Bayesian Optimal Auctions via Multi- to Single-agent Reduction

Saeed Alaei[*] Hu Fu[†] Nima Haghpanah[‡]
Jason Hartline[§] Azarakhsh Malekian[¶]

March 22, 2012

Abstract

We study an abstract optimal auction problem for a single good or service. This problem includes environments where agents have budgets, risk preferences, or multi-dimensional preferences over several possible configurations of the good (furthermore, it allows an agent's budget and risk preference to be known only privately to the agent). These are the main challenge areas for auction theory. A single-agent problem is to optimize a given objective subject to a constraint on the maximum probability with which each type is allocated, a.k.a., an allocation rule. Our approach is a reduction from multi-agent mechanism design problem to collection of single-agent problems. We focus on maximizing revenue, but our results can be applied to other objectives (e.g., welfare).

An optimal multi-agent mechanism can be computed by a linear/convex program on interim allocation rules by simultaneously optimizing several single-agent mechanisms subject to joint feasibility of the allocation rules. For single-unit auctions, Border (1991) showed that the space of all jointly feasible interim allocation rules for n agents is a D-dimensional convex polytope which can be specified by 2^D linear constraints, where D is the total number of all agents' types. Consequently, efficiently solving the mechanism design problem requires a separation oracle for the feasibility conditions and also an algorithm for ex-post implementation of the interim allocation rules. We show that the polytope of jointly feasible interim allocation rules is the projection of a higher dimensional polytope which can be specified by only $O(D^2)$ linear constraints. Furthermore, our proof shows that finding a preimage of the interim allocation rules in the higher dimensional polytope immediately gives an ex-post implementation.

We generalize Border's result to the case of k-unit and matroid auctions. For these problems we give a separation-oracle based algorithm for optimizing over feasible interim allocation rules

[*]email: saeed@cs.umd.edu, Dept. of Computer Science, University of Maryland, College Park, MD 20742. Partially supported by ONR YIP grant N000141110662. Part of this work was done when the author was visiting Northwestern University.

[†]email: hufu@cs.cornell.edu. Dept. of Computer Science, Cornell University, Ithaca, NY 14853. Supported by NSF grants CCF-0643934 and AF-0910940. Part of this work was done when the author was visiting Northwestern University.

[‡]email: nima.haghpanah@gmail.com. Dept. Electrical Engineering & Computer Science, Northwestern University, Evanston, IL 60201.

[§]email: hartline@eecs.northwestern.edu. Dept. Electrical Engineering & Computer Science, Northwestern University, Evanston, IL 60208.

[¶]email: azarakhshm@gmail.com. Dept. Electrical Engineering & Computer Science, Massachusetts Institute of Technology, Cambridge , MA 02139.

and a randomized rounding algorithm for ex post implementation. These ex post implementations have a simple form; they are randomizations over simple greedy mechanisms. Given a ordered subset of agent types, such a greedy mechanisms serves types in the specified order.

1 Introduction

Classical economics and game theory give fundamental characterizations of the structure of competitive behavior. For instance, Nash's (1951) theorem shows that mixed equilibrium gives a complete description of strategic behavior, and the Arrow-Debreu (1954) theorem shows the existence of market clearing prices in multi-party exchanges. In these environments computational complexity has offered further perspective. In particular, mixed equilibrium in general games can be computationally hard to find (Chen and Deng, 2006; Daskalakis et al., 2009), whereas market clearing prices are often easy to find (Devanur et al., 2008; Jain, 2004). In this paper we investigate an analogous condition for auction theory due to Kim Border (1991), give a computationally constructive generalization that further illuminates the structure of auctions, and thereby show that the theory of optimal auctions is tractable.

Consider an abstract optimal auction problem. A seller faces a set of agents. Each agent desires service and there may be multiple ways to serve each agent (e.g., when renting a car, you can get a GPS or not, you can get various insurance packages, and you will pay a total price). Each agent has preferences over the different possible ways she can be served and we refer to this preference as her type. The seller is restricted by the *feasibility* constraint that at most one agent can be served (e.g., only one car in the rental shop). When the agents' types are drawn independently from a known prior distribution, the seller would like to design an auction to optimize her objective, e.g., revenue, in expectation over this distribution, subject to feasibility. Importantly, in this abstract problem we have not made any of the following standard assumptions on the agents' preferences: quasi-linearity, risk-neutrality, or single-dimensionality.

We assume that agents behave strategically and we will analyze an auction's performance in *Bayes-Nash equilibrium*, i.e., where each agent's strategy is a best response to the other agents' strategies and the distribution over their preferences. Without loss of generality the revelation principle (Myerson, 1981) allows for the restriction of attention to *Bayesian incentive compatible* (BIC) mechanisms, i.e., ones where the truthtelling strategy is a Bayes-Nash equilibrium.

Any auction the seller proposes can be decomposed across the agents as follows. From an agent's perspective, the other agents are random draws from the known distribution. Therefore, the composition of these random draws (as inputs), the bid of the agent, and the mechanism induce an *interim allocation rule* which specifies the probability the agent is served as a function of her bid. BIC implies that the agent is at least as happy to bid her type as any other bid.

Applying the same argument to each agent induces a profile of interim allocation rules. These interim allocation rules are jointly feasible in the sense that there exists an auction that, for the prior distribution, induces them. As an example, suppose an agent's type is high or low with probability 1/2 each. Consider two interim allocation rules: rule (a) serves the agent with probability one when her type is high and with probability zero otherwise, and rule (b) serves the agent with probability 1/2 regardless of her type. It is feasible for both agents to have rule (b) or for one agent to have rule (a) and the other to have rule (b); on the other hand, it is infeasible for both agents to have rule (a). This last combination is infeasible because with probability one quarter both agents have high types but we cannot simultaneously serve both of them. An important question in the general theory of auctions is to decide when a profile of interim allocation rules is feasible, and furthermore, when it is feasible, to find an auction that implements it.

A structural characterization of the necessary and sufficient conditions for the aforementioned *interim feasibility* is important for the construction of optimal auctions as it effectively allows the auction problem to be decomposed across agents. If we can optimally serve a single agent for a given interim allocation rule and we can check feasibility of a profile of interim allocation rules, then we can optimize over auctions. Effectively, we can reduce the multi-agent auction problem to a collection of single-agent auction problems.

We now informally describe Border's (1991) characterization of interim feasibility. A profile of interim allocation rules is implementable if for any subspace of the agent types the expected number of items served to agents in this subspace is at most the probability that there is an agent with type in this subspace. Returning to our infeasible example above, the probability that there is an agent with a high type is 3/4 while the expected number of items served to agents with high types is one; Border's condition is violated.

The straightforward formulation of interim feasibility via Border's characterization has exponentially many constraints. Nonetheless, it can be simplified to a polynomial number of constraints in single-item auctions with symmetric agents (where the agents' type space and distribution are identical). This simplification of the characterization has lead to an analytically tractable theory of auctions when agents have budgets (Laffont and Robert, 1996) or are risk averse (Matthews, 1984; Maskin and Riley, 1984).

Results Our main theorem is to show computationally tractable (i.e., in polynomial time in the total number of agents' types) methods for each of the following problems. First, a given profile of interim allocation rules can be checked for interim feasibility. Second, for any feasible profile of interim allocation rules, an auction (i.e., ex post allocation rule) that induces these interim allocation rules can be constructed. In particular, for problems where the seller can serve at most one agent, we show that the exponentially-faceted polytope specified by the interim feasibility constraints is a projection of a quadratically-faceted polytope in a higher dimension, and an ex post allocation rule implementing a feasible profile of interim allocation rules is given immediately by the latter's preimage in the higher dimensional polytope. In particular, this implies that optimal interim allocation rules can be computed by solving a quadratically sized linear/convex program. These results combine to give a (computationally tractable) reduction from the multi-agent auction problem to a collection of single-agent problems. Furthermore, our algorithmic procedure characterizes every single service auction as implementable by a simple token passing game.

We also generalize the interim feasibility characterization and use it to design optimal auctions for the cases where the seller faces a k-unit feasibility constraint, i.e., at most k agents can be served, and more generally to matroid feasibility constraints. Our generalization of the feasibility characterization is based on a simpler network-flow-based approach. Although the number of constraints in this characterization is exponential in the sizes of type spaces, we observe that the constraints define a polymatroid, whose vertices correspond to particularly simple auctions which can be implemented by simple determinist allocation rules based on ranking. Algorithms for submodular function minimization give rise to fast separation oracles which, given a set of interim allocation rules, detect a violated feasibility constraint whenever there is one; expressing any point in the polymatroid as a convex combination of the vertices allows us to implement any feasible interim allocation rule as a distribution over the simple auctions. These enable us again to reduce the multi-agent problem to single-agent problems.

Auction theory is very poorly understood outside the standard single-dimensional quasi-linear revenue maximization environment of Myerson (1981). The main consequence of this work is that even without analytical understanding, optimal auctions can be computationally solved for in environments that include non-quasi-linear utility (e.g., budgets or risk aversion) and multi-dimensional preferences (assuming that the corresponding single-agent problem can be solved). Furthermore, unlike most work in auction theory with budgets or risk-aversion, our framework permits the budgets or risk parameters to be private to the agents.

Related Work Myerson (1981) characterized Bayesian optimal auctions in environments with quasi-linear risk-neutral single-dimensional agent preferences. Bulow and Roberts (1989) reinterpreted Myerson's approach as reducing the multi-agent auction problem to a related single-agent problem. Our work generalize this reduction-based approach to single-item multi-unit auction problems with general preferences.

An important aspect of our approach is that it can be applied to general multi-dimensional agent preferences. Multi-dimensional preferences can arise as distinct values different configurations of the good or service being auctioned, in specifying a private budget and a private value, or in specifying preferences over risk. We briefly review related work for agent preferences with multiple values, budgets, or risk parameters.

Multi-dimensional valuations are well known to be difficult. For example, Rochet and Chone (1998), showed that, because *bunching*[1] can not be ruled out easily, the optimal auctions for multi-dimensional valuations are dramatically different from those for single dimensional valuations. Because of this, most results are for cases with special structure (e.g., Armstrong, 1996; Wilson, 1994; McAfee and McMillan, 1988) and often, by using such structures, reduce the problems to single-dimensional ones (e.g., Spence, 1980; Roberts, 1979; Mirman and Sibley, 1980). Our framework does not need any such structure.

A number of papers consider optimal auctions for agents with budgets (see, e.g., Pai and Vohra, 2008; Che and Gale, 1995; Maskin, 2000). These papers rely on budgets being public or the agents being symmetric; our technique allows for a non-identical prior distribution and private budgets. Mechanism design with risk averse agents was studied by Maskin and Riley (1984) and Matthews (1983). Both works assume i.i.d. prior distributions and have additional assumptions on risk attitudes; our reduction does not require these assumptions.

Our work is also related to a line of work on approximating the Bayesian optimal mechanism. These works tend to look for simple mechanisms that give constant (e.g., two) approximations to the optimal mechanism. Chawla et al. (2007), Briest et al. (2010), and Cai and Daskalakis (2011) consider item pricing and lottery pricing for a single agent; the first two give constant approximations the last gives a $(1 + \epsilon)$-approximation for any ϵ. These problems are related to the single-agent problems we consider. Chawla et al. (2010) and Bhattacharya et al. (2010) extend these approaches to multi-agent auction problems. The point of view of reduction from multi- to single-agent presented in this paper bears close relationship to recent work by Alaei (2011) who gives a reduction from multi- to single-agent mechanism design that loses at most a constant factor of the objective. Our reductions, employing entirely different techniques, give rise to optimal mechanisms instead of approximations thereof.

[1]Bunching refers to the situation in which a group of distinct types are treated the same way in by the mechanism.

Characterization of interim feasibility plays a vital role in this work. For single-item single-unit auctions, necessary and sufficient conditions for interim feasibility were developed through a series of works (Maskin and Riley, 1984; Matthews, 1984; Border, 1991, 2007; Mierendorff, 2011); this characterization has proved useful for deriving properties of mechanisms, Manelli and Vincent (2010) being a recent example. Border (1991) characterized symmetric interim feasible auctions for single-item auctions with identically distributed agent preferences. His characterization is based on the definition of "hierarchical auctions." He observes that the space of interim feasible mechanisms is given by a polytope, where vertices of this polytope corresponding to hierarchical auctions, and interior points corresponding to convex combinations of vertices. Mierendorff (2011) generalize Border's approach and characterization to asymmetric single-item auctions. The characterization via hierarchical auctions differs from our characterization via ordered subset auctions in that hierarchical auctions allow for some types to be relatively unordered with the semantics that these unordered types will be considered in a random order; it is important to allow for this when solving for symmetric auctions. Of course convex combinations over hierarchical auctions and ordered subset auctions provide the same generality. Our work generalizes the characterization from asymmetric single-unit auctions to asymmetric multi-unit and matroid auctions.

Our main result provides computational foundations to the interim feasibility characterizations discussed above. We show that interim feasibility can be checked, that interim feasible allocation rules can be optimized over, and that corresponding ex post implementations can be found. Independently and contemporaneously Cai et al. (2012) provided similar computational foundations for the single-unit auction problem. Their approach to the single-unit auction problem is most comparable to our approach for the multi-unit and matroid auction problems where the optimization problem is written as a convex program which can be solved by the ellipsoid method; while these methods result in strongly polynomial time algorithms they are not considered practical. In contrast, our single-unit approach, when the single-agent problems can be solved by a linear program, gives a single linear program which can be practically solved.

While our work gives computationally tractable interim feasibility characterizations in "service based" environments like multi-unit auctions and matroid auctions; Cai et al. (2012) generalize the approach to multi-item auctions with agents with additive preferences. The problem of designing an optimal auction for agents with multi-dimensional additive preferences is considered one of the main challenges for auction theory and their result, from a computational perspective, solves this problem.

Organization. In Section 2 we describe single- and multi-agent mechanism design problems. In Section 3 we give algorithms for solving two kinds of single-agent problems: multi-item unit-demand preferences and private-value private-budget preferences. In Section 4, we give a high-level description of the multi- to single-agent reduction which allows for efficiently compute optimal mechanisms for many service based environments. The key step therein, an efficient algorithm that implements any jointly feasible set of interim allocation rules, is presented in Section 5. This section is divided into three parts which address single-unit, multi-unit, and matroid feasibility constraints, respectively. Conclusions and extensions are discussed in Section 6.

2 Preliminaries

Single-agent Mechanisms We consider the provisioning of an abstract service. This service may be parameterized by an *attribute*, e.g., quality of service, and may be accompanied by a required payment. We denote the outcome obtained by an agent as $w \in W$. We view this outcome as giving an indicator for whether or not an agent is served and as describing attributes of the service such as quality of service and monetary payments. Let $\mathrm{Alloc}(w) \in \{0, 1\}$ be an indicator for whether the agent is served or not; let $\mathrm{Payment}(w) \in \mathbb{R}$ denote any payment the agent is required to make. In a randomized environment (e.g., randomness from a randomized mechanism or Bayesian environment) the outcome an agent receives is a random variable from a distribution over W. The space of all such distributions is denoted $\Delta(W)$.

The agent has a type t from a finite type space T. This type is drawn from distribution $f \in \Delta(T)$ and we equivalently denote by f the probability mass function. I.e., for every $t \in T$, $f(t)$ is the probability that the type is t. The utility function $u : T \times W \to \mathbb{R}$ maps the agent's type and the outcome to real valued utility. The agent is a von Neumann–Morgenstern expected utility maximizer and we extend u to $\Delta(W)$ linearly, i.e., for $w \in \Delta(W)$, $u(t, w)$ is the expectation of u where the outcome is drawn according to w. We do not require the usual assumption of quasi-linearity.

A single-agent mechanism, without loss of generality by the revelation principle, is just an *outcome rule*, a mapping from the agent's type to a distribution over outcomes. We denote an *outcome rule* by $w : T \to \Delta(W)$. We say that an outcome rule w is *incentive compatible* (IC) and *individually rational* (IR) if for all $t, t' \in T$, respectively,

$$u(t, w(t)) \geq u(t, w(t')), \tag{IC}$$

$$u(t, w(t)) \geq 0. \tag{IR}$$

We refer to restriction of the outcome rule to the indicator for service as the *allocation rule.* As the allocation to each agent is a binary random variable, distributions over allocations are fully described by their expected value. Therefore the allocation rule $x : T \to [0, 1]$ for a given outcome rule w is $x(t) = \mathbf{E}\big[\mathrm{Alloc}(w(t))\big]$.

We give two examples to illustrate the abstract model described above. The first example is the standard quasi-linear risk-neutral preference which is prevalent in auction theory. Here the agent's type space is $T \subset \mathbb{R}_+$ where $t \in T$ represents the agent's valuation for the item. The outcome space is $W = \{0, 1\} \times \mathbb{R}_+$ where an outcome w in this space indicates whether or not the item is sold to the agent, by $\mathrm{Alloc}(w)$, and at what price, by $\mathrm{Payment}(w)$. The agent's quasi-linear utility function is $u(t, w) = t \cdot \mathrm{Alloc}(w) - \mathrm{Payment}(w)$. The second example is that of an m-item unit-demand (also quasi-linear and risk-neutral) preference. Here the type space is $T \subset \mathbb{R}^m_+$ and a type $t \in T$ indicates the agent's valuation for each of the items when the agent's value for no service is normalized to zero. An outcome space is $W = \{0, \ldots, m\} \times \mathbb{R}_+$. The first coordinate of w specifies which item the agent receives or none and $\mathrm{Alloc}(w) = 1$ if it is non-zero; the second coordinate of w specifies the required payment $\mathrm{Payment}(w)$. The agent's utility for w is the value the agent attains for the item received less her payment. Beyond these two examples, our framework can easily incorporate more general agent preferences exhibiting, e.g., risk aversion or a budget limit.

Consider the following single-agent mechanism design problem. A feasibility constraint is given by an upper bound $x(t)$ on the probability that the agent is served as a function of her type t; the distribution on types in T is given by f. The single-agent problem is to find the outcome rule

w^* that satisfies the allocation constraint of x and maximizes the performance, e.g., revenue. This problem is described by the following program:

$$\begin{aligned} \max_w : \quad & \mathbf{E}_{t\sim f, w(t)}\big[\text{Payment}(w(t))\big] \\ \text{s.t.} \quad & \mathbf{E}_{w(t)}\big[\text{Alloc}(w(t))\big] \leq x(t), \qquad \forall t \in T \\ & w \text{ is IC and IR.} \end{aligned} \tag{SP}$$

We denote the outcome rule w^* that optimizes this program by $\text{Outcome}(x)$ and its revenue by $\text{Rev}(x) = \mathbf{E}_{t\sim f, w^*(t)}\big[\text{Payment}(w^*(t))\big]$. We note that, although this paper focuses on revenue maximization, the same techniques presented can be applied to maximize (or minimize) general separable objectives such as social welfare.

Multi-agent Mechanisms There are n independent agents. Agents need not be identical, i.e., agent i's type space is T_i, the probability mass function for her type is f_i, her outcome space is W_i, and her utility function is u_i. The profile of agent types is denoted by $\mathbf{t} = (t_1, \ldots, t_n) \in T_1 \times \cdots \times T_n = \mathbf{T}$, the joint distribution on types is $\mathbf{f} \in \Delta(T_1) \times \cdots \times \Delta(T_n)$, a vector of outcomes is $(w_1, \cdots, w_n) \in \mathbf{W}$, and an allocation is $(x_1, \ldots, x_n) \in \{0,1\}^n$. The mechanism has an inter-agent feasibility constraint that permits serving at most k agents, i.e., $\sum_i x_i \leq k$.[2] A mechanism that obeys this constraint is *feasible*. The mechanism has no inter-agent constraint on attributes or payments.

A mechanism maps type profiles to a (distribution over) outcome vectors via an *ex post outcome rule*, denoted $\hat{\mathbf{w}} : \mathbf{T} \to \Delta(\mathbf{W})$ where $\hat{w}_i(\mathbf{t})$ is the outcome obtained by agent i. We will similarly define $\hat{\mathbf{x}} : \mathbf{T} \to [0,1]^n$ as the *ex post allocation rule* (where $[0,1] \equiv \Delta(\{0,1\})$). The ex post allocation rule $\hat{\mathbf{x}}$ and the probability mass function $\mathbf{f}$ on types induce *interim* outcome and allocation rules. For agent i with type t_i and $\mathbf{t} \sim \mathbf{Dist}_{\mathbf{t}}[\mathbf{t} \mid t_i]$ the interim outcome and allocation rules are $w_i(t_i) = \mathbf{Dist}_{\mathbf{t}}\big[\hat{w}_i(\mathbf{t}) \mid t_i\big]$ and $x_i(t_i) = \mathbf{Dist}_{\mathbf{t}}\big[\hat{x}_i(\mathbf{t}) \mid t_i\big] \equiv \mathbf{E}_{\mathbf{t}}\big[\hat{x}_i(\mathbf{t}) \mid t_i\big]$.[3] A profile of interim allocation rules is feasible if it is derived from an ex post allocation rule as described above; the set of all feasible interim allocation rules is denoted by $\mathbb{X}$. A mechanism is Bayesian incentive compatible and interim individually rational if equations (IC) and (IR), respectively, hold for all i and all t_i.

Consider again the examples described previously of quasi-linear single-dimensional and unit-demand preferences. For the single-dimensional example, the multi-agent mechanism design problem is the standard single-item k-unit auction problem. For the unit-demand example, the multi-agent mechanism design problem is an *attribute auction*. In this problem there are k-units available and each unit can be configured in one of m ways. Importantly, the designer's feasibility constraint restricts the number of units sold to be k but places no restrictions on how the units can be configured. E.g., a restaurant has k tables but each diner can order any of the m entrees on the menu.

A reduction from multi-agent mechanism design to single-agent mechanism design as we have described above would assume that for any types pace T_i, any probability mass function f_i, and interim allocation rule x_i, the optimal outcome rule $\text{Outcome}(x_i)$ and its performance $\text{Rev}(x_i)$ can be found efficiently (see Section 3 for examples). The goal then is to construct an optimal

[2]Furthermore, in Section 5.3, we review the theory of *matroids* and extend our basic results environments with feasibility constraint derived from a matroid set system.

[3]We use notation $\mathbf{Dist}[X \mid E]$ to denote the distribution of random variable X conditioned on the event E.

multi-agent auction from these single-agent mechanisms. Our approach to such a reduction is as follows.

1. Optimize, over all feasible profiles of interim allocation rules $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{X}$, the sum of performances of the allocation rules $\sum_i \mathrm{Rev}(x_i)$.
2. Implement the profile of interim outcome rules $\mathbf{w}$ given by $w_i = \mathrm{Outcome}(x_i)$ with a feasible ex post outcome rule $\hat{\mathbf{w}}$.

Two issues should be noted. First, Step 2 requires an argument that the existence of a feasible ex post outcome rule for a given profile of interim allocation rules implies the existence of one that combines the optimal interim outcome rules from Outcome$(\cdot)$. We address this issue in Section 4. Second, Step 1 requires that we optimize over jointly feasible interim allocation rules, and after solving for $\mathbf{x}$, its implementation by an ex post allocation rule is needed to guide Step 2. We address this issue in Section **??**. For single-unit (i.e., $k = 1$) auctions a characterization of the necessary and sufficient condition for interim feasibility was provided by Kim Border.

Theorem 1 (Border, 1991)**.** *In a single-item auction environment, interim allocation rules* $\mathbf{x}$ *are feasible (i.e.,* $\mathbf{x} \in \mathbb{X}$*) if and only if the following holds:*

$$\forall S_1 \subseteq T_1, \cdots, \forall S_n \subseteq T_n : \qquad \sum_{i=1}^{n} \mathbf{E}\big[x_i(t_i) \;\big|\; t_i \in S_i\big] \cdot \mathbf{Pr}[t_i \in S_i] \leq \mathbf{Pr}_{\mathbf{t}\sim\mathbf{f}}\big[\exists i \in [n] : t_i \in S_i\big] \tag{MRMB}$$

3 The Single-agent Problem

Given an allocation rule $x(\cdot)$ as a constraint the single-agent problem is to find the (possibly randomized) outcome rule $w(\cdot)$ that allocates no more frequently that $x(\cdot)$, i.e., $\forall t \in T$, $\mathbf{E}_{w(t)}\big[\mathrm{Alloc}(w(t))\big] \leq x(t)$, with the maximum expected performance. Recall that the optimal such outcome rule is denoted Outcome(x) and its performance (e.g., revenue) is denoted Rev(x). We first observe that Rev$(\cdot)$ is concave.

Proposition 1. Rev$(\cdot)$ *is a concave function in* x*.*

Proof. Consider any two allocation rules x and x', and any $\alpha \in [0, 1]$. Define x'' to be $\alpha x + (1-\alpha)x'$. We will show that $\alpha \,\mathrm{Rev}(x) + (1 - \alpha)\,\mathrm{Rev}(x') \leq \mathrm{Rev}(x'')$, which proves the claim. To see this, let w and w' be Outcome(x) and Outcome(x'), respectively. Define w'' to be the outcome rule that runs w with probability α, and w' with probability $1 - \alpha$. The incentive compatibility of outcome rules w and w' imply the incentive compatibility of w'', since for any $t, t' \in T$,

$$\begin{aligned}
\mathbf{E}\Big[u(t, w''(t))\Big] &= \alpha\mathbf{E}\big[u(t, w(t))\big] + (1-\alpha)\mathbf{E}\Big[u(t, w'(t))\Big] \\
&\geq \alpha\mathbf{E}\Big[u(t, w(t'))\Big] + (1-\alpha)\mathbf{E}\Big[u(t, w'(t'))\Big] \\
&= \mathbf{E}\Big[u(t, w''(t'))\Big].
\end{aligned}$$

Also, w'' is feasible as $\mathbf{E}\big[\mathrm{Alloc}(w''(t))\big] = \alpha\mathbf{E}\big[\mathrm{Alloc}(w(t))\big] + (1-\alpha)\mathbf{E}\big[\mathrm{Alloc}(w'(t))\big] \leq x''(t)$ for all $t \in T$. As a result, Rev(x'') is at least the revenue of w'', which is in turn equal to $\alpha \,\mathrm{Rev}(x) + (1 - \alpha)\,\mathrm{Rev}(x')$. □

We now give two examples for which the single-agent problem is computationally tractable. Both of these example are multi-dimensional. The first example is that of a standard multi-item unit-demand preferences. The second example that of a single-item with a private budget. For both of these problems the single-agent problem can be expressed as a linear program with size polynomial in the cardinality of the agent's type space.

3.1 Quasi-linear Unit-demand Preferences

There are m items available. There is a finite type space $T \subset \mathbb{R}_+^m$; the outcome space W is the direct product between an assignment to the agent of one of the m items, or none, and a required payment. $\Delta(W)$ is the cross product of a probability distribution over which item the agent receives and a probability distribution over payments. Without loss of generality for a quasi-linear agent such a randomized outcome can be represented as $w = (w_1, \ldots, w_m, w_p)$ where for $j \in [m]$, w_j is the probability that the agent receives item j and w_p is the agent's required payment.

A single-agent mechanism assigns to each type an outcome as described above. An outcome rule specifies an outcome for any type t of the agent as $w(t) = (w_1(t), \ldots, w_m(t), w_p(t))$. This gives $m+1$ non-negative real valued variables for each of $|T|$ types. The following linear program, which is a simple adaptation of one from Briest et al. (2010) to include the feasibility constraint given by x, solves for the optimal single-agent mechanism:

$$\begin{aligned}
\max: \quad & \textstyle\sum_{t\in T} f(t) w_p(t) & \\
\text{s.t.} \quad & \textstyle\sum_j w_j(t) \leq x(t) & \forall t \in T \\
& \textstyle\sum_j t_j w_j(t) - w_p(t) \geq \sum_j t_j w_j(t') - w_p(t') & \forall t, t' \in T \\
& \textstyle\sum_j t_j w_j(t) - w_p(t) \geq 0 & \forall t \in T.
\end{aligned}$$

The optimal outcome rule from this program is $w^* = \text{Outcome}(x)$ and its performance is $\text{Rev}(x) = \mathbf{E}_{t\sim f}\left[w_p^*(t)\right]$.

Proposition 2. *The single-agent m-item unit-demand problem can be solved in polynomial time in m and $|T|$.*

3.2 Private budget preferences.

There is a single item available. The agent has a private value for this item and a private budget, i.e., $T \subset \mathbb{R}_+^2$; we will denote by t_v and t_b this value and budget respectively. The outcome space is $W = \{0,1\} \times \mathbb{R}$ where for $w \in W$ the first coordinate w_x denotes whether the agent receives the item or not and the second coordinate w_p denotes her payment. The agent's utility is

$$u(t, w) = \begin{cases} t_v w_x - w_p & \text{if } w_p \leq t_b \text{, and} \\ -\infty & \text{otherwise.} \end{cases}$$

Claim 1 below implies that when optimizing over distributions on outcomes we can restrict attention to $[0,1] \times [0,1] \times \mathbb{R}_+ \subset \Delta(W)$ where the first coordinate denotes the probability that the agent receives the item, the second coordinate denotes the probability that the agent makes a non-zero payment, and the third coordinate denotes the non-zero payment made.

Claim 1. *Any incentive compatible and individually rational outcome rule can be converted into an outcome rule above with the same expected revenue.*

As a sketch of the argument to show this claim, note that if an agent with type t receives randomized outcome w she is just as happy to receive the item with the same probability and pay her budget with probability equal to her previous expected payment divided by her budget. Such a payment is budget feasible and has the same expectation as before. Furthermore, this transformation only increases the maximum payment that any agent makes which means that the relevant incentive compatibility constraints are only fewer. Importantly, the only incentive constraints necessary are ones that prevent types with higher budgets from reporting types with lower budgets.

A single-agent mechanism assigns to each type an outcome as described above. We denote the distribution over outcomes for t by $w(t) = (w_x(t), w_\rho(t), t_b)$ where only the first two coordinates are free variables. This gives two non-negative real valued variables for each of $|T|$ types. The following linear program solves for the optimal single-agent mechanism:

$$\begin{aligned}
\max: \quad & \textstyle\sum_{t\in T} f(t) t_b w_\rho(t) & \\
\text{s.t.} \quad & w_x(t) \le x(t) & \forall t \in T \\
& t_v w_x(t) - t_b w_\rho(t) \ge t_v w_x(t') - t'_b w_\rho(t') & \forall t, t' \in T \text{ with } t'_b \le t_b \\
& t_v w_x(t) - t_b w_\rho(t) \ge 0 & \forall t \in T \\
& w_\rho(t) \le 1 & \forall t \in T.
\end{aligned}$$

The optimal outcome rule from this program is $w^* = \text{Outcome}(x)$ and its performance is $\text{Rev}(x) = \mathbf{E}_{t\sim f}\left[t_b w^*_\rho(t)\right]$.

Proposition 3. *The single-agent private budget problem can be solved in polynomial time in $|T|$.*

4 Multi- to Single-agent Reductions

An ex post allocation rule $\hat{\mathbf{x}}$ takes as its input a profile of types $\mathbf{t} = (t_1, \dots, t_n)$ of the agents, and indicates by $\hat{x}_i(\mathbf{t})$ a set of at most k winners. Agent i's type $t_i \in T_i$ is drawn independently at random from distribution $f_i \in \Delta(T_i)$. An ex post allocation rule implements an interim allocation rule $x_i : T_i \to [0, 1]$, for agent i, if the probability of winning for agent i conditioned on her type $t_i \in T_i$ is exactly $x_i(t_i)$, where the probability is taken over the random types other agents and the random choices of the allocation rule. A profile of interim allocation rules $\mathbf{x} = (x_1, \dots, x_n)$ is feasible if and only if it can be implemented by some ex post allocation rule. $\mathbb{X}$ denotes the space of all feasible profiles of interim allocation rules.

The optimal performance (e.g., revenue) of the single-agent problem with allocation constraint given by x is denoted $\text{Rev}(x)$. The outcome rule corresponding to this optimal revenue is $\text{Outcome}(x)$. Given any feasible interim allocation rule $\mathbf{x} \in \mathbb{X}$ we would like to construct an auction with revenue $\sum_i \text{Rev}(x_i)$. We need to be careful because $\text{Outcome}(x_i)$, by definition, is only required to have allocation rules *upper bounded* by x_i (see (SP) in Section 2), while the ex post allocation rule $\hat{x}_i$ implements x_i exactly, and hence we may need to scale down $\hat{x}_i$ accordingly. This is defined formally as follows.

Definition 1. An optimal auction $\hat{\mathbf{w}}^*$ for feasible interim allocation rule $\mathbf{x}$ (with corresponding ex post allocation rule $\hat{\mathbf{x}}$) is defined as follows on $\mathbf{t}$. For agent i:

1. Let $w_i^* = \text{Outcome}(x_i)$ be the optimal outcome rule for allocation constraint x_i.
2. Let $x_i^* = \mathbf{E}\big[\text{Alloc}(w_i^*)\big]$ be the allocation rule corresponding to outcome rule w_i^*.
3. If $\hat{x}_i(\mathbf{t}) = 1$, output

$$\hat{w}_i^*(\mathbf{t}) \sim \begin{cases} \mathbf{Dist}\big[w_i^*(t_i) \bigm| \text{Alloc}(w_i^*(t_i)) = 1\big] & \text{w.p. } x_i^*(t_i)/x_i(t_i)\text{, and} \\ \mathbf{Dist}\big[w_i^*(t_i) \bigm| \text{Alloc}(w_i^*(t_i)) = 0\big] & \text{otherwise.} \end{cases}$$

4. Otherwise (when $\hat{x}_i(\mathbf{t}) = 0$), output $\hat{w}_i^*(\mathbf{t}) \sim \mathbf{Dist}\big[w_i^*(t_i) \bigm| \text{Alloc}(w_i^*(t_i)) = 0\big]$.

Proposition 4. *For any feasible interim allocation rule $\mathbf{x} \in \mathbb{X}$, the optimal auction for this rule has expected revenue $\sum_i \text{Rev}(x_i)$.*

Proof. The ex post outcome rule $\hat{\mathbf{w}}^*$ of the auction, by construction, induces interim outcome rule $\mathbf{w}^*$ for which the revenue is as desired. □

The optimal multi-agent auction is the solution to optimizing the cumulative revenue of individual single-agent problems subject to the joint interim feasibility constraint given by $\mathbf{x} \in \mathbb{X}$.

Proposition 5. *The optimal revenue is given by the convex program*

$$\max_{\mathbf{x} \in \mathbb{X}} : \sum\nolimits_i \text{Rev}_i(x_i). \tag{CP}$$

Proof. This is a convex program as $\text{Rev}(\cdot)$ is concave and $\mathbb{X}$ is convex (convex combinations of feasible interim allocation rules are feasible). By Proposition 4 this revenue is attainable; therefore, it is optimal. □

5 Optimization and Implementation of Interim Allocation Rules

In this section we address the computational issues pertaining to (i) solving optimization problems over the space of feasible interim allocation rules, and (ii) ex post implementation of such a feasible interim allocation rule. We present computationally tractable methods for both problems.

Normalized interim allocation rules. It will be useful to "flatten" the interim allocation rule $\mathbf{x}$ for which $\mathbf{x}_i(t_i)$ denotes the probability that i with type t_i is served (randomizing over the mechanism and the draws of other agent types); we do so as follows. Without loss of generality, we assume that the type spaces of different agents are disjoint.[4] Denoting the set of all types by $T_N = \bigcup_i T_i$, the interim allocation rule can be flattened as a vector in $[0,1]^{T_N}$.

Definition 2. The *normalized interim allocation rule* $\overline{x} \in [0,1]^{T_N}$ corresponding to interim allocation rule $\mathbf{x}$ under distribution $\mathbf{f}$ is defined as

$$\overline{x}(t_i) = x_i(t_i) f_i(t_i) \qquad\qquad \forall t_i \in T_N$$

[4]This can be achieved by labeling all types of each agent with the name of that agent, i.e., for each $i \in [n]$ we can replace T_i with $T_i' = \{(i,t) | t \in T_i\}$ so that $T_1', \cdots, T_n'$ are disjoint.

For the rest of this section, we refer to interim allocation rules via $\overline{x}$ instead of $\mathbf{x}$. Note that there is a one-to-one correspondence between $\overline{x}$ and $\mathbf{x}$ as specified by the above linear equation; so any linear of convex optimization problem involving $\mathbf{x}$ can be written in terms of $\overline{x}$ without affecting its linearity or convexity. As $\mathbb{X}$ denotes the space of feasible interim allocation rules $\mathbf{x}$, we will use $\overline{\mathbb{X}}$ to denote the space of feasible normalized interim allocation rules.

In the remainder of this section we characterize interim feasibility and show that normalized interim allocation rules can be optimized over and implemented in polynomial time.

5.1 Single Unit Feasibility Constraints

In this section, we consider environments where at most one agent can be allocated to. For such environments, we characterize interim feasibility as implementability via a particular, simple *stochastic sequential allocation* mechanism. Importantly, the parameters of this mechanism are easy to optimize efficiently.

A stochastic sequential allocation mechanism is parameterized by a stochastic transition table. Such a table specifies the probability by which an agent with a given type can steal a token from a preceding agent with a given type. For simplicity in describing the process we will assume the token starts under the possession of a "dummy agent" indexed by 0; the agents are then considered in the arbitrary order from 1 to n; and the agent with the token at the end of the process is the one that is allocated (or none are allocated if the dummy agent retains the token).

Definition 3 (stochastic sequential allocation mechanism)**.** Parameterized by a stochastic transition table π, the *stochastic sequential allocation mechanism (SSA)* computes the allocations for a type profile $\mathbf{t} \in \mathbf{T}$ as follows:

1. Give the token to the dummy agent 0 with dummy type t_0.
2. For each agent i: (in order of 1 to n)

   If agent i' has the token, transfer the token to agent i with probability $\pi(t_{i'}, t_i)$.
3. Allocate to the agent who has the token (or none if the dummy agent has it).

First, we present a dynamic program, in the form of a collection of linear equations, for calculating the interim allocation rule implemented by SSA for a given π. Let $y(t_{i'}, i)$ denote the ex-ante probability of the event that agent i' has type $t_{i'}$ and is holding the token at the end of iteration i. Let $z(t_{i'}, t_i)$ denote the ex-ante probability in iteration i of SSA that agent i has type t_i and takes the token from agent i' who has type $t_{i'}$.

The following additional notation will be useful in this section. For any subset of agents $N' \subseteq N = \{1, \dots, n\}$, we define $T_{N'} = \bigcup_{i \in N'} T_i$ (Recall that without loss of generality agent type spaces are assumed to be disjoint.). The shorthand notation $t_i \in S$ for $S \subseteq T_N$ will be used to quantify over all types in S and their corresponding agents (i.e., $\forall t_i \in S$ is equivalent to $\forall i \in N, \forall t_i \in S \cap T_i$).

The normalized interim allocation rule $\overline{x}$ resulting from the SSA is exactly given by the dynamic program specified by the following linear equations.

$$\begin{aligned}
y(t_0,0) &= 1, && && (\mathbb{S}.1)\\
y(t_i,i) &= \sum_{t_{i'}\in T_{\{0,\dots,i-1\}}} z(t_{i'},t_i), && \forall t_i \in T_{\{1,\dots,n\}} && (\mathbb{S}.2)\\
y(t_{i'},i) &= y(t_{i'},i-1) - \sum_{t_i\in T_i} z(t_{i'},t_i), && \forall i \in \{1,\dots,n\}, \forall t_{i'} \in T_{\{0,\dots,i-1\}} && (\mathbb{S}.3)\\
z(t_{i'},t_i) &= y(t_{i'},i-1)\pi(t_{i'},t_i)f_i(t_i), && \forall t_i \in T_{\{1,\dots,n\}}, \forall t_{i'} \in T_{\{0,\dots,i-1\}} && (\pi)\\
\overline{x}(t_i) &= y(t_i,n), && \forall t_i \in T_{\{1,\dots,n\}}
\end{aligned}$$

Note that π is the only adjustable parameter in the SSA algorithm, so by relaxing the equation (π) and replacing it with the following inequality we can specify all possible dynamics of the SSA algorithm.

$$0 \le z(t_{i'},t_i) \le y(t_{i'},i-1)f_i(t_i), \qquad \forall t_i \in T_{\{1,\dots,n\}}, \forall t_{i'} \in T_{\{0,\dots,i-1\}} \qquad (\mathbb{S}.4)$$

Let $\mathbb{S}$ denote the convex polytope captured by the 4 sets of linear constraints ($\mathbb{S}$.1) through ($\mathbb{S}$.4) above, i.e., $(y,z) \in \mathbb{S}$ iff y and z satisfy the aforementioned constraints. Note that every $(y,z) \in \mathbb{S}$ corresponds to some stochastic transition table π by solving equation (π) for $\pi(t_i,t_{i'})$. We show that $\mathbb{S}$ captures all feasible normalized interim allocation rules, i.e., the projection of $\mathbb{S}$ on $\overline{x}(\cdot) = y(\cdot,n)$ is exactly $\overline{\mathbb{X}}$, as formally stated by the following theorem.

Theorem 2. *A normalized interim allocation rule $\overline{x}$ is feasible if and only if it can be implemented by the SSA algorithm for some choice of stochastic transition table π. In other words, $\overline{x} \in \overline{\mathbb{X}}$ iff there exists $(y,z) \in \mathbb{S}$ such that $\overline{x}(t_i) = y(t_i,n)$ for all $t_i \in T_N$.*

Corollary 1. *Given a blackbox for each agent i that solves for the optimal expected revenue $\mathrm{Rev}_i(x_i)$ for any feasible interim allocation rule $\mathbf{x}$, the optimal interim allocation rule can be computed by the following convex program which is of quadratic size in the total number of types.*

$$\begin{aligned}
\textit{maximize} \quad & \sum_{i=1}^{n} \mathrm{Rev}_i(x_i)\\
\textit{subject to} \quad & y(t_i,n) = \overline{x}(t_i) = x_i(t_i)f_i(t_i), \qquad \forall t_i \in T_N\\
& (y,z) \in \mathbb{S}.
\end{aligned}$$

Furthermore, given an optimal assignment for this program, the computed interim allocation rule can be implemented by SSA using the the stochastic transition table defined by:[5]

$$\pi(t_{i'},t_i) = \frac{z(t_{i'},t_i)}{y(t_{i'},i-1)f_i(t_i)}, \qquad \forall t_i \in T_{\{1,\dots,n\}}, \forall t_{i'} \in T_{\{0,\dots,i-1\}}.$$

Next, we present a few definitions and lemmas that are used in the proof of Theorem 2. Two transition tables π and π' are considered *equivalent* if their induced normalized interim allocation rules for SSA are equal. Type t_i is called *degenerate* for π if in the execution of SSA the token

[5]If the denominator is zero, i.e., $y(t_i,i'-1) = 0$, we can set $\pi(t_i,t_{i'})$ to an arbitrary value in $[0,1]$.

is sometimes passed to type t_i but it is always taken away from t_i later, i.e., if $y(t_i, i) > 0$ but $y(t_i, n) = 0$. The stochastic transition table π is degenerate if there is a degenerate type. For π, type t_i is *augmentable* if there exists a π' (with a corresponding y') which is *equivalent* to π for all types expect t_i and has $y(t_i, n) > y'(t_i, n)$.[6]

Lemma 1. *For any stochastic transition table π there exists an equivalent π' that is non-degenerate.*

Lemma 2. *For any non-degenerate stochastic transition table π, any non-augmentable type t_i always wins against any augmentable type $t_{i'}$. I.e.,*

- *if $i' < i$ and $t_{i'}$ has non-zero probability of holding the token then $\pi(t_{i'}, t_i) = 1$, i.e., t_i always takes the token away from $t_{i'}$, and*
- *if $i < i'$ and t_i has non-zero probability of holding the token then $\pi(t_i, t_{i'}) = 0$, i.e., $t_{i'}$ never takes the token away from t_i.*

It is possible to view the token passing in stochastic sequential allocation as a network flow. From this perspective, the augmentable and non-augmentable types form a minimum-cut and Lemma 2 states that the token must eventually flow from the augmentable to non-augmentable types. We defer the proof of this lemma to Appendix A where the main difficulty in its proof is that the edges in the relevant flow problem have dynamic(non-constant) capacities.

Proof of Theorem 2. Any normalized interim allocation rule that can be implemented by the SSA algorithm is obviously feasible, so we only need to prove the opposite direction. The proof is by contradiction, i.e., given a normalized interim allocation rule $\overline{x}$ we show that if there is no $(y, z) \in \mathbb{S}$ such that $\overline{x}(\cdot) = y(\cdot, n)$, then $\overline{x}$ must be infeasible. Consider the following linear program for a given $\overline{x}$ (i.e., $\overline{x}$ is constant).

$$\begin{array}{lll} \text{maximize} & \displaystyle\sum_{t_i \in T_{\{1,\dots,n\}}} y(t_i, n) & \\ \text{subject to} & y(t_i, n) \le \overline{x}(t_i), & \forall t_i \in T_{\{1,\dots,n\}} \\ & (y, z) \in \mathbb{S}. & \end{array}$$

Let (y, z) be an optimal assignment of this LP. If the first set of inequalities are all tight (i.e., $\overline{x}(\cdot) = y(\cdot, n)$) then $\overline{x}$ can be implemented by the SSA, so by contradiction there must exists a type $\tau^* \in T_N$ for which the inequality is not tight. Note that τ^* cannot be augmentable — otherwise, by the definition of augmentability, the objective of the LP could be improved. Partition T_N to augmentable types T_N^+ and non-augmentable types T_N^-. Note that T_N^- is non-empty because $\tau^* \in T_N^-$. Without loss of generality, by Lemma 1 we may assume that (y, z) is non-degenerate.[7]

An agent wins if she holds the token at the end of the SSA algorithm. The ex ante probability that some agent with non-augmentable type wins is $\sum_{t_i \in T_N^-} y(t_i, n)$. On the other hand, Lemma 2 implies that the first (in the order agents are considered by SSA) agent with non-augmentable type will take the token from her predecessors and, while she may lose the token to another non-augmentable type, the token will not be relinquished to any augmentable type. Therefore,

[6]We define t_0 to be augmentable unless the dummy agent never retains the token in which case all agents are non-augmentable (and for technical reasons we declare the dummy agent to be non-augmentable as well).

[7]By Lemma 1, there exits an non-degenerate assignment with the same objective value.

the probability that an agent with a non-augmentable type is the winner is exactly equal to the probability that at least one such agent exists, therefore

$$\mathbf{Pr}_{\mathbf{t}\sim\mathbf{f}}\left[\exists i : t_i \in T_N^-\right] = \sum_{t_i \in T_N^-} y(t_i, n) < \sum_{t_i \in T_N^-} \overline{x}(t_i).$$

The second inequality follows from the assumption above that τ^* satisfies $y(\tau^*, n) < \overline{x}(\tau^*)$. We conclude that $\overline{x}$ requires an agent with non-augmentable type to win more frequently than such an agent exists, which is a contradiction to interim feasibility of $\overline{x}$. □

The contradiction that we derived in the proof of Theorem 2 yields a necessary and sufficient condition, as formally stated in the following corollary, for feasibility of any given normalized interim allocation rule.

Corollary 2. *A normalized interim allocation rule $\overline{x}$ is feasible if and only if*

$$\sum_{\tau \in S} \overline{x}(\tau) \leq \mathbf{Pr}_{\mathbf{t}\sim\mathbf{f}}\left[\exists i : t_i \in S\right], \qquad \forall S \subseteq T_N \qquad \text{(MRMB)}$$

The necessity of condition (MRMB) is trivial and its sufficiency was previously proved by Border (1991). This condition implies that the space of all feasible normalized interim allocation rules, $\overline{\mathbb{X}}$, can be specified by 2^D linear constraints on D-dimensional vectors $\overline{x}$. An important consequence of Theorem 2 is that $\overline{\mathbb{X}}$ can equivalently be formulated by only $O(D^2)$ variables and $O(D^2)$ linear constraints as a projection of $\mathbb{S}$, therefore any optimization problem over $\overline{\mathbb{X}}$ can equivalently be solved over $\mathbb{S}$.

5.2 k-Unit Feasibility Constraints

In this section, we consider environments where at most k agents can be simultaneously allocated to. First, we generalize Border's characterization of interim feasibility to environments with k-unit feasibility constraint. Our generalization implies that the space of feasible normalized interim allocation rules is a polymatroid. Second, we observe that optimization problems can be efficiently solved over polymatroids; this allows us to optimize over feasible interim allocation rules. Third, we show that the normalized interim allocation rules corresponding to the vertices of this polymatroid are implemented by simple deterministic ordered-subset-based allocation mechanisms. Furthermore, for any point in this polymatroid, the corresponding normalized interim allocation rule can be implemented by, (i) expressing it as a convex combination of the vertices of the polymatroid, (ii) sampling from this convex combination, and (iii) using the ordered subset mechanism corresponding to the sampled vertex. We present an efficient randomized rounding routine for rounding a point in a polymatroid to a vertex which combines the steps (i) and (ii). These approaches together yield efficient algorithms for optimizing and implementing interim allocation rules.

Polymatroid Preliminaries. Consider an arbitrary set function $\mathcal{F} : 2^U \to \mathbb{R}_+$ defined over an arbitrary finite set U; let $P_\mathcal{F}$ denote the polytope associate with $\mathcal{F}$ defined as

$$P_\mathcal{F} = \left\{ y \in \mathbb{R}_+^U \middle| \forall S \subseteq U : y(S) \leq \mathcal{F}(S) \right\}$$

where $y(S)$ denotes $\sum_{s \in S} y(s)$. The convex polytope $P_{\mathcal{F}}$ is called a *polymatroid* if $\mathcal{F}$ is a submodular function. Even though a polymatroid is defined by an exponential number of linear inequalities, the separation problem for any given $y \in \mathbb{R}_+^U$ can be solved in polynomial time as follows: find $S^* = \arg\min_S \mathcal{F}(S) - y(S)$; if y is infeasible, the inequality $y(S^*) \leq \mathcal{F}(S^*)$ must be violated, and that yields a separating hyperplane for y. Note that $\mathcal{F}(S) - y(S)$ is itself submodular in S, so it can be minimized in strong polynomial time. Consequently, optimization problems can be solved over polymatroids in polynomial time. Next, we describe a characterization of the vertices of a polymatroid. This characterization plays an important role in our proofs and also in our ex post implementation of interim allocation rules.

Definition 4 (ordered subset)**.** For an arbitrary finite set U, an *ordered subset* $\pi \subset U$ is given by an ordering on elements $\pi = (\pi_1, \ldots, \pi_{|\pi|})$ where shorthand notation $\pi_r \in \pi$ denotes the rth element in π.

Proposition 6. *Let $\mathcal{F} : 2^U \to \mathbb{R}_+$ be an arbitrary non-decreasing submodular function with $\mathcal{F}(\emptyset) = 0$ and let $P_{\mathcal{F}}$ be the associated polymatroid with the set of vertices* $\textsc{Vertex}(P_{\mathcal{F}})$*. Every ordered subset π of U (see Definition 4) corresponds to a vertex of $P_{\mathcal{F}}$, denoted by* $\textsc{Vertex}(P_{\mathcal{F}}, \pi)$*, which is computed as follows.*

$$\forall s \in U : \qquad\qquad y(s) = \begin{cases} \mathcal{F}(\{\pi_1, \ldots, \pi_r\}) - \mathcal{F}(\{\pi_1, \ldots, \pi_{r-1}\}) & \text{if } s = \pi_r \in \pi \\ 0 & \text{if } s \notin \pi \end{cases}$$

Furthermore, for every $y \in$ $\textsc{Vertex}(P_{\mathcal{F}})$ *there exist a corresponding π.*

It is easy to see that for any $y \in \textsc{Vertex}(P_{\mathcal{F}})$, an associated π can be found efficiently by a greedy algorithm (see Schrijver (2003) for a comprehensive treatment of polymatroids).

Ordered subset allocation mechanisms. The following class of allocation mechanisms are of particular importance both in our characterization of interim feasibility and in ex post implementation of interim allocation rules.

Definition 5 (ordered subset allocation mechanism)**.** Parameterized by a *ordered subset* π of T_N (see Definition 4), the *ordered subset mechanism*, on profile of types $\mathbf{t} \in \mathbf{T}$, orders the agents based on their types according to π, and allocates to the agents greedily (e.g., with k units available the k first ordered agents received a unit). If an agent i with type $t_i \notin \pi$ is never served.[8]

Characterization of interim feasibility. Border's characterization of interim feasibility for $k = 1$ unit auctions states that the probability of serving a type in a subspace of type space is no more than the probability that a type in that subspace shows up. This upper bound is equivalent to the expected minimum of one and the number of types from the subspace that show up; furthermore, this equivalent phrasing of the upper bound extends to characterize interim feasibility in k-unit auctions.

[8]The virtual valuation maximizing mechanisms from the classical literature on revenue maximizing auctions are ordered subset mechanisms, see, e.g., Myerson (1981), an observation made previousl by **?**. The difference between these ordered subset mechanisms and the classic virtual valuation maximization mechanisms is that our ordered subset will come from solving an optimization on the whole auction problem where as the Myerson's virtual values come directly from single-agent optimizations.

Consider expressing an ex post allocation for type profile $\mathbf{t}$ by $\hat{x}^{\mathbf{t}} \in \{0,1\}^{T_N}$ as follows. For all $t'_i \in T_N$, $\hat{x}^{\mathbf{t}}(t'_i) = 1$ if player i is served and $t_i = t'_i$ and 0 otherwise. This definition of ex post allocations is convenient as the normalized interim allocation rule is calculated by taking its expecation, i.e., $\overline{x}(t'_i) = \mathbf{E}_{\mathbf{t}}\Big[\hat{x}^{\mathbf{t}}(t'_i)\Big]$. Ex post feasibility requires that,

$$\hat{x}^{\mathbf{t}}(S) \leq \min(|\mathbf{t} \cap S|, k), \qquad \forall \mathbf{t} \in \mathbf{T}, \forall S \subseteq T_N \tag{1}$$

In other words: for any profile of types $\mathbf{t}$, the number of types in S that are served by $\hat{x}^{\mathbf{t}}$ must be at most the number of types in S that showed up in $\mathbf{t}$ and the upper bound k. Taking expectations of both sides of this equation with respect to $\mathbf{t}$ motivates the following definition and theorem.

Definition 6. The *expected rank function* for distribution $\mathbf{f}$ and subspace $S \subset T_N$ is

$$g_k(S) = \mathbf{E}_{\mathbf{t}\sim\mathbf{f}}\Big[\min\big(|\mathbf{t} \cap S|, k\big)\Big] \tag{g_k}$$

where $\mathbf{t} \cap S$ denotes $\{t_1, \ldots, t_n\} \cap S$.

Theorem 3. *For supply constraint k and distribution $\mathbf{f}$, the space of all feasible normalized interim allocation rules, $\overline{\mathbb{X}}$, is the polymatroid associated with g_k, i.e., $\overline{\mathbb{X}} = P_{g_k}$, i.e., for all $\overline{x} \in \overline{\mathbb{X}}$,*

$$\overline{x}(S) \leq \mathbf{E}_{\mathbf{t}}\left[\min(|\mathbf{t} \cap S|, k)\right] = g_k(S), \qquad \forall S \subseteq T_N \tag{2}$$

The proof of this theorem will be deferred to the next section where we will derive a more general theorem. A key step in the proof will be relating the statement of the theorem to the polymatroid theory described already. To show that the constraint of the theorem is a polymatroid, we observe that the expected rank function is submoduler.

Lemma 3. *The expected rank function g_k is submodular.*

Proof. Observe that for any fixed $\mathbf{t}$, $\min(\mathbf{t} \cap S, k)$ is obviously a submodular function in S, and therefore g_k is a convex combination[9] of submodular functions, so g_k is submodular. □

We now relate vertices of the polymatroid to ordered subset allocation mechanisms.

Theorem 4. *For supply constraint k, if $\overline{x} \in \overline{\mathbb{X}}$ is the vertex* $\textsc{Vertex}(P_{g_k}, \pi)$ *of the polymatroid P_{g_k} the unique ex post implementation is the ordered subset mechanism induced by π (Definition 5).*

Proof. Let $\overline{x} = \textsc{Vertex}(P_{g_k}, \pi)$ be an arbitrary vertex of P_{g_k} with a corresponding ordered subset π; by Proposition 6, such a π exists for every vertex of a polymatroid. For every integer $r \leq |\pi|$, define $S^r = \{\pi_1, \ldots, \pi_r\}$ as the r-element prefix of the ordering. By Proposition 6, inequality (2) must be tight for every S^r which implies that inequality (1) must also be tight for every S^r and every $\mathbf{t} \in \mathbf{T}$. Observe that inequality (1) being tight for a subset S of types implies that any ex post allocation mechanism implementing $\overline{x}$ must allocate as much as possible to types in S. By definition, an ordered subset mechanism allocates to as many types as possible (up to k) from each S^r; this is the unique outcome given that inequality (1) is tight for every S^r. □

[9]Note that taking the expectation is the same as taking a convex combination.

Optimization over feasible interim allocation rules. The characterization of interim feasibility as a polymatroid constraint immediately enables efficient solving of optimization problems over the feasible interim allocation rules as long as we can compute g_k efficiently (see Schrijver (2003) for optimization over polymatroids). The following lemma states that g_k can be computed efficiently.

Lemma 4. *For independent agent (i.e., if* $\mathbf{f}$ *is a product distribution),* $g_k(S)$ *can be exactly computed in time* $O((n + |S|) \cdot k)$ *for any* $S \in T_N$ *using dynamic programming.*

Ex post implementation of feasible interim allocation rules. We now address the task of finding an ex post implementation corresponding to any $\overline{x} \in \overline{\mathbb{X}}$. By Theorem 7, if $\overline{x}$ is a vertex of $\overline{\mathbb{X}}$, it can be easily implemented by an ordered subset allocation mechanism (Definition 5). As any point in the polymatroid (or any convex polytope) can be specified as a convex combination of its vertices, to implement the corresponding interim allocation rule it is enough to show that this convex combination can be efficiently sampled from. An ex post implementation can then be obtained by sampling a vertex and using the ordered subset mechanism corresponding to that vertex. Instead of explicitly computing this convex combination, we present a general randomized rounding routine RandRound$(\cdot)$ which takes a point in a polymatroid and returns a vertex of the polymatroid such that the expected value of every coordinate of the returned vertex is the same as the original point. This approach is formally described next.

Definition 7 (randomized ordered subset allocation mechanism)**.** Parameterized by a normalized interim allocation rule $\overline{x} \in \overline{\mathbb{X}}$, a *randomized ordered subset allocation mechanism (RRA)* computes the allocation for a profile of types $\mathbf{t} \in \mathbf{T}$ as follows.

1. Let $(\overline{x}^*, \pi^*) \leftarrow$ RandRound$(\overline{x})$.
2. Run the ordered subset mechanism (Definition 5) with ordered subset π^*.

Theorem 5. *Any normalized interim allocation rule* $\overline{x} \in \overline{\mathbb{X}}$ *can be implemented by the randomized ordered subset allocation mechanism (Definition 7) as a distribution over deterministic ordered subset allocation mechanisms.*

Proof. The proof follows from linearity of expectation. □

Randomized rounding for polymatroids. We describe RandRound$(\cdot)$ for general polymatroids. First, we present a few definitions and give an overview of the rounding operator. Consider an arbitrary finite set U and a polymatroid $P_{\mathcal{F}}$ associated with a non-decreasing submodular function $\mathcal{F} : 2^U \to \mathbb{R}_+$ with $\mathcal{F}(\emptyset) = 0$. A set $S \subseteq U$ is called *tight* with respect to a $y \in P_{\mathcal{F}}$, if and only if $y(S) = \mathcal{F}(S)$. A set $\mathbb{S} = \{S^0, \ldots, S^m\}$ of subsets of U is called a *nested family of tight sets* with respect to $y \in P_{\mathcal{F}}$, if and only the elements of $\mathbb{S}$ can be or ordered and indexed such that $\emptyset = S^0 \subset \cdots \subset S^m \subseteq U$, and such that S^r is tight with respect to y (for every $r \in [m]$).

RandRound(y) takes an arbitrary $y \in P_{\mathcal{F}}$ and iteratively makes small changes to it until a vertex is reached. At each iteration ℓ, it computes $y^\ell \in P_{\mathcal{F}}$, and a nested family of tight sets $\mathbb{S}^\ell$ (with respect to y^ℓ) such that

- $\mathbf{E}[y^\ell | y^{\ell-1}] = y^{\ell-1}$, and

- y^ℓ is closer to a vertex (compared to $y^{\ell-1}$) in the sense that either the number of non-zero coordinates has decreased by one or the number of tight sets has increased by one.

Observe that the above process must stop after at most $2|U|$ iterations[10]. At each iteration ℓ of the rounding process, a vector $\hat{y} \in \mathbb{R}^U$ and $\delta, \delta' \in \mathbb{R}_+$ are computed such that both $y^{\ell-1} + \delta \cdot \hat{y}$ and $y^{\ell-1} - \delta' \cdot \hat{y}$ are still in $P_{\mathcal{F}}$, but closer to a vertex. The algorithm then chooses a random $\delta'' \in \{\delta, -\delta'\}$ such that $\mathbf{E}[\delta''] = 0$, and sets $y^\ell \leftarrow y^{\ell-1} + \delta'' \cdot \hat{y}$.

Definition 8 (RandRound(y))**.** This operator takes as its input a point $y \in P_{\mathcal{F}}$ and returns as its output a pair (y^*, π^*), where y^* is a random vertex of $P_{\mathcal{F}}$ and π^* is its associated ordered subset (see Proposition 6), and such that $\mathbf{E}[y^*] = y$.

The algorithm modifies y iteratively until a vertex is reached. It also maintains a nested family of tight sets $\$$ with respect to y. As we modify $\$$, we always maintain an ordered labeling of its elements, i.e., if $\$ = \{S^0, \dots, S^m\}$, we assume that $\emptyset = S^0 \subset \cdots \subset S^m \subseteq U$; in particular, the indices are updated whenever a new tight set is added. For each $s \in U$, define $\mathbf{1}_s \in [0,1]^U$ as a vector whose value is 1 at coordinate s and 0 everywhere else.

1. Initialize $\$ \leftarrow \{\emptyset\}$.
2. Repeat each of the following steps until no longer applicable:
   - If there exist distinct $s, s' \in S^r \setminus S^{r-1}$ for some $r \in [m]$:
     (a) Set $\hat{y} \leftarrow \mathbf{1}_s - \mathbf{1}_{s'}$, and compute $\delta, \delta' \in \mathbb{R}_+$ such that $y + \delta \cdot \hat{y}$ has a new tight set S and $y - \delta' \cdot \hat{y}$ has a new tight set S', i.e,
       – set $S \leftarrow \arg\min_{S^{r-1}+s \subseteq S \subseteq S^r - s'} \mathcal{F}(S) - y(S)$, and $\delta \leftarrow \mathcal{F}(S) - y(S)$;
       – set $S' \leftarrow \arg\min_{S^{r-1}+s' \subseteq S' \subseteq S^r - s} \mathcal{F}(S') - y(S')$, and $\delta' \leftarrow \mathcal{F}(S') - y(S')$.
     (b) $\begin{cases} \text{with prob. } \frac{\delta}{\delta+\delta'}: & \text{set } y \leftarrow y + \delta \cdot \hat{y}, \text{ and add } S \text{ to } \$. \\ \text{with prob. } \frac{\delta'}{\delta+\delta'}: & \text{set } y \leftarrow y - \delta' \cdot \hat{y}, \text{ and add } S' \text{ to } \$. \end{cases}$
   - If there exists $s \in U \setminus S^m$ for which $y(s) > 0$:
     (a) Set $\hat{y} \leftarrow \mathbf{1}_s$, and compute $\delta, \delta' \in \mathbb{R}_+$ such that $y + \delta \cdot \hat{y}$ has a new tight set S and $y - \delta' \cdot \hat{y}$ has a zero at coordinate s, i.e,
       – set $S \leftarrow \arg\min_{S \supseteq S^m + s} \mathcal{F}(S) - y(S)$, and $\delta \leftarrow \mathcal{F}(S) - y(S)$;
       – set $\delta' \leftarrow y(s)$.
     (b) $\begin{cases} \text{with prob. } \frac{\delta}{\delta+\delta'}: & \text{set } y \leftarrow y + \delta \cdot \hat{y}, \text{ and add } S \text{ to } \$. \\ \text{with prob. } \frac{\delta'}{\delta+\delta'}: & \text{set } y \leftarrow y - \delta' \cdot \hat{y} \end{cases}$
3. Set $y^* \leftarrow y$ and define the ordered subset $\pi^* : S^m \to [m]$ according to $\$$, i.e., for each $r \in [m]$ and $s \in S^r \setminus S^{r-1}$, define $\pi^*(s) = r$.
4. Return (y^*, π^*).

Theorem 6. *For any non-decreasing submodular function $\mathcal{F} : 2^U \to \mathbb{R}_+$ and any $y \in P_{\mathcal{F}}$, the operator* RandRound(y) *returns a random (y^*, π^*) such that $y^* \in$* Vertex*$(P_{\mathcal{F}})$, and π^* is the ordered subset corresponding to y^* (see Proposition 6), and such that $\mathbf{E}[y^*] = y$. Furthermore, the algorithm runs in strong polynomial time. In particular, it runs for $O(|U|)$ iterations where each iteration involves solving two submodular minimizations.*

[10]In fact we will show that it stops after at most $|U|$ iterations.

5.3 Matroid Feasibility Constraints

In this section, we consider environments where the feasibility constraints are encoded by a matroid $\mathcal{M} = (T_N, \mathcal{I})$. For every type profile $\mathbf{t} \in \mathbf{T}$, a subset $S \subseteq \{t_1, \ldots, t_n\}$ can be simultaneously allocated to if and only if $S \in \mathcal{I}$. We show that the results of subsection 5.2 can be easily generalized to environments with matroid feasibility constraints.

Matroid Preliminaries. A matroid $\mathcal{M} = (U, \mathcal{I})$ consists of a ground set U and a family of independent sets $\mathcal{I} \subseteq 2^U$ with the following two properties.

- For every I, I', if $I' \subset I \in \mathcal{I}$, then $I' \in \mathcal{I}$.
- For every $I, I' \in \mathcal{I}$, if $|I'| < |I|$, there exists $s \in I \setminus I'$ such that $I' \cup \{s\} \in \mathcal{I}$.

For every matroid $\mathcal{M}$, the rank function $r_{\mathcal{M}} : 2^U \to \mathbb{N} \cup \{0\}$ is defined as follows: for each $S \subseteq U$, $r_{\mathcal{M}}(S)$ is the size of the maximum independent subset of S. A matroid can be uniquely characterized by its rank function, i.e., a set $I \subseteq U$ is an independent set if and only if $r_{\mathcal{M}}(I) = |I|$. A matroid rank function has the following two properties:

- $r_{\mathcal{M}}(\cdot)$ is a non-negative non-decreasing integral submodular function.
- $r_{\mathcal{M}}(S) \leq |S|$, for all $S \subseteq U$.

Furthermore, every function with the above properties defines a matroid.

Any set $S \subseteq U$ can be equivalently represented by its incidence vector $\chi^S \in \{0,1\}^U$ which has a 1 at every coordinate $s \in S$ and 0 everywhere else.

Proposition 7. *Consider an arbitrary finite matroid $\mathcal{M} = (U, \mathcal{I})$ with rank function $r_{\mathcal{M}}(\cdot)$. Let $P_{r_{\mathcal{M}}}$ denote the polymatroid associated with $r_{\mathcal{M}}(\cdot)$ (see subsection 5.2); the vertices of $P_{r_{\mathcal{M}}}$ are exactly the incidence vectors of the independent sets of $\mathcal{M}$.*

See Schrijver (2003) for a comprehensive treatment of matroids.

Characterization of interim feasibility. We now generalize the characterization of interim feasibility as the polymatroid given by the expected rank of the matroid. From this generalization the computational results of the preceeding section can be extended from k-unit environments to matroids.

Let b denote the random bits used by an ex post allocation rule, and let $\hat{x}^{\mathbf{t},b} \in \{0,1\}^{T_N}$ denote the ex post allocation rule (i.e., the incidence vector of the subset of types that get allocated to) for type profile $\mathbf{t} \in \mathbf{T}$ and random bits b. It is easy to see that $\hat{x}^{\mathbf{t},b}$ is a feasible ex post allocation if and only if it satisfies the following class of inequalities.

$$\hat{x}^{\mathbf{t},b}(S) \leq r_{\mathcal{M}}(\mathbf{t} \cap S), \qquad \forall \mathbf{t} \in \mathbf{T}, \forall S \subseteq T_N \tag{3}$$

The above inequality states that the subset of types that get allocated to must be an independent set of the restriction of matroid $\mathcal{M}$ to $\{t_1, \ldots, t_n\}$. The expectation of the left-hand-side is exactly the normalized interim allocation rule, i.e., for any $t'_i \in T_N$, $\overline{x}(t'_i) = \mathbf{E}_{\mathbf{t},b}\left[\hat{x}^{\mathbf{t},b}(t'_i)\right]$. Taking expectations of both sides of (3) then motivates the following definition and theorem that characterize interim feasibility.

Definition 9. The *expected rank* for distribudion $\mathbf{f}$, subspace $S \subset T_N$, and matroid $\mathcal{M}$ with rank function $r_{\mathcal{M}}$ is

$$g_{\mathcal{M}}(S) = \mathbf{E}_{\mathbf{t}\sim\mathbf{f}}\left[r_{\mathcal{M}}(\mathbf{t} \cap S)\right] \tag{$g_{\mathcal{M}}$}$$

where $\mathbf{t} \cap S$ denotes $\{t_1, \ldots, t_n\} \cap S$.

Theorem 7. *For matroid $\mathcal{M}$ and distribution $\mathbf{f}$, the space of all feasible normalized interim allocation rules, $\overline{\mathbb{X}}$, is the polymatroid associated with g_k, i.e., $\overline{\mathbb{X}} = P_{g_{\mathcal{M}}}$, i.e., for all $\overline{x} \in \overline{\mathbb{X}}$,*

$$\overline{x}(S) \leq \mathbf{E}_{\mathbf{t}}\left[r_{\mathcal{M}}(\mathbf{t} \cap S)\right] = g_{\mathcal{M}}(S), \qquad \forall S \subseteq T_N \tag{4}$$

Theorem 8. *For matroid $\mathcal{M}$, if $\overline{x} \in \overline{\mathbb{X}}$ is the vertex* $\textsc{Vertex}(P_{g_{\mathcal{M}}}, \pi)$ *of the polymatroid $P_{g_{\mathcal{M}}}$ the unique ex post implementation is the ordered subset mechanism induced by π (Definition 5).*

To prove the above theorems, we use the following decomposition lemma which applies to general polymatroids.

Lemma 5 (Polymatroidal Decomposition)**.** *Let U be an arbitrary finite set, $\mathcal{F}^1, \ldots, \mathcal{F}^m : 2^U \to \mathbb{R}_+$ be arbitrary non-decreasing submodular functions, and $\mathcal{F}^* = \sum_{j=1}^m \lambda^j \mathcal{F}^j$ be an arbitrary convex combination of them. For every y^* the following holds: $y^* \in P_{\mathcal{F}^*}$ if and only if it can be decomposed as $y^* = \sum_{j=1}^m \lambda^j y^j$ such that $y^j \in P_{\mathcal{F}^j}$ (for each $j \in [m]$). Furthermore, if y^* is a vertex of $P_{\mathcal{F}^*}$, this decomposition is unique. More precisely, if $y^* =$ $\textsc{Vertex}(P_{\mathcal{F}^*}, \pi)$ for some ordered subset π, then $y^j =$ $\textsc{Vertex}(P_{\mathcal{F}^j}, \pi)$ (for each $j \in [m]$).*

Proof. First, observe that the only-if part is obviously true, i.e., if $y^j \in P_{\mathcal{F}^j}$ (for each $j \in [m]$), we can write

$$y^j(S) \leq \mathcal{F}^j(S) \qquad \forall S \subseteq U, \tag{5}$$

multiplying both sides by λ^j and summing over all $j \in [m]$ we obtain

$$y^*(S) = \sum_{i=1}^m \lambda^j y^j(S) \leq \sum_{j=1}^m \lambda^j \mathcal{F}^j(S) = \mathcal{F}^*(S) \qquad \forall S \subseteq U, \tag{6}$$

which implies that $y^* \in P_{\mathcal{F}^*}$.

Next, we prove that for every $y^* \in P_{\mathcal{F}^*}$ such a decomposition exists. Note that a polymatroid is a convex polytope, so any $y^* \in P_{\mathcal{F}^*}$ can be written as a convex combination of vertices as $y^* = \sum_\ell \alpha^\ell y^{*\ell}$, where each $y^{*\ell}$ is a vertex of $P_{\mathcal{F}^*}$; consequently, if we prove the claim for the vertices of $P_{\mathcal{F}^*}$, i.e., that $y^{*\ell} = \sum_{j=1}^m \lambda^j y^{\ell,j}$ for some $y^{\ell,j} \in P_{\mathcal{F}^j}$, then a decomposition of $y^* = \sum_{j=1}^m \lambda^j y^j$ can be obtained by setting $y^j = \sum_\ell \alpha^\ell y^{\ell,j}$.

Next, we prove the second part of the theorem which also implies that a decomposition exists for every vertex of $P_{\mathcal{F}^*}$. Let $y^* =$ $\textsc{Vertex}(P_{\mathcal{F}^*}, \pi)$ be an arbitrary vertex of $P_{\mathcal{F}^*}$ with a corresponding ordered subset π; by Proposition 6, such a π exists for every vertex of a polymatroid. For every integer $r \leq |\pi|$, define $S^r = \{\pi_1, \ldots, \pi_r\}$ as the r-element prefix of the ordering. By Proposition 6, inequality (6) is tight for every S^r, which implies that inequality (5) must also be tight for each S^r and for every $j \in [m]$. Consequently, for each $r \in [|\pi|]$, and each $j \in [m]$, by taking the difference of the inequality (6) for S^r and S^{r-1}, given that they are tight, we obtain

$$y^j(s) = \mathcal{F}^j(S^r) - \mathcal{F}^j(S^{r-1})$$

Furthermore, for each $s \notin \pi$, $y^*(s) = 0$ which implies that $y^j(s) = 0$ for every $j \in [m]$. Observe that we have obtained a unique y^j for each $j \in [m]$ which is exactly the vertex of $P_{\mathcal{F}^j}$ corresponding to π as described in Proposition 6. It is easy to verify that indeed $y^* = \sum_{j=1}^m \lambda^j y^j$. □

Proof of Theorem 7. The inequality in equation (3) states that the subset of types that get allocated to must be an independent set of the restriction of matroid $\mathcal{M}$ to $\{t_1, \dots, t_n\}$. Define $r^{\mathbf{t}}_{\mathcal{M}}(S) = r_{\mathcal{M}}(\mathbf{t} \cap S)$ (for all $(S \subseteq T_N)$. Notice that $r^{\mathbf{t}}_{\mathcal{M}}$ is a submodular function. The above inequality implies that $\hat{x}^{\mathbf{t},b} \in P_{r^{\mathbf{t}}_{\mathcal{M}}}$. Define $\hat{x}^{\mathbf{t}} = \mathbf{E}_b[\hat{x}^{\mathbf{t},b}]$. Observe that $\overline{x} = \mathbf{E}_{\mathbf{t}}[\hat{x}^{\mathbf{t}}]$, so $\overline{x}$ is a feasible normalized interim allocation rule if and only if it can be decomposed as $\overline{x} = \sum_{\mathbf{t} \in \mathbf{T}} f(\mathbf{t}) \hat{x}^{\mathbf{t}}$ where $\hat{x}^{\mathbf{t}} \in P_{r^{\mathbf{t}}_{\mathcal{M}}}$ for every $\mathbf{t} \in \mathbf{T}$; by Lemma 5, this is equivalent to $\overline{x} \in P_{g_{\mathcal{M}}}$ where $g_{\mathcal{M}}(S) = \sum_{\mathbf{t} \in \mathbf{T}} f(\mathbf{t}) r^{\mathbf{t}}_{\mathcal{M}}(S) = \mathbf{E}_{\mathbf{t}}[r^{\mathbf{t}}_{\mathcal{M}}(S)]$ (for all $S \subseteq T_N$), as defined in Definition 9. That completes the proof of the first part of the theorem. □

Proof of Theorem 8. Suppose $\overline{x} = \textsc{Vertex}(P_{g_{\mathcal{M}}}, \pi)$ for some ordered subset π. By Lemma 5, the decomposition of $\overline{x}$ is unique and is given by $\hat{x}^{\mathbf{t}} = \textsc{Vertex}(P_{r^{\mathbf{t}}_{\mathcal{M}}}, \pi)$. Notice that this is the same allocation obtained by the deterministic rank-based allocation mechanism which ranks according to π (see Definition 5). □

Optimization over feasible interim allocation rules. As in subsection 5.2, the characterization of interim feasibility as a polymatroid constraint immediately enables efficient solving of optimization problems over the feasible normalized interim allocation rules as long as we can compute $g_{\mathcal{M}}$ efficiently (see Schrijver (2003) for optimization over polymatroids). Depending on the specific matroid, it might be possible to exactly compute $g_{\mathcal{M}}$ in polynomial time (e.g., as in Lemma 4); otherwise, it can be computed approximately within a factor of $1 - \epsilon$ and with probability $1 - \delta$, by sampling, in time polynomial in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$.

Ex post implementation of feasible interim allocation rules. An ex post implementation for any $\overline{x} \in \overline{\mathbb{X}}$ can be obtained exactly as in subsection 5.2.

Corollary 3. *Any normalized interim allocation rule $\overline{x} \in \overline{\mathbb{X}}$ can be implemented by the randomized rank-based allocation mechanism (Definition 7) as a distribution over deterministic rank-based allocation mechanisms (Definition 5).*

6 Conclusions and Extensions

In this paper we have focused on binary allocation problems where an agent is either served or not served. For these binary allocation problems distributions over allocations are given by a single number, i.e., the probability that the agent is served. Our results can be extended to environments with multi-unit demand when the agents utility is linear in the expected number of units the agent receives.

In Section 5 we described algorithms for optimizing over feasible interim allocation rules and for (ex post) implementation of the resulting rules. Neither these algorithms nor the generalization of Border's condition require the types of the agents to be independently distributed. However, our formulation of incentive compatibility for interim allocation rules does require independence. For correlated distributions the interim allocation rule is a function of the actual type of the agent (which conditions the types of the other agents) and the reported type of the agent. Therefore, this generalization of our theorem to correlated environments has little relevance for mechanism design.

The algorithms in Section 5 do not require the feasibility constraint to be known in advance. A simple example where this generalization is interesting is a multi-unit auction where the supply k is stochastically drawn from a known distribution. Our result shows that the optimal auction in such an environment can be described by picking the random ordering on types and allocating greedily by this ordering while supplies last. We do not know of many examples other than this where this generalization is interesting.

Our techniques can also be used in conjunction with the approach of Cai et al. (2012) for solving multi-item auction problems for agents with additive values.

One important extension of our work is to scenarios where the type space and distribution are only available via oracle access (and can be very large or even infinite). Given a polynomial time approximation scheme (PTAS) for a variant of the single agent problem we can construct a polynomial time approximation scheme for the multi-agent problem. Such a model is important, for instance, when the agents type space is multi-dimensional but succinctly describable, e.g., for unit demand agents with independently distributed values for various items for sale. Such a type space would be exponentially large in the number of items but succinctly described in polynomial space in the number of items. While reduction can be applied to this scenario; however, we do not know of any solution to the optimal single-agent problem with which to instantiate the reduction.

Finally, in symmetric environments, i.e., when the agents' type distribution and the designer's feasibility constraint are symmetric, e.g., for i.i.d. multi-unit auctions, the optimal interim allocation constraint imposed by feasibility is symmetric; furthermore, the constraint is given by a simple formula. Therefore, symmetric multi-agent problems reduce to solving the single-agent problem for a very specific constraint on the interim allocation rule. In comparison to the computational task of optimizing over feasible interim allocation rules and solving for ex post implementations, e.g., from Section 5, this multi-agent reduction for symmetric environments is computationally trivial.

References

Alaei, S. (2011). Bayesian combinatorial auctions: Expanding single buyer mechanisms to many buyers. In *IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 512–521.

Armstrong, M. (1996). Multiproduct nonlinear pricing. *Econometrica*, 64(1):51–75.

Arrow, K. J. and Debreu, G. (1954). Existence of an equilibrium for a competitive economy. *Econometrica*, 22(3):pp. 265–290.

Bhattacharya, S., Goel, G., Gollapudi, S., and Munagala, K. (2010). Budget constrained auctions with heterogeneous items. In *44th ACM Symposium on Theory of Computing*, pages 379–388.

Border, K. (2007). Reduced form auctions revisited. *Economic Theory*, 31(1):167–181.

Border, K. C. (1991). Implementation of reduced form auctions: A geometric approach. *Econometrica*, 59(4):pp. 1175–1187.

Briest, P., Chawla, S., Kleinberg, R., and Weinberg, S. M. (2010). Pricing randomized allocations. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 585–597.

Bulow, J. and Roberts, J. (1989). The simple economics of optimal auctions. *Journal of Political Economy*, 97(5):1060–90.

Cai, Y. and Daskalakis, C. (2011). Extreme-value theorems for optimal multidimensional pricing. In *IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 522–531.

Cai, Y., Daskalakis, C., and Weinberg, M. (2012). An algorithmic characterization of multi-dimensional mechanisms.

Chawla, S., Hartline, J. D., and Kleinberg, R. D. (2007). Algorithmic pricing via virtual valuations. In MacKie-Mason, J. K., Parkes, D. C., and Resnick, P., editors, *ACM Conference on Electronic Commerce*, pages 243–251. ACM.

Chawla, S., Hartline, J. D., Malec, D. L., and Sivan, B. (2010). Multi-parameter mechanism design and sequential posted pricing. In *42nd ACM Symposium on Theory of Computing*, pages 311–320.

Che, Y. and Gale, I. (1995). The optimal mechanism for selling to budget-constrained consumers. Technical report.

Chen, X. and Deng, X. (2006). Settling the complexity of two-player nash equilibrium. In *IEEE 47th Annual Symposium on Foundations of Computer Science*, pages 261–272.

Daskalakis, C., Goldberg, P. W., and Papadimitriou, C. H. (2009). The complexity of computing a nash equilibrium. *SIAM J. Comput.*, 39(1):195–259.

Devanur, N. R., Papadimitriou, C. H., Saberi, A., and Vazirani, V. V. (2008). Market equilibrium via a primal–dual algorithm for a convex program. *J. ACM*, 55(5).

Jain, K. (2004). A polynomial time algorithm for computing the arrow-debreu market equilibrium for linear utilities. In *IEEE 45th Annual Symposium on Foundations of Computer Science*, pages 286–294.

Laffont, J.-J. and Robert, J. (1996). Optimal auction with financially constrained buyers. *Economics Letters*, 52(2):181–186.

Manelli, A. M. and Vincent, D. R. (2010). Bayesian and dominant-strategy implementation in the independent private-values model. *Econometrica*, 78(6):1905–1938.

Maskin, E. and Riley, J. (1984). Optimal auctions with risk averse buyers. *Econometrica*, 52(6):pp. 1473–1518.

Maskin, E. S. (2000). Auctions, development, and privatization: Efficient auctions with liquidity-constrained buyers. *European Economic Review*, 44(4-6):667–681.

Matthews, S. (1983). Selling to risk averse buyers with unobservable tastes. *Journal of Economic Theory*, 30(2):370–400.

Matthews, S. A. (1984). On the implementability of reduced form auctions. *Econometrica*, 52(6):pp. 1519–1522.

McAfee, R. P. and McMillan, J. (1988). Multidimensional incentive compatibility and mechanism design. *Journal of Economic Theory*, 46(2):335–354.

Mierendorff, K. (2011). Asymmetric reduced form auctions. *Economics Letters*, 110(1):41–44.

Mirman, L. J. and Sibley, D. (1980). Optimal nonlinear prices for multiproduct monopolies. *Bell Journal of Economics*, 11(2):659–670.

Myerson, R. B. (1981). Optimal auction design. *Mathematics of Operations Research*, 6(1):pp. 58–73.

Nash, J. (1951). Non-cooperative games. *The Annals of Mathematics*, 54(2):pp. 286–295.

Pai, M. M. and Vohra, R. (2008). Optimal auctions with financially constrained bidders. Discussion papers, Northwestern University, Center for Mathematical Studies in Economics and Management Science.

Roberts, K. W. S. (1979). Welfare considerations of nonlinear pricing. *Economic Journal*, 89(353):66–83.

Rochet, J.-C. and Chone, P. (1998). Ironing, sweeping, and multidimensional screening. *Econometrica*, 66(4):783–826.

Schrijver, A. (2003). *Combinatorial Optimization : Polyhedra and Efficiency (Algorithms and Combinatorics)*. Springer.

Spence, A. M. (1980). Multi-product quantity-dependent prices and profitability constraints. *Review of Economic Studies*, 47(5):821–41.

Wilson, R. (1994). Nonlinear pricing. *Journal of Political Economy*, 102(6):1288–1291.

A Proofs from Section 5.1

We first describe a network flow formulation of $\mathbb{S}$, which is used to prove Lemma 1 and Lemma 2.

A network flow formulation of $\mathbb{S}$. We construct a network in which every feasible flow corresponds to some $(y, z) \in \mathbb{S}$. The network (see Figure 1) has a source node $\langle \textsc{Src} \rangle$, a sink node $\langle \textsc{Snk} \rangle$, and $n - i + 1$ nodes for every $t_i \in T_N$ labeled as $\langle t_i, i \rangle, \cdots, \langle t_i, n \rangle$ where each node $\langle t_{i'}, i \rangle$ corresponds to the type $t_{i'}$ at the time SSA algorithm is visiting agent i. For each $t_{i'} \in T_N$ and for each $i \in \{i', \ldots, n-1\}$ there is an edge $(\langle t_{i'}, i \rangle, \langle t_{i'}, i+1 \rangle)$ with infinite capacity whose flow is equal to $y(t_{i'}, i)$; we refer to these edges as *"horizontal edges"*. For every $t_{i'}$ and every t_i where $i' < i$ there is an edge $(\langle t_{i'}, i \rangle, \langle t_i, i \rangle)$ whose flow is equal to $z(t_{i'}, t_i)$ and whose capacity is equal to the total amount of flow that enters $\langle t_{i'}, i \rangle$ multiplied by $f_i(t_i)$, i.e., it has a dynamic capacity which is equal to $y(t_{i'}, i-1) f_i(t_i)$; we refer to these edges as *"diagonal edges"*. There is an edge $(\langle \textsc{Src} \rangle, t_0)$ through which the source node pushes exactly one unit of flow. Finally, for every $t_i \in T_N$, there is an edge $(\langle t_i, n \rangle, \langle \textsc{Snk} \rangle)$ with unlimited capacity whose flow is equal to $y(t_i, n)$. To simplify the proofs we sometimes use $\langle t_0, 0 \rangle$ as an alias for the source node $\langle \textsc{Src} \rangle$ and $\langle t_i, n+1 \rangle$ as aliases for the sink node $\langle \textsc{Snk} \rangle$. The network always has a feasible flow because all the flow can be routed along the path $\langle \textsc{Src} \rangle, \langle t_0, 1 \rangle, \ldots, \langle t_0, n \rangle, \langle \textsc{Snk} \rangle$.

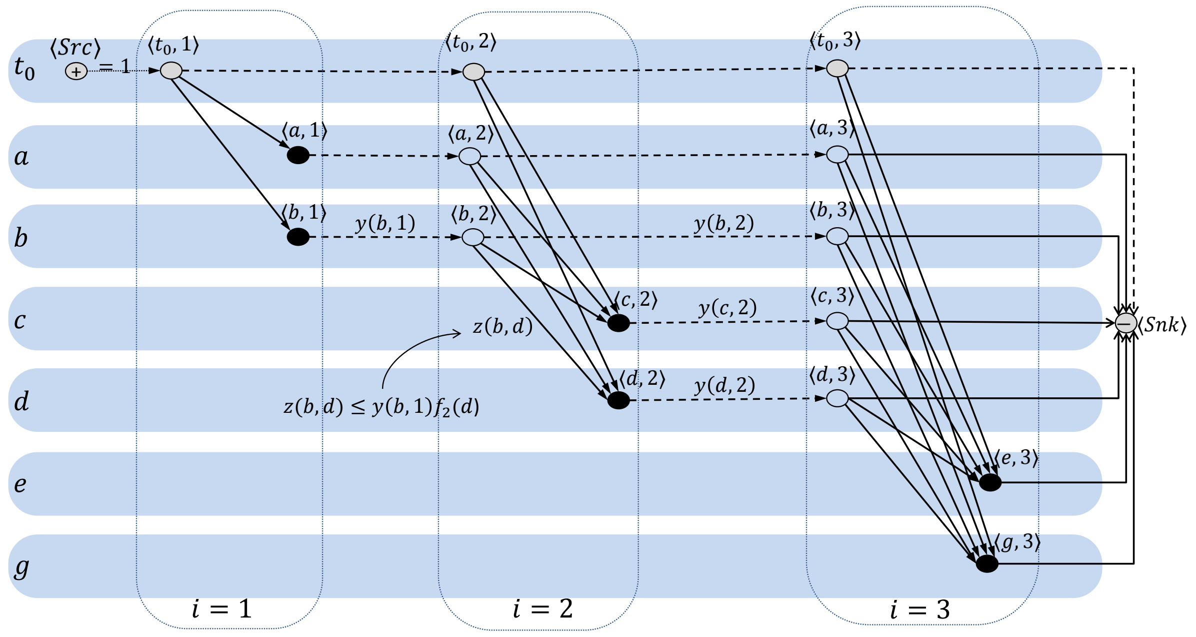

Figure 1: The flow network corresponding to the SSA algorithm 3. In this instance, there are three agents with type spaces $T_1 = \{a, b\}$, $T_2 = \{c, d\}$, and $T_3 = \{e, g\}$. All nodes in the same row correspond to the same type. The diagonal edges have dynamic capacity constraints while all other edges have no capacity constraints. The flow going from $\langle t_{i'}, i \rangle$ to $\langle t_i, i \rangle$ corresponds to the ex-ante probability of t_i taking the token away from $t_{i'}$. The flow going from $\langle t_{i'}, i \rangle$ to $\langle t_{i'}, i + 1 \rangle$ corresponds to the ex-ante probability of $t_{i'}$ still holding the token after agent i is visited.

We define the *residual capacity* between two types $t_{i'}, t_i \in T_N$ with respect to a given $(y, z) \in \mathbb{S}$ as follows.

$$\textsc{ResCap}_{y,z}(t_{i'}, t_i) = \begin{cases} y(t_{i'}, i-1)f_i(t_i) - z(t_{i'}, t_i) & i > i' \\ z(t_i, t_{i'}) & i < i' \\ 0 & \text{otherwise} \end{cases} \tag{ResCap}$$

Due to dynamic capacity constraints, it is not possible to augment a flow along a path with positive residual capacity by simply changing the amount of the flow along the edges of the path, because reducing the total flow entering a node also decreases the capacity of the diagonal edges leaving that node, which could potentially violate their capacity constraints. Therefore, we introduce an operator $\textsc{Reroute}(t_{i'}, t_i, \rho)$ (algorithm 1 and Figure 2) which modifies an existing $(y, z) \in \mathbb{S}$, while maintaining its feasibility, to transfer a ρ-fraction of $y(t_i, n)$ to $y(t_{i'}, n)$ by changing the flow along the cycle

$$\langle \textsc{Snk} \rangle, \langle t_{i'}, n \rangle, \langle t_{i'}, n-1 \rangle, \ldots, \langle t_{i'}, \max(i', i) \rangle, \langle t_i, \max(i', i) \rangle, \ldots, \langle t_i, n-1 \rangle, \langle t_i, n \rangle, \langle \textsc{Snk} \rangle$$

and adjusting the flow of the the diagonal edges which leave this cycle. More precisely, $\textsc{Reroute}(t_{i'}, t_i, \rho)$ takes out a ρ-fraction of the flow going through the subtree rooted at $\langle t_{i'}, \max(i', i) \rangle$ [11] and reassigns it to the subtree rooted at $\langle t_i, \max(i', i) \rangle$ (see Figure 2).

Algorithm 1 $\textsc{Reroute}(t_{i'}, t_i, \rho)$.

Input: An existing $(y, z) \in \mathbb{S}$ given implicitly, a source type $t_{i'} \in T_N$, a destination type $t_i \in T_N$ where $i' \neq i$, and a fraction $\rho \in [0, 1]$.

Output: Modify (y, z) to transfer a ρ-fraction of $y(t_{i'}, n)$ to $y(t_i, n)$ while ensuring that the modified assignment is still in $\mathbb{S}$.

```
1:  if $i' < i$ then
2:      Increase $z(t_{i'}, t_i)$ by $\rho \cdot y(t_{i'}, i)$.
3:  else
4:      Decrease $z(t_i, t_{i'})$ by $\rho \cdot y(t_{i'}, i')$.
5:  end if
6:  for $i'' = \max(i', i)$ to $n$ do
7:      Increase $y(t_i, i'')$ by $\rho \cdot y(t_{i'}, i'')$.
8:      Decrease $y(t_{i'}, i'')$ by $\rho \cdot y(t_{i'}, i'')$.
9:  end for
10: for $t_{i''} \in T_{\{\max(i',i)+1,\ldots,n\}}$ do
11:     Increase $z(t_i, t_{i''})$ by $\rho \cdot z(t_{i'}, t_{i''})$.
12:     Decrease $z(t_{i'}, t_{i''})$ by $\rho \cdot z(t_{i'}, t_{i''})$.
13: end for
```

Proof of Lemma 1. For any given $(y, z) \in \mathbb{S}$ we show that it is always possible to modify y and z to obtain a non-degenerate feasible assignment with the same induced interim allocation probabilities

[11] This subtree consists of the path $\langle t_{i'}, \max(i', i) \rangle, \ldots, \langle t_{i'}, n \rangle, \langle \textsc{Snk} \rangle$ and all the diagonal edges leaving this path.

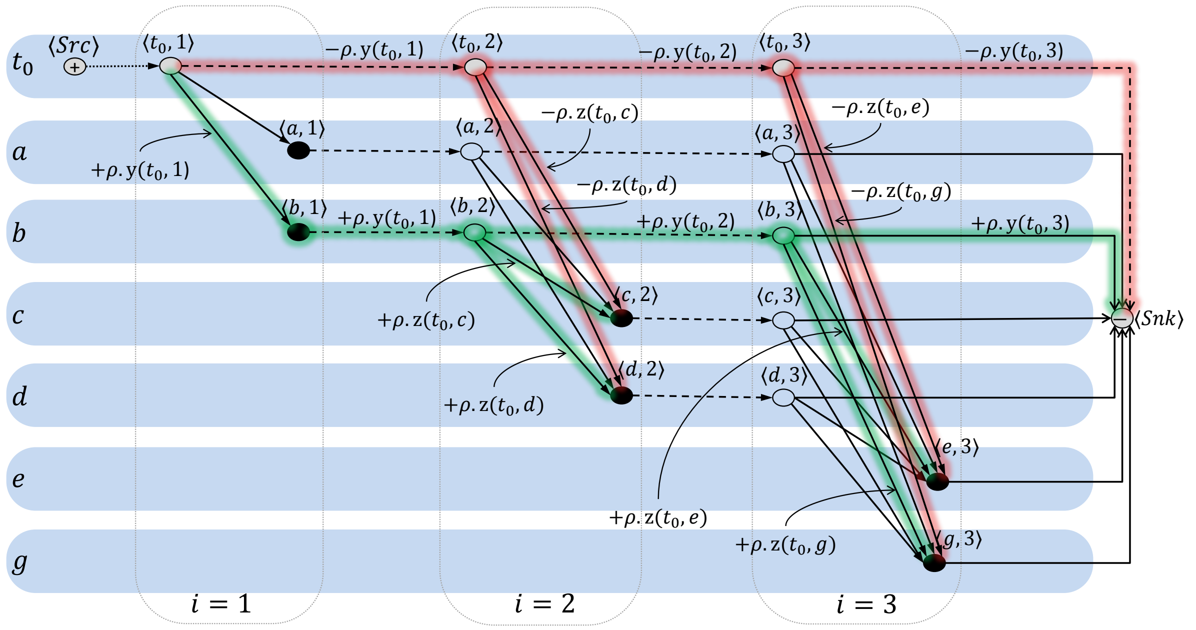

Figure 2: Changes made by applying $\text{Reroute}(t_0, b, \rho)$. A ρ-fraction of the red subtree rooted at t_0 is take out and reassigned to the green subtree rooted at b. The exact amount of change is indicated for each green and each red edge. The flow along all other edges stay intact. The operator has the effect of reassigning ρ-fraction of ex-ante probability of allocation for type t_0 to type b.

(i.e., the same $y(\cdot, n)$). Let d denote the number of degenerate types with respect to (y, z), i.e., define

$$d = \#\left\{t_i \in T_{\{1,\ldots,n\}} \middle| y(t_i, n) = 0, y(t_i, i) > 0\right\}$$

The proof is by induction on d. The base case is $d = 0$ which is trivial. We prove the claim for $d > 0$ by modifying y and z, reducing the number of degenerate types to $d-1$, and then applying the induction hypothesis. Let t_i be a degenerate type. For each $t_{i'} \in T_{\{0,\ldots,i-1\}}$, we apply the operator $\text{Reroute}(t_i, t_{i'}, \frac{z(t_{i'},t_i)}{y(t_i,i)})$ unless $y(t_i, i)$ has already reached 0. Applying this operator to each type $t_{i'}$ eliminates the flow from $\langle t_{i'}, i\rangle$ to $\langle t_i, i\rangle$, so eventually $y(t_i, i)$ reaches 0 and t_i is no longer degenerate and also no new degenerate type is introduced, so the number of degenerate types is reduced to $d-1$. It is also easy to see that $y(t_{i'}, n)$ is not modified because $y(t_i, n) = 0$. That completes the proof. $\square$

Proof of Lemma 2. To prove the lemma it is enough to show that for any augmentable type $t_{i'}$ and any non-augmentable type t_i, $\text{ResCap}_{y,z}(t_{i'}, t_i) = 0$ which is equivalent to the statement of the lemma (the equivalence follows from the definition of ResCap and equation (π)). The proof is by contradiction. Suppose $t_{i'}$ is augmentable and $\text{ResCap}_{y,z}(t_{i'}, t_i) = \delta$ for some positive δ; we show that t_i is also augmentable. Since $t_{i'}$ is augmentable, there exists a $(y', z') \in \mathbb{S}$ such that $y'(\tau, n) = y(\tau, n)$ for all $\tau \in T_N \setminus \{t_0, t_{i'}\}$ and $y'(t_{i'}, n) - y(t_{i'}, n) = \epsilon > 0$. Define

$$(y'', z'') = (1-\alpha)\cdot(y, z) + \alpha\cdot(y', z')$$

where $\alpha \in [0,1]$ is a parameter that we specify later. Note that in (y'', z''), $t_{i'}$ is augmented by $\alpha\epsilon$, and $\textsc{ResCap}_{y'',z''}(t_{i'}, t_i) \geq (1-\alpha)\delta$, and $(y'', z'') \in \mathbb{S}$ because it is a convex combination of (y, z) and (y', z'). Consider applying $\textsc{Reroute}(t_{i'}, t_i, \rho)$ to (y'', z'') for some parameter $\rho \in [0,1]$. The idea is to choose α and ρ such that the exact amount, by which $t_{i'}$ was augmented, gets reassigned to t_i, by applying $\textsc{Reroute}(t_{i'}, t_i, \rho)$; so that eventually t_i is augmented while every other type (except t_0) has the same allocation probabilities as they originally had in (y, z). It is easy to verify that by setting

$$\alpha = \frac{y(t_{i'}, n)\delta}{2} \qquad\qquad \rho = \frac{\epsilon\delta}{2 + \epsilon\delta}$$

we get a feasible assignment in which the allocation probability of t_i is augmented by $\alpha\epsilon$ while every other type (except t_0) has the same allocation probabilities as in (y, z). We still need to show that $\alpha > 0$. The proof is again by contradiction. Suppose $\alpha = 0$, so it must be $y(t_{i'}, n) = 0$, which would imply that $t_{i'}$ is a degenerate type because $y(t_{i'}, i') > 0$ (because $\textsc{ResCap}_{y,z}(t_{i'}, t_i) > 0$), however (y, z) is a non-degenerate assignment by the hypothesis of the lemma, which is a contradiction. That completes the proof. $\square$

B Proofs from Section 5.2

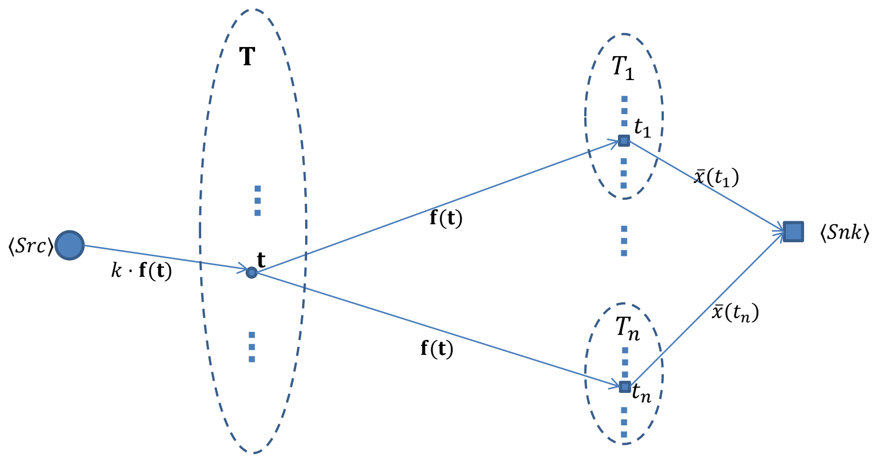

Figure 3: The bipartite graph used in the max-flow/min-cut argument of the proof of Theorem 3. The capacities are indicated on the edges.

Rest of the proof of Theorem 3. We give a proof of $P_{g_k} \subseteq \overline{\mathbb{X}}$ based on the min-cut/max-flow theorem. We start by constructing a directed bipartite graph as illustrated in Figure 3. On one side we put a node $\langle \mathbf{t} \rangle$, for each type profile $\mathbf{t} \in \mathbf{T}$. On the other side we put a node $\langle t_i \rangle$, for each type $t_i \in T_{\{1,\dots,n\}}$. We also add a source node $\langle \textsc{Src} \rangle$ and a sink node $\langle \textsc{Snk} \rangle$. We add a directed edge from $\langle \textsc{Src} \rangle$ to the node $\langle \mathbf{t} \rangle$, for each $\mathbf{t} \in \mathbf{T}$ and set the capacity of this edge to $k \cdot \mathbf{f}(\mathbf{t})$. We also add n outgoing edges for every node $\langle \mathbf{t} \rangle$, each one going to one of the nodes $\langle t_1 \rangle, \dots, \langle t_n \rangle$ and with a capacity of $\mathbf{f}(\mathbf{t})$. Finally we add a directed edge from the node $\langle t_i \rangle$, for each $t_i \in T_{\{1,\dots,n\}}$, to $\langle \textsc{Snk} \rangle$

with capacity of $\overline{x}(t_i)$. Consider a maximum flow from $\langle\textsc{Src}\rangle$ to $\langle\textsc{Snk}\rangle$. It is easy to see that there exists a feasible ex post implementation for $\overline{x}$ if and only if all the edges to the sink node $\langle\textsc{Snk}\rangle$ are saturated. In particular, if $\rho(\mathbf{t}, t_i)$ denotes the amount of flow from $\langle\mathbf{t}\rangle$ to $\langle t_i\rangle$, a feasible ex post implementation can be obtained by allocating to each type t_i with probability $\rho(\mathbf{t}, t_i)/\mathbf{f}(\mathbf{t})$ when the type profile $\mathbf{t}$ is reported by the agents.

We show that if a feasible ex post implementation does not exist, then $\overline{x} \notin P_{g_k}$. Observe that if a feasible ex post implementation does not exist, then some of the incoming edges of $\langle\textsc{Snk}\rangle$ are not saturated by the max-flow. Let (A, B) be a minimum cut such that $\langle\textsc{Src}\rangle \in A$ and $\langle\textsc{Snk}\rangle \in B$. Let $B' = B \cap T_N$. We show that the polymatroid inequality

$$\overline{x}(B') \leq g_k(B') \tag{7}$$

must have been violated. It is easy to see that the size of the cut is given by the following equation.

$$\textsc{Cut}(A, B) = \sum_{\mathbf{t}\in\mathbf{T}\cap A} \#\left\{i|t_i \in B\right\}\mathbf{f}(\mathbf{t}) + \sum_{\mathbf{t}\in\mathbf{T}\cap B} k\cdot\mathbf{f}(\mathbf{t}) + \sum_{\tau\in T_N\cap A} \overline{x}(\tau)$$

Observe that for each $\mathbf{t} \in \mathbf{T} \cap A$, it must be that $\#\{i|t_i \in B\} \leq k$, otherwise moving $\langle\mathbf{t}\rangle$ to B would decrease the size of the cut. So the size of the minimum cut can be in simply written as:

$$\textsc{Cut}(A, B) = \sum_{\mathbf{t}\in\mathbf{T}} \min\left(\#\left\{i|t_i \in B\right\}, k\right)\mathbf{f}(\mathbf{t}) + \sum_{\tau\in T_N\cap A} \overline{x}(\tau)$$

On the other hand, since some of the incoming edges of $\langle\textsc{Snk}\rangle$ are not saturated by the max-flow, it must be that

$$\sum_{\tau\in T_N} \overline{x}(\tau) = \textsc{Cut}(A \cup B - \langle\textsc{Snk}\rangle, \langle\textsc{Snk}\rangle) > \textsc{Cut}(A, B),$$

so

$$\sum_{\tau\in T_N\cap B} \overline{x}(\tau) > \sum_{\mathbf{t}\in\mathbf{T}} \min\left(\#\left\{i|t_i \in B\right\}, k\right)\mathbf{f}(\mathbf{t}).$$

The right hand side of the above inequality is the same as $\mathbf{E}_{\mathbf{t}\sim\mathbf{f}}[\min(\#\{i|t_i \in B\}, k)]$ which shows that polymatroid inequality (7) of P_{g_k} is violated so $\overline{x} \notin P_{g_k}$. That completes the proof. $\square$

C Proofs from Section 5.3

Proof of Lemma 4. Assuming that agents are independent (i.e., assuming $\mathbf{f}(\cdot)$ is a product distribution), $g_k(S)$ can be computed in time $O((n + |S|)\cdot k)$ using the following dynamic program in which G^i_j denotes the probability of the event that $\min(|\mathbf{t} \cap S \cap T_{\{1,\ldots,i\}}|, k) = j$.

$$g_k(S) = \sum_{j=1}^{k} j \cdot G^n_j$$

$$G^i_j = \begin{cases} G^{i-1}_k + (\sum_{t_i \in S \cap T_i} f_i(t_i)) \cdot G^{i-1}_{k-1} & 1 \leq i \leq n, j = k \\ G^{i-1}_j + (\sum_{t_i \in S \cap T_i} f_i(t_i)) \cdot (G^{i-1}_{j-1} - G^{i-1}_j) & 1 \leq i \leq n, 0 \leq j < k \\ 1 & i = 0, j = 0 \\ 0 & \text{otherwise} \end{cases}$$

□

???EDITING HERE.

D Independent Bidders: A Simplification of Border Constraints

In this section we show that, when bidders' types are independently drawn, feasibility condition can be checked by normalized interim allocation rules alone, and therefore the number of sufficient constraints of Theorem **??** in Section **??** can be reduced to be exponential only in the number of agents (and not in the typespace sizes). This is possible via certain local properties of monotonicity of the constraints in **??**. These properties will eventually enable us to find the "tightest" constraint by following a remarkably short sequence of constraints prescribed by a proper ordering of all bidders' types, even though the potential search space is exponential. Theorem 9 coarsely employs such local properties and constitutes as a first step in reducing the search space.

Theorem 9. *Assuming that agents are independent and there can be at most k winners, any given set of interim allocation rules $x_1, \cdots, x_n$ is feasible iff the following holds:*

$$\forall q_1, \cdots, q_n \in [0,1]: \qquad \sum_{i=1}^{n} \widehat{X}_i(q_i) \leq g_k(q_1, \cdots, q_n), \qquad (\text{MRM}'_k)$$

*where $\widehat{X}_i$ is defined according to Definition **??**, and g_k is defined as in Theorem **??** in Section **??**.*

Proof. It is not hard to see that conditions (MRM'_k) are a subset of (**??**), and its necessity follows from **??**. We therefore need only prove its sufficiency.

We first give a proof for the simple case when k is 1. In this case, by Theorem **??**, the interm allocation rules are feasible iff the maximum value of the function

$$H(S_1, \cdots, S_n) = \sum_{i=1}^{n} \sum_{t_i \in S_i} x_i(t_i) f_i(t_i) - 1 + \prod_{i=1}^{n} (1 - f_i(S_i))$$

is non-positive. Let $(S^*_1, \cdots, S^*_n)$ be a maximizer of $H(\cdot)$ and, if there are multiple maximizers, take one in which each S^*_i is set theoretically maximal. As in **??**, we relabel the types in T_i as $\theta^1_i, \theta^2_i, \ldots$ such that $x_i(\theta^1_i) \geq x_i(\theta^2_i) \geq \cdots$. We show that, for each i, if S^*_i contains θ^j_i, then it contains all elements θ_i^ℓ where $\ell < j$. For the sake of contradiction, suppose θ_i^ℓ ($\ell < j$) is not in S_i^*. We connect two facts to derive a contradiction: first, removing θ_i^j should not strictly decrease the value of H, and second, adding θ_i^ℓ should not weakly increase the value of H. Writing out these two facts, we have

$$\begin{aligned}\Delta' &\overset{\triangle}{=} H(S_1^*, \cdots, S_n^*) - H(S_1^*, \cdots, S_{i'}^* - \{\theta_i^j\}, \cdots, S_n^*) \\ &= x_i(\theta_i^j) f_i(\theta_i^j) - \prod_{d \neq i}^{n} (1 - f_d(S_d^*)) \cdot f_i(\theta_i^j) \\ &= \left(x_i(\theta_i^j) - \prod_{d \neq i}^{n} (1 - f_d(S_d^*)) \right) \cdot f_i(\theta_i^j) \geq 0.\end{aligned}$$

$$\begin{aligned}\Delta'' &= H(S_1^*, \cdots, S_i^* \cup \{\theta_i^\ell\}, \cdots, S_n^*) - H(S_1^*, \cdots, S_n^*) \\ &= x_i(\theta_i^\ell) f_i(\theta_i^\ell) - \prod_{d \neq i}^{n} (1 - f_d(S_d^*)) f_i(\theta_i^\ell) \\ &= \left(x_i(\theta_i^\ell) - \prod_{d \neq i}^{n} (1 - f_d(S_d^*)) \right) f_i(\theta_i^\ell) < 0.\end{aligned}$$

We therefore have $x_i(\theta_i^j) \geq \prod_{d \neq i} (1 - f_d(S_d^*)) > x_i(\theta_i^\ell)$, a contradiction. Now that we know that each S_i^* is downward-closed in the ordering of x_i, by **??** and the definition of H, $H(S_1^*, \cdots, S_n^*) = \sum_i \widehat{X}_i(f_i(S_i^*)) - g_1(f_1(S_1^*), \cdots, f_n(S_n^*))$. Its non-positivity therefore follows from condition (MRM$'_k$). □

E Separation Oracles

In Appendix D we showed that, when bidders are independent, the feasibility constraints have local structures which allow us to check constraints on noramlized interim allocation rules. Here we make more careful use of these properties and show that a proper ordering of all individual types of bidders allows us to check only linearly many constraints to find a violated one if there is any, for the case of independent bidders and $k = 1$.

Recall that, when k is 1, the set of constraints we need to check is

$$\forall S_1 \subseteq T_1, \cdots, \forall S_n \subseteq T_n : \qquad \sum_{i=1}^{n} \sum_{t_i \in S_i} x_i(t_i) f_i(t_i) \leq 1 - \prod_{i=1}^{n} (1 - f_i(S_i)) \tag{MRM$_1$}$$

Theorem 10. *For $k = 1$, algorithm 2 can check the feasibility of any given set of interim allocation rules $x_1, \cdots, x_n$, and if infeasible, it always finds $S_1, \cdots, S_n$ for which* (MRM$_1$) *is violated. Furthermore, this algorithm runs in time $O(N \log N)$, where $N = \sum_{i=1}^{n} |T_i|$ is the size of the input to the algorithm.*

Algorithm 2 CheckFeasibility($x_1, \cdots, x_n$)

Input: A set of interim allocation rules $x_1, \cdots, x_n$.
Output: $S_1, \cdots, S_n$, for which (MRM$_1$) is violated (if any).

```
 1: For each $i$, let $T_i = \{\theta_i^1, \theta_i^2, \cdots\}$ be a relabeling of the types in decreasing order of $x_i(\cdot)$.
 2: For every $i \in [n]$, $j \in [|T_i|]$, let $f_i^j = f_i(\theta_i^j)$, and $F_i^j = \sum_{r=1}^{j} f_i^j$.
 3: For each type $\theta_i^j$ define a weight $w(\theta_i^j) = (1 - F_i^{j-1}) \cdot x_i(\theta_i^j)$.
 4: Make a list $L$ of all types in type spaces of all agents and sort it in decreasing order of $w(\cdot)$.
 5: Set $S_i \leftarrow \emptyset$, for all $i \in [n]$.
 6: while $L \neq \emptyset$ do
 7:     Let $\theta_i^j$ denote the first element of the current list $L$.
 8:     Remove $\theta_i^j$ from $L$ and add it to $S_i$.
 9:     Check (MRM$_1$) for $S_1, \cdots, S_n$, and if violated, return $S_1, \cdots, S_n$.
10: end while
```

Proof. We need to show that if $x_1, \cdots, x_n$ are infeasible, then the above algorithm always finds $S_1, \cdots, S_n$ that violate the inequality (MRM$_1$). We prove this by contradiction. Suppose the algorithm does not find any violation. It is easy to see that the interim allocations are infeasible iff the maximum value of the following function is strictly positive.

$$H(S_1, \cdots, S_n) = \sum_{i=1}^{n} \sum_{t_i \in S_i} x_i(t_i) f_i(t_i) - 1 + \prod_{i=1}^{n} (1 - f_i(S_i))$$

Let $S_1^* \subset T_1, \cdots, S_n^* \subset T_n$ be a maximizer of H and if there are multiple maximizers, take the one in which each S_i^* is of maximal size. We show that there exists a point during the running of the algorithm where for all i, $S_i = S_i^*$, which would be a contradiction. During the running of the algorithm, consider the first time[12] that $S_i^* \subseteq S_i$ for all $i \in [n]$. We show that it must be $S_i = S_i^*$, so the algorithm must have indeed checked the (MRM$_1$) for $S_1^*, \cdots, S_n^*$. The proof is again by contradiction. Let i' be the index of the set $S_{i'}$ that was last updated, and let $\theta_{i'}^{j'}$ denote the element that was last added to it. It is easy to see[13] that $S_{i'} = S_{i'}^*$. Suppose there is some i'' such that $S_{i''} \neq S_{i''}^*$. Choose $\theta_{i''}^{j''} \in S_{i''} - S_{i''}^*$ with maximum weight. We show that adding $\theta_{i''}^{j''}$ to $S_{i''}^*$ may only increase H which would contradict the optimally or the maximality of $S_1^*, \cdots, S_n^*$.

[12]Since at the end of the algorithm $S_i = T_i$ for all i, we can always find such a first time.
[13]That is because each one of $S_{i''}$ and $S_{i''}^*$ consists of a proper prefix of the sequence of types $\theta_{i''}^1, \theta_{i''}^2, \cdots$

Define

$$\begin{aligned}
\Delta'' &= H(S_1^*, \cdots, S_{i''}^* \cup \{\theta_{i''}^{j''}\}, \cdots, S_n^*) - H(S_1^*, \cdots, S_n^*) \\
&= x_{i''}(\theta_{i''}^{j''}) f_{i''}(\theta_{i''}^{j''}) - \frac{\prod_{i=1}^n (1 - f_i(S_i^*))}{(1 - f_{i''}(S_{i''}^*))} f_{i''}(\theta_{i''}^{j''}) \\
&= \left(x_{i''}(\theta_{i''}^{j''})(1 - f_{i''}(S_{i''}^*)) - \prod_{i=1}^n (1 - f_i(S_i^*)) \right) \cdot \frac{f_{i''}(\theta_{i''}^{j''})}{1 - f_{i''}(S_{i''}^*)} \\
&= \left(w(\theta_{i''}^{j''}) - \prod_{i=1}^n (1 - f_i(S_i^*)) \right) \cdot \frac{f_{i''}^{j''}}{1 - F_{i''}^{j''-1}} \qquad (\Delta'')
\end{aligned}$$

The last equation follows[14] from $f_{i''}(S_{i''}^*) = F_{i''}^{j''-1}$ and the definition of $w(\cdot)$. Next, define Δ' to be the amount of change in H by removing $\theta_{i'}^{j'}$ from $S_{i'}^*$. Using a similar argument as above we can get

$$\begin{aligned}
\Delta' &= H(S_1^*, \cdots, S_n^*) - H(S_1^*, \cdots, S_{i'}^* - \{\theta_{i'}^{j'}\}, \cdots, S_n^*) \\
&= \left(w(\theta_{i'}^{j'}) - \left(\prod_{i=1}^n (1 - f_i(S_i^*)) \right) \frac{(1 - f_{i'}(S_{i'}^*) - f_{i'}(\theta_{i'}^{j'}))}{(1 - f_{i'}(S_{i'}^*))} \right) \cdot \frac{f_{i'}^{j'}}{1 - F_{i'}^{j'-1}} \\
&\leq \left(w(\theta_{i'}^{j'}) - \prod_{i=1}^n (1 - f_i(S_i^*)) \right) \cdot \frac{f_{i'}^{j'}}{1 - F_{i'}^{j'-1}}
\end{aligned}$$

Observe that by optimality of $S_1^*, \cdots, S_n^*$, Δ' must be non-negative, which implies that the inside of the big parenthesis in the last equation should be non-negative. On the other hand, since the algorithm must have removed $\theta_{i''}^{j''}$ from the list L prior to $\theta_{i'}^{j'}$, it must be that $w(\theta_{i''}^{j''}) \geq w(\theta_{i'}^{j'})$. This implies that the inside of the big parenthesis of equation (Δ'') must also be non-negative which proves that $\Delta'' \geq 0$. That completes the proof. $\square$

F Proofs in Section ??

Proof of Theorem **??**. The essence of the proof can be seen as a Birkhoff-von Neumann-Rado style decomposition to a vector by another one majorizing it. However, due to the discreteness of the type space, extra care need be taken. We start from the following definition, .

Definition 10 (Distribution Preserving Stochastic Transformation (DPST))**.** A pair of functions (π_i, μ_i), where $\pi_i, \mu_i : T_i \times T_i \to [0, 1]$ is called a *"stochastic transformation"* for agent i, iff for all $\theta \in T_i$, $\sum_{\theta' \in T_i} \pi_i(\theta', \theta) = 1$. A stochastic transformation can be applied to agent i by replacing agent i with a proxy that does the following. If the actual type reported by agent i is t_i, the proxy instead reports a type $t_i' = \Pi_i(t_i)$ where $\Pi_i(t_i)$ is a random function that returns each $\theta' \in T_i$ with probability $\pi_i(\theta', t_i)$. If the proxy is chosen as a winner, then the proxy announces agent i as

[14]That is because each one of $S_{i''}$ and $S_{i''}^*$ consists of a proper prefix of the sequence of types $\theta_{i''}^1, \theta_{i''}^2, \cdots$, and $\theta_{i''}^{j''}$ was the element with maximum weight from $S_{i''} - S_{i''}^*$.

a winner with probability $\mu_i(t'_i, t_i)$, otherwise agent i is not a winner. Furthermore, A stochastic transformation (π_i, μ_i) is called *"distribution preserving"* iff for all $\theta' \in T_i$, $\sum_{\theta \in T_i} \pi_i(\theta', \theta) f_i(\theta) = f_i(\theta')$, i.e., the the distribution of the types reported by the proxy is the same as the distribution of types of the actual agent i. We use the notation $(\pi_i, \mu_i) \times x_i$ to denote the new interim allocation rule that results from applying (π_i, μ_i) to x_i.

Note that for any given ex-post allocation rule $x : T \to [0, 1]$, applying a DPST (π_i, μ_i) to some agent i yields a new feasible ex-post allocation rule in which all the agents, except for agent i, observe the same interim allocation rules as before. We will show that by carefully choosing (π_i, μ_i) we can convert x_i to any other interim allocation rule x'_i given that $x'_i \preceq x_i$.

Theorem 11. *Given any two interim allocation rules x_i and x^*_i such that $x_i \succeq x^*_i$, there exists a DPST (π_i, μ_i) such that $x^*_i = (\pi_i, \mu_i) \times x_i$.*

We prove the previous theorem by showing that there is sequence of DPSTs that can be applied to x_i to obtain x^*_i. We first show that any such sequence can be converted to a single DPST.

Lemma 6. *Any sequence of two or more DPSTs can be substituted by an equivalent DPST.*

Proof. Let (π_i, μ_i) and (π'_i, μ'_i) be two DPSTs. We define $(\pi''_i, \mu''_i) = (\pi_i, \mu_i) \times (\pi'_i, \mu'_i)$ as follows.

$$\forall \theta, \theta'' \in T_i : \qquad \pi''_i(\theta'', \theta) = \sum_{\theta' \in T_i} \pi_i(\theta'', \theta') \cdot \pi'_i(\theta', \theta)$$

$$\forall \theta, \theta'' \in T_i : \qquad \mu''_i(\theta'', \theta) = \sum_{\theta' \in T_i} \mu_i(\theta'', \theta') \cdot \mu'_i(\theta', \theta)$$

It is easy to see that for any x_i, $(\pi_i, \mu_i) \times \big((\pi'_i, \mu'_i) \times x_i\big) = (\pi''_i, \mu''_i) \times x_i$. □

Next, we describe three kinds of DPST that we will use later. All of the following DPSTs take an initial allocation x_i as one of their arguments and assume that $T_i = \{\theta^1_i, \theta^2_i, \cdots\}$ is a relabeling of types in decreasing order of x_i, and $F^j_i = \sum_{r=1}^{j} f_i(\theta^r_i)$. Each one of them returns a DPST (π_i, μ_i) that can be be applied to x_i to achieve a certain effect.

- ReorderTypes$(x_i, \{\theta^{*1}_i, \theta^{*2}_i, \cdots\})$. Given an interim allocation rule x_i and an ordering of types of agent i, i.e., $\theta^{*1}_i, \theta^{*2}_i, \cdots$, this DPST converts x_i to another interim allocation rules that gives the highest probability of allocation to θ^{*1}_i, then to θ^{*2}_i, and so on. Let $F^{*j}_i = \sum_{r=1}^{j} f_i(\theta^{*r}_i)$. This DPST can be described as follows. If agent i reports a type $t_i = \theta^{j^*}_i$, then the proxy chooses Z uniformly at random from the interval $[F^{*j^*-1}_i, F^{*j^*}_i)$, and finds the index j' such that $Z \in [F^{j'-1}_i, F^{j'}_i)$. The proxy reports $t'_i = \theta^{j'}_i$. The agent wins whenever the proxy wins. Formally, (π_i, μ_i) can be defined as

$$\forall j', j^* \in [|T_i|] : \qquad \pi_i(\theta^{j'}_i, \theta^{*j^*}_i) = \left| [F^{j'-1}_i, F^{j'}_i) \cap [F^{*j^*-1}_i, F^{*j^*}_i) \right|$$

$$\forall j', j^* \in [|T_i|] : \qquad \mu_i(\theta^{j'}_i, \theta^{*j^*}_i) = 1$$

- LowerTypes(x_i, a). This DPST tries to zero the the interim allocation probability for all types θ^j_i such that $j \geq a$ as explained next. If agent i reports a type $t_i = \theta^j_i$, then the proxy reports the same type, i.e., $t'_i = \theta_i^j$. The agent wins iff $j < a$ and the proxy wins. Formally, (π_i, μ_i) can be defined as

$$\forall j, j' \in [|T_i|]: \qquad \pi_i(\theta_i^{j'}, \theta_i^j) = \begin{cases} 1 & j = j' \\ 0 & j \neq j' \end{cases}$$

$$\forall j, j' \in [|T_i|]: \qquad \mu_i(\theta_i^{j'}, \theta_i^j) = \begin{cases} 1 & j < a \\ 0 & j \geq a \end{cases}$$

- MixTypes(x_i, a, b). This DPST mixes the types in range $\theta_i^a, \cdots, \theta_i^b$ uniformly as explained next. If agent i reports a type $t_i = \theta_i^j$, then if $j \notin a \cdots b$ the proxy reports the same type $t'_i = \theta_i^j$, otherwise the proxy chooses Z uniformly at random from the interval $[F_i^{a-1}, F_i^b)$ and finds j' such that $Z \in [F_i^{j'-1}, F_i^{j'})$. The proxy reports $t'_i = \theta_i^{j'}$. The agent wins whenever the proxy wins. Formally, (π_i, μ_i) can be defined as

$$\forall j, j' \in [|T_i|]: \qquad \pi_i(\theta_i^{j'}, \theta_i^j) = \begin{cases} \frac{F_i^{j'} - F_i^{j'-1}}{F_i^b - F_i^{a-1}} & a \leq j \leq b \\ 1 & \text{otherwise} \end{cases}$$

$$\forall j, j' \in [|T_i|]: \qquad \mu_i(\theta_i^{j'}, \theta_i^j) = 1$$

Next, we present an algorithm that can convert any interim allocation rule x_i to any other interim allocation rule x_i^* by combining a sequence of the DPSTs described above, given that $x_i^* \preceq x_i$.

The next theorem implies Theorem 11.

Theorem 12. *For any x_i and x_i^* such that $x_i \succeq x_i^*$, the algorithm 3 returns a DPST (π_i, μ_i) such that $x_i^* = (\pi_i, \mu_i) \times x_i$. The algorithm runs for at most $|T_i|$ iterations and runs in $O(|T_i|^3)$ time. Note that the size of the input of the algorithm is $O(|T_i|)$.*

Proof. Let $J_+^{(\ell)} = \{j \in [|T_i|] | \widehat{X}_i^{(\ell)}(F_i^j) > \widehat{X}_i^*(F_i^j)\}$, i.e., $J_+^{(\ell)}$ is the indices of all the points at which $\widehat{X}_i^{(\ell)}$ is strictly more than $\widehat{X}_i^*$. By induction it is easy to see that all of the following statements hold at every iteration ℓ of the algorithm 3.

1. $x_i^{(\ell)} = (\pi_i^{(\ell)}, \mu_i^{(\ell)}) \times x_i$.

2. $x_i^* \preceq x_i^{(\ell)} \prec x_i^{(\ell-1)}$.

3. $|J_+^{(\ell+1)}| < |J_+^{(\ell)}|$. This can be shown as follows. If $\widehat{X}_i^{(\ell)}(q) = \widehat{X}_i^*(q)$ for some $q \in [0,1]$ at some iteration $\ell = \ell'$, then it will also hold for the same q and all iterations $\ell \geq \ell'$, so $J_+^{(\ell+1)} \subseteq J_+^{(\ell)}$. Furthermore, the way the algorithm chooses α at iteration ℓ implies that for some index in $j \in a \cdots b$, $\widehat{X}_i^{(\ell)}(F_i^j) > \widehat{X}_i^*(F_i^j)$, but $\widehat{X}_i^{(\ell)+1}(F_i^j) = \widehat{X}_i^*(F_i^j)$ which means $j \in J_+^{(\ell)}$ but $j \notin J_+^{(\ell+1)}$ which proves the statement.

Algorithm 3 ComputeDPST(x_i^*, x_i)

```
Input: Two interim allocation rules $x_i$ and $x_i^*$. Let $T_i = \{\theta^{*1}_i, \theta^{*2}_i, \cdots\}$ denote a relabeling of types in decreasing order of $x_i^*$. Also let $F^{*j}_i = \sum_{r=1}^{j} f_i(\theta^{*r}_i)$.
Output: A DPST that converts $x_i$ to $x_i^*$.
1: Let $(\pi_i^{(0)}, \mu_i^{(0)}) \leftarrow$ ReorderTypes$(x_i, \{\theta^{*1}_i, \theta^{*2}_i, \cdots\})$.
2: Let $x_i^{(0)} \leftarrow (\pi_i^{(0)}, \mu_i^{(0)}) \times x_i^*$ and let $\ell \leftarrow 0$.
3: while $x_i^{(\ell)} \neq x_i^*$ do
4:     Let $b$ be the highest index such that $\widehat{X}_i^{(\ell)}(F^{*b}_i) > \widehat{X}_i^*(F^{*b}_i)$.
5:     Let $a$ be the lowest index such that for all $j \in a \cdots b$, $\widehat{X}_i^{(\ell)}(F^{*j}_i) > \widehat{X}_i^*(F^{*j}_i)$.
6:     if $b = |T_i|$ then
7:         Let $(\pi_i', \mu_i') \leftarrow$ LowerTypes$(x_i^{(\ell)}, a)$.
8:     else
9:         Let $(\pi_i', \mu_i') \leftarrow$ MixTypes$(x_i^{(\ell)}, a, b)$.
10:    end if
11:    Pick the largest $\alpha \in [0,1]$ for which still $\alpha \cdot (\pi_i', \mu_i') \times x_i^{(\ell)} + (1-\alpha) \cdot x_i^{(\ell)} \succeq x_i$.
12:    Let $x_i^{(\ell+1)} \leftarrow \alpha \cdot (\pi_i', \mu_i') \times x_i^{(\ell)} + (1-\alpha) \cdot x_i^{(\ell)}$.
13:    Let $(\pi_i^{(\ell+1)}, \mu_i^{(\ell+1)}) \leftarrow \alpha \cdot (\pi_i', \mu_i') \times (\pi_i^{(\ell)}, \mu_i^{(\ell)}) + (1-\alpha) \cdot (\pi_i^{(\ell)}, \mu_i^{(\ell)})$.
14:    $\ell \leftarrow \ell + 1$.
15: end while
16: return $(\pi_i^{(\ell)}, \mu_i^{(\ell)})$.
```

Note that if $J_+^{(\ell)} = \emptyset$, the second statement implies that $x_i^{(\ell)} = x_i^*$. But then the last statement implies that the algorithm must stop after at most $|T_i|$ iterations. Finally, when the algorithm stops, the first statement implies that $x_i^* = x_i^{(\ell)} = (\pi_i, \mu_i) \times x_i$. That completes the proof. □

This completes the proof of Thoerem **??**. □

Proof of Theorem **??**. Note that (**??**) is a special case of (MRM$'_k$) so it is necessary. We will show that it is also sufficient. We prove the theorem by explicitly constructing a symmetric ex-post allocation rule x^* such that the corresponding interim allocation rules x_i^* dominate x_i. By Theorem 12, a DPST (π_i, μ_i) can be computed to convert x_i^* to the the desired interim allocation x_i and because of symmetry the same DPST should work for all agents. Let $T_i = \{\theta_i^1, \theta_i^2, \cdots\}$ denote a relabeling of types in decreasing order of x_i and let $F_i^j = \sum_{r=1}^{j} f_i(\theta_i^r)$. Suppose the vector of types reported by the agents is $t = (\theta_1^{j_1}, \cdots, \theta_n^{j_n})$, so that $j_1, \cdots, j_n$ are the indices of the types reported by the agents. Define x^* to be the following allocation rule. Choose the k agents who have the lowest indices as the winners, breaking ties uniformly at random. To complete the proof, we need to show that the resulting interim allocations x_i^* dominate the x_i. Let $\widehat{X}_i^*$ denote the resulting cumulative normalized interim allocation rule. For any $\bar{j} \in [|T_i|]$, $\widehat{X}_i^*(F_i^{\bar{j}})$ is the expected probability that agent i has a type with index at most $\bar{j}$ and also wins, so $n \cdot \widehat{X}_i^*(F_i^{\bar{j}})$ is the expected number of winners whose type indices are at most $\bar{j}$, which is equal to the expected number of agents, up to k, whose types have index at most $\bar{j}$, i.e.,

$$\begin{aligned} n \cdot \widehat{X}_i^*(F_i^{\bar{j}}) &= \mathop{\mathrm{E}}_{j_1,\cdots,j_n}[\min(k, |\{i | j_i \leq \bar{j}\}|)] \\ &= g_k(F_i^{\bar{j}}, \cdots, F_i^{\bar{j}}) \end{aligned}$$

on the other hand from (**??**) we can argue that

$$n \cdot \widehat{X}_i(F_i^{\bar{j}}) \leq g_k(F_i^{\bar{j}}, \cdots, F_i^{\bar{j}})$$

So $\widehat{X}_i(q) \leq \widehat{X}_i^*(q)$ for all $q \in \{F_i^{\bar{j}} | \bar{j} \in [|T_i|]\}$. Since $\widehat{X}_i^*$ is concave and $\widehat{X}_i$ is piece-wise linear and its derivative changes only at $q \in \{F_i^{\bar{j}} | \bar{j} \in [|T_i|]\}$, it must be that $\widehat{X}_i(q) \leq \widehat{X}_i^*(q)$ for all $q \in [0,1]$. That means x_i^* dominates x_i, which completes the proof. $\square$

G Context-freeness

Recall: $X : [0,1] \to [0,1]$ is *cumulative normalized interim service rule.* $\tilde{x} : T \to [0,1]$ is *interim service rule.* $x : T \to [0,1]^n$ is *interim allocation rule.*

Consider a set of lotteries L. Let $0 \leq \alpha_1 < \ldots < \alpha_K \leq 1$ be the distinct service probabilities of lotteries offered in L. Given each type $t \in T$ and integer $k, 1 \leq k \leq K$, define ℓ_k^t to be type t's favorite lottery among those that service him with probability α_k. When t is offered the set of lotteries L, he chooses a lottery among $\ell_1^t, \ldots, \ell_K^t$ that maximizes his utility if that utility is non-negative, and chooses no lotteries otherwise. Given any k define S_k to be the set of types that choose a lottery that service with probability α_k. Now define $L_k = \{\ell_k^t - \ell_{k-1}^t | t \in S_k\}$, where ℓ_0^t is the zero lottery. Notice that the probability of service in each element of L_k is $\alpha_k - \alpha_{k-1}$, which is between 0 and 1.

Definition 11. A set of lotteries L with K distinct probabilities of service is well behaved if,
Given any k and type $t \in S_i$,

1. Each member of L_k is a feasible lottery, i.e., all its coefficients are non-negative.
2. For $j \leq i$, The lottery $\ell_j^t - \ell_{j-1}^t$ maximizes t's utility among the lotteries in L_i.

Definition 12. An environment satisfies Property 1 if given any cumulative normalized service rule X, the set of lotteries offered in its optimal allocation rule is well-behaved.

BEGINNOTE: the above definition is equivalent to the well-behaved contour sets condition which we previously had. I think the new definition is more formal.ENDNOTE

For any $q \in [0,1]$, let X_q be the step function such that $X_q(z) = 0$ if $z \leq q$, and $X_q(z) = 1$ otherwise. Let x_q be the optimal allocation rule for X_q. Define the function $R : [0,1] \to [0,1]$, to be $R(q) = R(X_q)$ for each $q \in [0,1]$.

Definition 13. An environment satisfies *revenue linearity* if for all cumulative normalized interim service rule X, we have $R(X) = \int_0^1 X'(q)R(q)dq = \int_0^1 X(q)R'(q)dq$.

First, note that the second equality follows using integration by parts, and the fact that $R(0) = R(1) = 0$.

Second, notice that to prove revenue linearity it is sufficient to prove revenue sub-linearity, that is, $R(X) \leq \int_0^1 X'(q)R(q)dq$. This is because in *any* environment, $R(X) \geq \int_0^1 X'(q)R(q)dq$, since we can define the interim allocation rule x to be one that for each $0 \leq q \leq 1$, runs x_q with probability $X'(q)$.

Theorem 13. *An environment is context-free if and only if it satisfies Property 1 if and only if it satisfies revenue linearity.*

Proof. The theorem follows from the following 3 lemmas. $\square$

Lemma 7. *If an environment satisfies Property 1 , then it satisfies revenue linearity.*

Proof. To prove $R(X) = \int_0^1 X'(q)R(q)dq$, consider the optimal allocation rule x and let L be the set of lotteries offered in it with K distinct probabilities of service (How to say it if K is infinite?). Property 1 implies that L is well-behaved. Consider allocation rules $x_1, \ldots, x_K$ that are constructed by offering lottery sets $L_1, \ldots, L_K$, respectively. Recall that the first property of well-behaved-ness states that $L_1, \ldots, L_K$ are feasible lottery sets. The second property implies that $t \in S_i$ chooses $\ell_j^t - \ell_{j-1}^t$ when offered L_j, for $j \leq i$. As a result, $\sum_{j=1}^K x_j(t) = \sum_{j=1}^i \ell_j^t - \ell_{j-1}^t = \ell_i^t = x(t)$. We conclude that $x = \sum_1^K x_j$. Given the definition of the function $R(.)$, we have $R(x_j) \leq R(q_j)$, where $q_j = F(\bigcup_{i=j}^K S_i)$. As a result, $R(X) \leq \int_0^1 X'(q)R(q)$. (Actually I have assumed that K is finite, and therefore the result we get is in summation form rather than integral.)

$\square$

Lemma 8. *If an environment satisfies revenue linearity, then it is context-free.*

Proof. As observed before, linearity implies that the optimal allocation for X is to run $R(q)$ with probability $X'(q)$. This proves that $R'(q)$ is a virtual value function for the environment. $\square$

Lemma 9. *If an environment is context-free, then it satisfies Property 1 .*

Proof. Consider an interim service rule $\tilde{x}$ that is optimal form some cumulative normalized interim service rule X. For any q the contour set is define to be $S_q^{\tilde{x}} = \{t : \tilde{x}(t) \leq q\}$. By regularity, the set $T \backslash S_q$ is equal to the set of agents serviced with probability 1 in the optimal allocation subject to the step function $X_{q'}$, where q' is the measure of $T \backslash S_q$, that is, $q' = F(T \backslash S_q)$. This implies the no negative gradient condition and also convexity of S_q. $\square$

G.1 Understanding Property 1

Here we define Property 2 and prove that it is equivalent to Property 1 .

Definition 14. A set of lotteries L with K distinct probabilities of service is well behaved 2 if, Given any k and type $t \in S_i$,

1. Each member of L_k is a feasible lottery, i.e., all its coefficients are non-negative.
2. For each i, $T \backslash \bigcup_{j=i}^K S_j$ is a convex set.

Definition 15. An environment satisfies Property 2 if given any cumulative normalized service rule X, the set of lotteries offered in its optimal allocation rule is well behaved 2 .

Lemma 10. *An environment satisfies Property 2 if and only if it satisfies Property 1 .*