, and let $p_i(v) = \sum_{j \neq i} v_j(T_{-i}) - \sum_{j \neq i} v_j(T)$. Using Equation (24) and the fact that $\mathcal{A}$ is MIDR with range $\mathcal{R}$, we conclude that $\mathbf{E}[p_i(v)] = p_i^{vcg}(v)$ for each i and v. This completes the proof. □

We note that the mechanism resulting from Proposition B.3 is individually rational in expectation, and each payment is non-negative in expectation. We leave open the question of whether it is possible to enforce individual rationality and non-negative payments for our mechanism ex-post.

B.3 Beyond CGS Valuations

In this section, we discuss the prospect of extending our results beyond CGS valuations. First, we argue that our restriction to a subset of submodular functions is not merely an artifact of our analysis. Specifically, we exhibit a submodular function that is not in the CGS family, and moreover both the Poisson and k-bounded-lottery rounding schemes can be non-convex when a player has this function as their valuation. Then, we briefly argue that our mechanism may yet apply to some valuations that are not in CGS. Finally, we discuss recent results that rule out extending our results to all submodular functions.

We define a budget additive (and therefore submodular) function v on four items $\{1, 2, 3, 4\}$. Three of the items are "small", one item is "big", and the budget equals the value of the big item.

$$v(S) = \begin{cases} 1 & \text{if } S = \{j\} \text{ for } j \in \{1, 2, 3\}, \\ 2 & \text{if } S = \{4\}, \\ \min\left(\sum_{j \in S} v(\{j\}), 2\right) & \text{otherwise} \end{cases}$$

We can show that v is not a CGS function by invoking Claim 4.5. Specifically, one can manually check that the discrete Hessian matrix $\mathcal{H}_{\emptyset}^{v}$ of v at $\emptyset$ (see Definition 4.4) is not negative semi-definite. Moreover, for a player with valuation v, Poisson rounding renders the player's expected value function $G_v(x)$ (Equation (11)) non-concave in x: By Equation (12), the Hessian matrix of $G_v(x)$ approaches the discrete Hessian $\mathcal{H}_{\emptyset}^{v}$ as x tends to zero. Since $\mathcal{H}_{\emptyset}^{v}$ is not negative semi-definite, $G_v(x)$ is non-concave for x near zero. We note that we can construct a large family of similar counter examples by simply increasing the number of "small items" in v. Similar difficulties hold for the k-bounded-lottery rounding scheme as well.

We observe that our mechanisms may yet apply to some valuations that are not in CGS. Our results only used two properties of CGS functions: their discrete Hessian matrices are negative semi-definite (Claim 4.5, which is used to prove Lemmas 4.2 and 5.2), and they are submodular (used to prove Lemmas 4.3 and 5.3). Therefore, our results extends directly to the class of all set functions satisfying both of these properties. We leave open the question of whether there exist interesting functions in this class that are not in CGS. More generally, understanding the class of set functions with negative semi-definite discrete Hessian matrices — in particular the relationship of this class to other classes of set functions studied in the literature — may be an interesting direction for future inquiry.

Finally, as noted in the introduction, recent impossibility results [25, 20] have ruled out polynomial-time and constant-approximate truthful mechanisms for combinatorial auctions and public projects with general submodular valuations. The proofs in [25, 20] exploit *double-peaked* submodular valuations, which are evidently not in CGS and do not admit negative semi-definite discrete Hessians.

C Solving The Convex Programs

In this section, we overcome some technical difficulties related to the solvability of convex programs for both combinatorial auctions and public projects. Since both combinatorial auctions and public projects are similar in this respect, we first present these ideas in the context of combinatorial auctions, and later discuss the necessary modifications for combinatorial public projects.

C.1 Combinatorial Auctions

We show in Section C.1.1 that, in the product-lottery-value oracle model, the four conditions for "solvability" of convex programs, as stated in Fact A.3, are easily satisfied for convex program (8). However, an additional challenge remains: "solving" a convex program – as in Definition A.2 – returns an approximately optimal solution. Indeed the optimal solution of a convex program may be irrational in general, so this is unavoidable.

We show how to overcome this difficulty if we settle for polynomial runtime in expectation. While the optimal solution x^* of (8) cannot be computed explicitly, the random variable $r_{\text{poiss}}(x^*)$ can be sampled in expected polynomial-time. The key idea is the following: *sampling the random variable $r_{\text{poiss}}(x^*)$ rarely requires precise knowledge of x^**. Depending on the coin flips of r_{poiss}, we decide how accurately we need to solve convex program (8) in order compute $r_{\text{poiss}}(x^*)$. Roughly speaking, we show that the probability of requiring a $(1-\epsilon)$-approximation falls exponentially in $\frac{1}{\epsilon}$. As a result, we can sample $r_{\text{poiss}}(x^*)$ in expected polynomial-time. We implement this plan in Section C.1.2 under the simplifying assumption that convex program (8) is *well-conditioned* – i.e. is "sufficiently concave" everywhere. In Section C.1.3, we show how to remove that assumption by slightly modifying our algorithm.

C.1.1 Approximating the Convex Program

Claim C.1. *There is an algorithm for Combinatorial Auctions with CGS valuations in the product-lottery-value oracle model that takes as input an instance of the problem and an approximation parameter $\epsilon > 0$, runs in $\text{poly}(n, m, \log(1/\epsilon))$ time, and returns a $(1-\epsilon)$-approximate solution to convex program (8).*

It suffices to show that the four conditions of Fact A.3 are satisfied in our setting. The first three are immediate from elementary combinatorial optimization (see for example [49]). It remains to show that the first-order oracle, as defined in Fact A.3, can be implemented in polynomial-time in the product-lottery-value oracle model. The objective $f(x)$ of convex program (8) can, by definition, be written as

$$f(x) = \sum_i G_{v_i}(x_i),$$

where v_i is the valuation function of player i, x_i is the vector $(x_{i1}, \ldots, x_{im})$, and and G_{v_i} is as defined in (11). By definition, $G_{v_i}(x_i)$ is the outcome of querying the product-lottery-value oracle

of player i with $(1-e^{-x_{i1}},\ldots,1-e^{-x_{im}})$. Therefore, we can evaluate $f(x)$ using n product-lottery-value query, one for each player. It remains to show that we can also evaluate the (multi-variate) derivative $\bigtriangledown f(x)$ of $f(x)$. Using definition (11), we take the partial derivative corresponding to x_{ij}. By rearranging the sum appropriately, we get that

$$\frac{\partial f}{\partial x_{ij}}(x) = e^{-x_{ij}}\left(F_{v_i}\big((1-e^{-x_{i1}},\ldots,1-e^{-x_{im}})\vee 1_j\big) - F_{v_i}\big((1-e^{-x_{i1}},\ldots,1-e^{-x_{im}})\wedge 0_j\big)\right),$$

where F_{v_i} is as defined in Equation (4). Here, $\vee$ and $\wedge$ denote entry-wise minimum and maximum respectively, 1_j denotes the vector with all entries equal to 0 except for a 1 at position j, and 0_j denotes the vector with all entries equal to 1 except for a 0 at position j. It is clear that this entry of the gradient of f can be evaluated using two product-lottery-value queries. Therefore, $\bigtriangledown f(x)$ can be evaluated using $2n$ product-lottery-value queries, 2 for each player. This completes the proof of Claim C.1.

C.1.2 The Well-Conditioned Case

In this section, we make the following simplifying assumptions on convex program (8): We are given $C, \lambda \geq 0$ with $\lambda = \frac{C}{\exp(\text{poly}(n,m))}$ such that C upperbounds the optimal value $f(x^*)$, and λ lower-bounds the magnitude of the second-derivative of $f(x)$ when restricted to any line in the feasible set. For the former condition, the reader may think of C as $\sum_{i=1}^{n} v_i([m])$. The latter condition is equivalent to requiring that every eigenvalue of the Hessian matrix of $f(x)$ has magnitude at least λ when evaluated at any point in the feasible set. Under these assumptions, we prove Lemma C.2.

Lemma C.2. *Assume we are given $C \geq f(x^*)$, and a lowerbound $\lambda = \frac{C}{\exp(\text{poly}(n,m))}$ on the magnitude of the second derivative of $f(x)$ everywhere in the feasible set. Algorithm 1, instantiated for combinatorial auctions with $r = r_{\text{poiss}}$, can be simulated in expected time polynomial in n and m.*

Algorithm 1 allocates items according to the distribution $r_{\text{poiss}}(x^*)$. The Poisson rounding scheme, as described in Algorithm 2, requires making m independent decisions, one for each item j. Consider how to simulate this decision for an individual item j. It suffices to do the following in expected polynomial-time: flip uniform coin $p_j \in [0,1]$, and find the minimum index $a(j)$ (if any) such that $\sum_{i\leq a(j)}(1-e^{-x^*ij}) \geq p_j$. For most realizations of p_j, this can be decided using only coarse estimates $\widetilde{x}_{ij}$ to x^*_{ij}. Assume we have an *estimation oracle* for x^* that takes as input a parameter $\delta > 0$, and returns a *δ-estimate* $\widetilde{x}$ of x^*: specifically, satisfying $\widetilde{x}_{ij} - x^*_{ij} \leq \delta$ for each i. When p_j falls outside the "uncertainty regions" of $\widetilde{x}$, such as when $|p_j - \sum_{i'\leq i}(1-e^{-\widetilde{x}_{i'j}})| > \delta n$ for each $i \in [n]$, it is easy to see that we can correctly determine $a(j)$ by using $\widetilde{x}$ in lieu of x. The total measure of these regions is at most $2n^2\delta$, therefore p_j lands outside the uncertainty regions with probability at least $1-2n^2\delta$. By the union bound, $p_1,\ldots,p_m$ all fall outside the uncertainty regions of $\widetilde{x}$ with probability at least $1-2mn^2\delta$.

The following claim shows that if the estimation oracle for x^* can be implemented in time polynomial in $\log(1/\delta)$, then we can simulate the Poisson rounding procedure in expected polynomial-time. It hinges on the fact that the required precision $\log(\frac{1}{\delta})$ follows a distribution with exponentially small tails.

Claim C.3. *Let x^* be the optimal solution of convex program (8). Assume access to a subroutine $B(\delta)$ that returns a δ-estimate of x^* in time $\text{poly}(n,m,\log(1/\delta))$. Algorithm (1) with $r = r_{\text{poiss}}$ can be simulated in expected $\text{poly}(n,m)$ time.*

Proof. Let the random variables $p_1, \ldots, p_m \in [0,1]$ be as in Algorithm (2). Let $\delta = \delta(p_1, \ldots, p_m)$ be the largest value such that $|p_j - \sum_{i' \le i}(1 - e^{-\widetilde{x}_{i'j}})| > \delta n$ for $i \in [n]$ and $j \in [m]$. Any δ-estimate $\widetilde{x}$ of x^* suffices to simulate Algorithm (2) for these draws of $p_1, \ldots, p_m$. We compute such an estimate by invoking $B(\delta)$, which takes time $\text{poly}(n, m, \log(1/\delta))$, and simulate Algorithm (2) accordingly.

To bound the expected running time of the above-described simulation, we need to bound the tail probability of the random variable $\frac{1}{\delta}$. As argued above, $\mathbf{Pr}[\delta(p) \ge \delta_0] \ge 1 - 2mn^2\delta_0 = 1 - \text{poly}(n,m)\delta_0$ for every δ_0. Therefore, the tail probability of $\frac{1}{\delta(p)}$ satisfies $\mathbf{Pr}\left[\frac{1}{\delta(p)} \ge c\right] \le \frac{\text{poly}(n,m)}{c}$. Since the runtime of $B(\delta(p))$ grows only logarithmically in $\frac{1}{\delta(p)}$, a simple geometric summation shows that the expected runtime is bounded by a polynomial in n and m. □

It remains to show that the estimation oracle $B(\delta)$ can be implemented in $\text{poly}(n, m, \log(1/\delta))$ time. At first blush, one may expect that the ellipsoid method can be used in the usual manner here. However, there is one complication: we require an estimate $\widetilde{x}$ that is close to x^* *in solution space* rather than in terms of objective value. Using our assumption on the curvature of $f(x)$, we will reduce finding a δ-estimate of x^* to finding an $1 - \epsilon(\delta)$ approximate solution to convex program (8). The dependence of ϵ on δ will be such that $\epsilon \ge \text{poly}(\delta)/\exp(\text{poly}(n,m))$, therefore we can invoke Claim C.1 to deduce that $B(\delta)$ can be implemented in $\text{poly}(n, m, \log(1/\delta))$ time.

Let $\epsilon = \epsilon(\delta) = \frac{\delta^2\lambda}{2C}$. Plugging in the definition of λ, we deduce that $\epsilon \ge \delta^2/\exp(\text{poly}(n,m))$, which is the desired dependence. It remains to show that if $\widetilde{x}$ is $(1-\epsilon)$-approximate solution to (8), then $\widetilde{x}$ is also a δ-estimate of x^*. Using the fact that $f(x)$ is concave, and moreover its second derivative has magnitude at least λ, it a simple exercise to bound distance of any point x from the optimal point x^* in terms of its sub-optimality $f(x^*) - f(x)$, as follows:

$$f(x^*) - f(x) \ge \frac{\lambda}{2}||x - x^*||_2^2. \tag{25}$$

Assume $\widetilde{x}$ is a $(1-\epsilon)$-approximate solution to (8). Equation (25) implies that

$$||\widetilde{x} - x^*||_2^2 \le \frac{2}{\lambda}\epsilon f(x^*) = \frac{\delta^2}{C} f(x^*) \le \delta^2$$

Therefore, $||x - x^*||_2 \le \delta$, implying that x is a δ-estimate. This completes the proof of Lemma C.2.

C.1.3 Guaranteeing Good Conditioning

In this section, we propose a modification r^+_{poiss} of the Poisson rounding scheme r_{poiss}. We will argue that r^+_{poiss} satisfies all the properties of r_{poiss} established so far, with one exception: the approximation guarantee of Lemma 4.3 is reduced to $1 - \frac{1}{e} - 2^{-mn}$. Then we will show that r^+_{poiss} satisfies the curvature assumption of Lemma C.2, demonstrating that said assumption may be removed. Therefore Algorithm 1, instantiated with $r = r^+_{\text{poiss}}$ for combinatorial auctions with CGS valuations in the product-lottery-value oracle model, is a $(1 - 1/e - 2^{-mn})$ approximate and can be implemented in expected $\text{poly}(n,m)$ time. Finally, we show in Remark C.4 how to recover the 2^{-mn} term to get a clean $1 - 1/e$ approximation ratio, as claimed in Theorem 4.1.

We define r^+_{poiss} in Algorithm 4. Intuitively, r^+_{poiss} allocates as in r_{poiss} with probability $1 - \mu$. Otherwise, with probability $\beta(x)$ it allocates all items to a player chosen at random. β is chosen to be a concave function of x, and can be thought of as adding "concave noise" to r_{poiss}.

Algorithm 4 Modified Poisson Rounding Scheme r^+_{poiss}

Input: Fractional allocation x with $\sum_i x_{ij} \le 1$ for all j, and $0 \le x_{ij} \le 1$ for all i, j.
Output: Feasible allocation $(S_1, \ldots, S_n)$.

Let $\mu = 2^{-nm}$ and $\beta = \frac{1}{nm}\sum_{ij}(1 - x_{ij}^2)$
Draw $q \in [0, 1]$ uniformly at random.
if $q \ge \mu$ **then**
  Let $(S_1, \ldots, S_n) \sim r_{\text{poiss}}(x)$.
else if $q \le \mu\beta$ **then**
  Choose a player i^* uniformly at random.
  Let $S_{i^*} = [m]$, and $S_i = \emptyset$ for all $i \ne i^*$.
else
  Let $(S_1, \ldots, S_n) = (\emptyset, \ldots, \emptyset)$.
end if

We can write the expected welfare $\mathrm{E}[w(r^+_{\text{poiss}}(x))]$ as follows.

$$\begin{aligned}\mathrm{E}[w(r^+_{\text{poiss}}(x))] &= (1-\mu)\,\mathrm{E}[w(r_{\text{poiss}}(x))] + \mu\beta\frac{\sum_i v_i([m])}{n} \\ &= (1 - 2^{-mn})\,\mathrm{E}[w(r_{\text{poiss}}(x))] + \frac{\sum_i v_i([m])}{mn^2 2^{-mn}}\sum_{i,j}(1 - x_{ij}^2). \qquad (26)\end{aligned}$$

It is clear that the expected welfare when using $r = r^+_{\text{poiss}}$ is within $1 - \mu = 1 - 2^{-mn}$ of the expected welfare when using $r = r_{\text{poiss}}$ in the instantiation of Algorithm 1. Using Lemma 4.3, we conclude that r^+_{poiss} is a $(1 - 1/e - 2^{-2mn})$-approximate rounding scheme. Moreover, using Lemma 4.2, as well as the fact that $\sum_{i,j}(1 - x_{ij}^2)$ is a concave function of x, we conclude that r^+_{poiss} is a convex rounding scheme. Therefore, this establishes the analogues of Lemmas 4.3 and 4.2 for r^+_{poiss}. It is elementary to verify that our proof of Lemma C.2 extends to r^+_{poiss} without change.

It remains to show that the well-conditioning assumptions of Section C.1.2 are unnecessary for r^+_{poiss}. Let $C = \sum_{i=1}^n v_i([m])$ and $\lambda = \frac{\sum_{i=1}^n v_i([m])}{mn^2 2^{mn}}$. Clearly, C is an upperbound on the optimal welfare. Moreover, since both terms in (26) are concave, the curvature of $\mathrm{E}[w(r^+_{\text{poiss}}(x))]$ is lowerbounded by λ times the curvature of the function $\sum_{i,j}(1 - x_{ij}^2)$, which is 2 everywhere.

Remark C.4. *In this section, we sacrificed 2^{-mn} in the approximation ratio in order to guarantee expected polynomial runtime of our algorithm even when convex program (8) is not well-conditioned. This loss can be recovered to get a clean $1 - 1/e$ approximation as follows. Given our $(1 - 1/e - 2^{-mn})$-approximate MIDR algorithm $\mathcal{A}$, construct the following algorithm $\mathcal{A}'$: Given an instance of combinatorial auctions, $\mathcal{A}'$ runs $\mathcal{A}$ on the instance with probability $1 - e2^{-mn}$, and with the remaining probability solves the instance optimally by brute force enumeration in time $2^{mn}\,\mathrm{poly}(m, n)$. It was shown in [22] that a random composition of MIDR mechanisms is MIDR, therefore $\mathcal{A}'$ is MIDR. The expected runtime of A' is bounded by the expected runtime of $\mathcal{A}$ plus $e2^{-mn} \cdot 2^{mn}\,\mathrm{poly}(m, n) = O(\mathrm{poly}(n, m))$. Finally, the expected approximation of A' is the weighted average of the approximation ratio of $\mathcal{A}$ and the optimal approximation ratio 1, and is at least $(1 - e2^{-mn}) \cdot (1 - 1/e - 2^{-mn}) + e2^{-mn} \cdot 1 \ge 1 - 1/e$.*

C.2 Combinatorial Public Projects

We now adapt the techniques of Section C.1 to combinatorial public projects.

C.2.1 Approximating the Convex Program

Claim C.5. *There is an algorithm for Combinatorial Public Projects with CGS valuations in the bounded-lottery-value oracle model that takes as input an instance of the problem and an approximation parameter $\epsilon > 0$, runs in $\text{poly}(n, m, \log(1/\epsilon))$ time, and returns a $(1-\epsilon)$-approximate solution to convex program (14).*

As in combinatorial auctions, the first three conditions of Fact A.3 follow from elementary combinatorial optimization. It remains to show that a first-order oracle, as defined in Fact A.3, can be implemented in polynomial-time in the bounded-lottery-value oracle model. We let $f(x)$ denote the objective function of convex program (6) when $r = r_k$. This objective can, by definition, be written as follows.

$$f(x) = \mathop{\mathrm{E}}_{S \sim r_k(x)} \left[\sum_i v_i(S) \right] = \sum_i G_k^{v_i}(x)$$

where v_i is the valuation function of player i and $G_k^{v_i}$ is as defined in (15). By definition, $G_k^{v_i}(x)$ is the outcome of querying the bounded-lottery-value oracle of v_i with bound k and marginals x/k. Therefore, we can evaluate $f(x)$ using n bounded-lottery-value queries, one for each player. It remains to show that we can also evaluate the (multi-variate) derivative $\bigtriangledown f(x)$ of $f(x)$. Using definition (15) and Claim 5.4, we take the partial derivative of $G_k^{v_i}$ with respect to x_j and simplify the resulting expression.

$$\begin{aligned}
\frac{\partial G_k^{v_i}}{\partial x_j}(x) &= \sum_{S \subseteq [m]} -1^{|S|} v_i(S) \sum_{R \subseteq S \setminus \{j\}} -1^{|R|+1} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-1} \\
&= \sum_{S \subseteq [m] \setminus \{j\}} -1^{|S|} \left(v_i(S \cup \{j\}) - v_i(S)\right) \sum_{R \subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-1} \\
&= \sum_{S \subseteq [m]} -1^{|S|} \left(v_i(S \cup \{j\}) - v_i(S)\right) \sum_{R \subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-1} \\
&= \sum_{S \subseteq [m]} v_i(S \cup \{j\}) \Pr\left[r_{k-1}\left(\frac{k-1}{k}x\right) = S\right] - \sum_{S \subseteq [m]} v_i(S) \Pr\left[r_{k-1}\left(\frac{k-1}{k}x\right) = S\right].
\end{aligned} \tag{27}$$

The second equality follows by grouping the terms of the summation by the projection of S onto $[m] \setminus \{j\}$. The third equality follows from the observation that $v(S \cup \{j\}) - v(S) = 0$ when S includes j. The fourth equality follows by a simple re-arrangement and application of Claim 5.4.

Inspect the final form (27) in light of the definition of bounded-lottery-value oracles (Definition 2.4) and the definition of r_k (Section 5.1). Notice that the first term is the expected value of v_i over the $(k-1)$-bounded-lottery with marginals $\frac{k-1}{k}x$ and promise $\{j\}$. The second term is the expected value of v_i over the same lottery without the promise. Therefore, we can evaluate $\frac{\partial G_k^{v_i}}{\partial x_j}(x)$ using two queries to the bounded-lottery-value oracle of player i. This completes the proof of Claim C.5.

C.2.2 The Well-Conditioned Case

Analogous to Section C.1.2, we prove the following.

Lemma C.6. *Assume we are given $C \geq f(x^*)$, and a lowerbound $\lambda = \frac{C}{\exp(\text{poly}(n,m))}$ on the magnitude of the second derivative of $f(x)$ everywhere in the feasible set. Algorithm 1, instantiated for combinatorial public projects with $r = r_k$, can be simulated in expected time polynomial in n and m.*

The k-bounded-lottery rounding scheme (Algorithm 3) requires making k independent random decisions. For $\ell \in \{1, \ldots, k\}$, we draw p_ℓ uniformly from $[0, 1]$ and decide which interval I_j, if any, p_ℓ falls into. In other words, we find the minimum index j_ℓ (if any) such that $\sum_{j \leq j_\ell} x^*_j/k \geq p_\ell$. As in Section C.1.2, an estimate of x^* suffices to make this decision for most realizations of p_ℓ. Specifically, given a *δ-estimate* $\widetilde{x}$ of x^*, i.e. satisfying $\widetilde{x}_j - x^*_j \leq \delta$ for each $j \in [m]$, we can safely use $\widetilde{x}$ in lieu of x^* when $|p_\ell - \sum_{j' \leq j} \widetilde{x}_{j'}/k| > \delta m/k$ for each $j \in [m]$. This occurs with probability at least $1 - 2m^2\delta$ for an individual p_ℓ, and therefore with probability at least $1 - 2km^2\delta$ for all $p_1, \ldots, p_k$ simultaneously. The proof of the following claim is essentially identical to the proof of Claim C.3, and therefore omitted.

Claim C.7. *Let x^* be the optimal solution of convex program* (14)*. Assume access to a subroutine $B(\delta)$ that returns a δ-estimate of x^* in time* $\text{poly}(n, m, \log(1/\delta))$*. Algorithm* (1) *with $r = r_k$ can be simulated in expected* $\text{poly}(n, m)$ *time.*

Finally, the proof that the estimation oracle $B(\delta)$ can be implemented in $\text{poly}(n, m, \log(1/\delta))$ time is identical to that in Section C.1.2. This completes the proof of Lemma C.6.

C.2.3 Guaranteeing Good Conditioning

Analogous to Section C.1.3, we present a modification r^+_k of the k-bounded-lottery rounding which adds "concave noise" to r_k. This is given in Algorithm 5.

Algorithm 5 Modified k-bounded-lottery Rounding Scheme r^+_k

Input: Fractional solution $x \in \mathbb{R}^m$ with $\sum_j x_j \leq k$, and $0 \leq x_j \leq 1$ for all j.
Output: Feasible solution $S \subseteq [m]$ with $|S| \leq k$

Let $\mu = 2^{-nm}$ and $\beta = \frac{1}{m}\sum_{j=1}^{m}(1 - x_j^2)$
Draw $q \in [0, 1]$ uniformly at random.
if $q \geq \mu$ **then**
    Let $S \sim r_k(x)$.
else if $q \leq \mu\beta$ **then**
    Choose project $j^* \in [m]$ uniformly at random.
    Let $S = \{j^*\}$
else
    Let $S = \emptyset$.
end if

We can write the expected welfare $\mathrm{E}[w(r_k^+(x))]$ as follows.

$$\begin{aligned}\mathrm{E}[w(r_k^+(x))] &= (1-\mu)\,\mathrm{E}[w(r_k(x))] + \mu\beta\frac{\sum_{j=1}^m\sum_{i=1}^n v_i(\{j\})}{m} \\ &= (1-2^{-mn})\,\mathrm{E}[w(r_k(x))] + \frac{\sum_{j=1}^m\sum_{i=1}^n v_i(\{j\})}{m^2 2^{-mn}}\sum_{j=1}^m(1-x_j^2). \qquad (28)\end{aligned}$$

As in Section C.1.3, Lemma 5.3 with a reduced approximation ratio of $1-\frac{1}{e}-2^{-mn}$ holds for r_k^+, and Lemmas 5.2 and C.6 extend to r_k^+ without change.

It remains to show that the well-conditioning assumptions of Section C.2.2 are unnecessary for r_k^+. Let $C = m\sum_{j=1}^m\sum_{i=1}^n v_i(\{j\})$ and $\lambda = \frac{C}{m^3 2^{mn}}$. By submodularity of each v_i, we have that C is an upperbound on the optimal welfare. Moreover, as in Section C.1.3, the curvature of $\mathrm{E}[w(r_k^+(x))]$ is lowerbounded by λ times the curvature of the function $\sum_j(1-x_j^2)$, which is 2 everywhere.

Finally, we note that the approximation guarantee can be improved to a clean $1-\frac{1}{e}$ by an essentially identical argument to that in Remark C.4.