Dütting, P., Fischer, F., Jirapinyo, P., Lai, J. K., Lubin, B., and Parkes, D. C. (2015) Payment rules through discriminant-based classifiers. ACM Transactions on Economics and Computation, 3(1), 5.

There may be differences between this version and the published version. You are advised to consult the publisher's version if you wish to cite from it.

http://eprints.gla.ac.uk/131325/

Deposited on: 18 November 2016

Enlighten – Research publications by members of the University of Glasgow
http://eprints.gla.ac.uk

A

Payment Rules through Discriminant-Based Classifiers

PAUL DÜTTING, Stanford University, USA
FELIX FISCHER, University of Cambridge, UK
PICHAYUT JIRAPINYO, Bain & Company, Singapore
JOHN K. LAI, Harvard University, USA
BENJAMIN LUBIN, Boston University, USA
DAVID C. PARKES, Harvard University, USA

In mechanism design it is typical to impose incentive compatibility and then derive an optimal mechanism subject to this constraint. By replacing the incentive compatibility requirement with the goal of minimizing expected ex post regret, we are able to adapt statistical machine learning techniques to the design of payment rules. This computational approach to mechanism design is applicable to domains with multi-dimensional types and situations where computational efficiency is a concern. Specifically, given an outcome rule and access to a type distribution, we train a support vector machine with a specific structure imposed on the discriminant function, such that it implicitly learns a corresponding payment rule with desirable incentive properties. We extend the framework to adopt succinct k-wise dependent valuations, leveraging a connection with maximum *a posteriori* assignment on Markov networks to enable training to scale up to settings with a large number of items; we evaluate this construction in the case where $k = 2$. We present applications to multi-parameter combinatorial auctions with approximate winner determination, and the assignment problem with an egalitarian outcome rule. Experimental results demonstrate that the construction produces payment rules with low ex post regret, and that penalizing classification error is effective in preventing failures of ex post individual rationality.

Categories and Subject Descriptors: J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms: Algorithms, Economics, Theory

Additional Key Words and Phrases: Computational Mechanism Design, Support Vector Machines

ACM Reference Format:
Paul Dütting, Felix Fischer, Pichayut Jirapinyo, John K. Lai, Benjamin Lubin, David C. Parkes, 2013. Payment Rules through Discriminant-Based Classifiers. *ACM* V, N, Article A (January YYYY), 39 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Mechanism design studies situations where a set of agents each hold private information regarding their preferences over different outcomes. A mechanism receives claims about agent preferences, selects and enforces an outcome, and optionally collects payments. The classical approach is to impose *incentive compatibility* on the design, ensuring that agents truthfully report their preferences in equilibrium. Subject to this incentive constraint, the goal is to identify a mechanism, i.e., a way of choosing an outcome and payments based on agents' reports, that optimizes a given design objective such as welfare or revenue.

A preliminary version of the results presented in this article appeared in the *Proceedings of the 13th ACM Conference on Electronic Commerce*.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© YYYY ACM 0000-0000/YYYY/01-ARTA $15.00
DOI:http://dx.doi.org/10.1145/0000000.0000000

There are, however, significant challenges associated with this classical approach. First, it can be analytically cumbersome to derive optimal mechanisms for domains that are multi-dimensional, in the sense that each agent's private information is described through more than a single number, and few results are known in this case. An example of a multi-dimensional domain is a combinatorial auction, where an agent's preferences are described by a value for each of several different bundles of items. Second, incentive compatibility can be costly, in that adopting it as a hard constraint can preclude mechanisms with other desirable properties. For example, imposing the strongest form of incentive compatibility, truthfulness in a dominant strategy equilibrium or *strategyproofness*, necessarily leads to poor revenue, vulnerability to collusion, and vulnerability to false-name bidding in combinatorial auctions where valuations exhibit complementarities among items [Yokoo et al. 2004; Ausubel and Milgrom 2006; Rastegari et al. 2011].[1] A third difficulty occurs when the optimal mechanism has an outcome or payment rule that is computationally intractable. In this case, the challenge is to simultaneously handle both the incentive constraints and the requirements of worst-case tractability in the design.

1.1. Our Approach

In the face of these difficulties, we adopt statistical machine learning to automatically infer mechanisms with good incentive properties. Rather than imposing incentive compatibility as a hard constraint, we start from a given outcome rule, typically expressed as an algorithm, and then use machine learning techniques to identify a payment rule that minimizes agents' *expected ex post regret*. The ex post regret of an agent for truthful reporting in a given instance, or just *regret* where it causes no confusion, is the maximum amount by which its utility could increase through a misreport, while holding constant the reports of others. The expected ex post regret is the average ex post regret over all agents and all preference types, calculated with respect to a distribution on types. Our approach is applicable to domains that are multi-dimensional, and domains for which the computational efficiency of outcome rules is a concern. The methodology seeks a payment rule that obtains the best possible ex post incentive properties, and views all other aspects related to payments, such as revenue, as secondary. Rather, a designer needs to experiment with modified outcome rules in order to achieve different payment desiderata.

In place of ex post incentives, an alternative design stance would adopt the goal of minimizing *interim regret*. The interim regret for an agent with a particular type is the maximum amount by which the agent's expected utility, given the conditional distribution on types of others, could increase through a misreport. The expected interim regret of a mechanism would average interim regret over all possible types. Whereas a mechanism with zero expected ex post regret is strategyproof, a mechanism with zero expected interim regret is Bayes-Nash incentive compatible, both with the exception of a set of types with measure zero. In this sense, our design stance, being about ex post incentives, is more in the spirit of strategyproofness than Bayes-Nash incentive compatibility.

Still, it is important to emphasize that the interesting application of our approach is to settings in which there is no payment rule that can provide strategyproofness for the given outcome rule. We thus depart in a significant way from the typical approach to mechanism design, which assumes equilibrium behavior on the part of agents. We will not achieve an expected ex post regret of zero, and will therefore not obtain strategyproof designs. The analysis we provide is not an equilibrium analysis. This noted, we can make two observations

[1]By the revelation principle, this weakness should be ascribed to insisting on mechanisms that are analyzed in equilibrium (dominant-strategy or otherwise), and not to the imposition of incentive constraints *per se*. Our approach seeks approximate incentive compatibility, and we do not study the incentive compatible analogues to the mechanisms that are designed through discriminant-based payment rules, nor do we study the equilibrium properties of our designed mechanisms.

that start from ex post incentive properties but also yield interim incentive properties in the usual way.

First, our formulation ensures that an agent's payment, conditioned on an outcome, is independent of its report. Because of this, the only way an agent can improve its utility is by changing its report in a way that changes the outcome. Generically, this provides mechanisms where there is no gain in utility from infinitesimal changes in reports around an agent's true report, i.e., no marginal benefit from misreports; this holds ex post, and thus also ex interim for an agent with knowledge only of its own type. This local stability property occurs in practice in the *generalized second-price* (GSP) auction used for sponsored search, and has also been emphasized by Erdil and Klemperer [2010] in the context of combinatorial auction design.

Second, if the expected ex post regret is bounded from above by some constant $\epsilon_1 > 0$, then for a setting with finite types this immediately implies a bound of the following form: the expected ex post regret for an agent with a type sampled from the type distribution is at most $\epsilon_2 > 0$ with probability at least 1-δ, for some $\delta > 0$, where $\epsilon_2\delta \leq \epsilon_1$. Since the interim regret is bounded from above by the expected ex post regret of an agent with the same type, we additionally have a bound of the following form: the interim regret for an agent with a type sampled from the type distribution is at most $\epsilon_2 > 0$ with probability at least 1-δ, for some $\delta > 0$. The effect of this is that an agent for whom strategic behavior is costly, and with interim beliefs about others' types, would be truthful when the cost is greater than the interim regret; in particular, if the cost is greater than ϵ_2, then truthful behavior would be optimal for the agent with probability at least $1 - \delta$.

Returning now to the main theme, the approach that we take to the design of payment rules is to recognize that the payment rule of a strategyproof mechanism can be thought of as a classifier for predicting the outcome. In particular, the payment rule implies a price to each agent for each outcome, and the selected outcome must simultaneously maximize the reported value minus price for every agent. By limiting ourselves to discriminant-based classifiers, which use a discriminant function to score different outcomes and predict the outcome with the highest score, and in particular to discriminant functions with "value-minus-price" structure where the price can be an arbitrary function of the outcome and the reports of other agents, we obtain a remarkably direct connection between multi-class classification and mechanism design.

For an appropriate loss function, the discriminant function of a classifier that minimizes generalization error over a hypothesis class has a corresponding payment rule that minimizes expected ex post regret among all payment rules corresponding to classifiers in this class. Conveniently, an appropriate method exists for multi-class classification with large outcome spaces that supports the specific structure of the discriminant function, namely the method of *structural support vector machines* [Tsochantaridis et al. 2005; Joachims et al. 2009]. While use of this method restricts us to learning discriminant functions that are linear in feature vectors depending on agents' reported types, the restriction is not severe: the feature vectors can be non-linear functions of the reported types, and as with standard support vector machines it is possible to adopt non-linear kernels. This ultimately enables discriminant functions and thus price functions that depend in a non-linear way on the outcome and the reported types of agents.

The computational cost associated with our approach occurs offline during training, when a payment rule is learned for a given outcome rule. The learned payment rules have a succinct representation, through the standard support vector machine approach, and are fast to evaluate at run-time in the context of a deployed mechanism. A challenge in the context of structural support vector machines is to handle the large number of possible outcomes, or labels of the classification problem, during training. One way to address this in our setting is to work with valuation functions for which training can be formulated as a succinct convex optimization problem. In particular, we adopt k-wise dependent valuations [Conitzer et al.

2005] and leverage a connection with maximum *a posteriori* assignment on Markov networks to scale-up our framework in application to combinatorial auctions.

1.2. Evaluation

In illustrating the framework, we focus on three situations where strategyproof payment rules are not available:

(*i*) multi-minded combinatorial auctions, in which each agent is interested in a constant number of bundles, and where winner determination is provided through a greedy allocation rule,
(*ii*) an assignment problem with multiple distinct items, agents with unit-demand valuations, and an *egalitarian* outcome rule, i.e., an outcome rule that maximizes the minimum value of any agent, and
(*iii*) combinatorial auctions with k-wise dependent valuations, in which each agent's valuation has a graphical representation, and where winner determination is provided through a greedy allocation rule.

The egalitarian rule, also referred to as *max-min fairness*, has been used by others to illustrate the challenge of truthful mechanism design; e.g., Lavi et al. [2003], who motivate this in the context of Rawls' theory of justice. Although one might also wish to define fairness with regard to utility, i.e., including payments, we follow others in adopting the egalitarian rule as a canonical example of a non-implementable outcome rule.

Our experimental results demonstrate low expected regret even when the 0/1 classification accuracy is only moderately good, and better regret properties than those obtained through the simple Vickrey-Clarke-Groves (VCG) based payment rules that we adopt as a baseline. In addition, we give special consideration to the failure of ex post individual rationality (IR), and introduce methods to bias the classifier to avoid these kinds of errors and also *post hoc* methods to adjust trained payments, or even allocations, to reduce or eliminate them.

For setting (i), we find that our learned rules perform similarly to VCG-based rules. In setting (ii), our learned rules perform significantly better than VCG-based rules, which is understandable given that the egalitarian objective is quite different from the welfare maximization objectives for which the VCG idea is designed. In setting (iii), our learned rules provide better regret properties than VCG-based rules for large numbers of items, and allow us to trade off IR violation and regret more effectively than VCG-based rules. We are able to scale to instances with tens of items in setting (iii), as our training problem is polynomial in the number of items even though we are running a combinatorial auction.

1.3. Related Work

Conitzer and Sandholm [2002] introduced the agenda of *automated mechanism design* (AMD) and formulated mechanism design as the search for an allocation rule and a payment rule among a class of rules satisfying incentive constraints. While the basic idea of optimal design is familiar from the seminal work of Myerson [1981], a novel aspect of AMD is its formulation as a search problem over the space of all possible mappings from discrete type profiles to outcomes and payments. AMD is intractable when an explicit representation of the outcome and payment rules is used, because the type space is exponentially large in the number of agents.

One way to make AMD more tractable is to search through a parameterized space of incentive-compatible mechanisms [Guo and Conitzer 2010]. More recently, advances in AMD have been made by considering domains with additive valuations and symmetry among agents, and by adopting Bayes-Nash incentive compatibility (BIC) rather than strategyproofness [Cai et al. 2012]. Still, these approaches seem limited to domains in which the

outcome rule can be succinctly represented, which likely is not the case for the kinds of combinatorial auction problems we consider.

Lavi and Swamy [2005] describe a method that takes any approximation algorithm for a set packing problem with a matching integrality gap and turns it into a mechanism with the same approximation guarantee that is strategyproof in expectation. Set packing includes combinatorial auctions as a special case. Bei and Huang [2011] and Hartline et al. [2011] describe an approach for turning an allocation rule into a mechanism that yields essentially the same expected amount of social welfare or social surplus and satisfies BIC. The approach computes an allocation and prices based on types sampled from probability distributions derived from the revealed types, and is applicable to both single-parameter and multi-parameter domains.

The target of minimizing expected ex post regret and the imposition of agent-independent prices make the incentive properties of mechanisms designed through our approach incomparable to BIC. On one hand, we are interested in minimizing statistics of *ex post* regret, and thus provide stronger guarantees than those of BIC. On the other hand, we don't guarantee zero expected regret (which would imply strategyproofness, and therefore BIC.) Another distinction is that our approach can accommodate objectives that are non-separable across agents, such as in the egalitarian assignment problem.

In addition, in determining the outcome and payments for a given instance, the approach of Bei and Huang and Hartline et al. evaluates the outcome rule on a number of randomly perturbed replicas of that instance that is polynomial in the number of agents, the desired approximation ratio, and a notion capturing the complexity of the type spaces. When type spaces are large, as in the case of combinatorial auctions, this may become intractable. By contrast, our approach evaluates the outcome rule and the trained payment rule once for a given instance and incurs additional computational costs only during training.

The work of Lahaie [2009, 2010] precedes our work in adopting a kernel-based approach for combinatorial auctions, but focuses not on learning a payment rule for a given outcome rule but rather on solving the winner determination and pricing problem for a given instance of a combinatorial auction. Lahaie introduces the use of kernel methods to compactly represent non-linear price functions, which is also present in our work, but obtains incentive properties more indirectly through a connection between regularization and price sensitivity. The main distinction between the two lines of work is that Lahaie focuses on the design of scalable methods for clearing and pricing approximately welfare-maximizing combinatorial auctions, while we advance a framework for the automated design of payment rules that provide good incentive properties for a given outcome rule, which need not be welfare-maximizing.

Our discussion of k-wise dependent valuations builds on valuation structure for combinatorial auctions introduced by Conitzer et al. [2005] and Abraham et al. [2012]. Our tractable training results rely on connections between k-wise dependent valuations and associative Markov networks [Taskar et al. 2004].

Carroll [2011] and Lubin and Parkes [2012] provide surveys of related work on approximate incentive compatibility, or incentive compatibility in the large-market limit. A fair amount of attention has been devoted to regret-based metrics for quantifying the incentive properties of mechanisms [e.g., Parkes et al. 2001; Day and Milgrom 2008; Lubin 2010; Carroll 2011]. Pathak and Sönmez [2013] provide a qualitative ranking of different mechanisms without payments in terms of the number of manipulable instances. Budish [2011] introduces an asymptotic, absolute design criterion regarding incentive properties in a large replica economy limit. Lubin and Parkes [2009] provide experimental support that relates the divergence between the payoffs in a mechanism and the payoffs in a strategyproof "reference" mechanism to the amount by which agents deviate from truthful bidding in the Bayes-Nash equilibrium of a mechanism.

2. PRELIMINARIES

A mechanism design problem is given by a set $N = \{1, 2, \ldots, n\}$ of *agents* that interact to select an element from a set $\Omega \subseteq \times_{i \in N} \Omega_i$ of *outcomes*, where Ω_i denotes the set of possible outcomes for agent $i \in N$. Agent $i \in N$ is associated with a *type* θ_i from a set Θ_i of possible types, corresponding to the private information available to this agent.

We write $\theta = (\theta_1, \ldots, \theta_n)$ for a profile of types for the different agents, $\Theta = \times_{i \in N} \Theta_i$ for the set of possible type profiles, and $\theta_{-i} \in \Theta_{-i}$ for a profile of types for all agents but i. Each agent $i \in N$ is further assumed to employ preferences over Ω_i, represented by a *valuation function* $v_i : \Theta_i \times \Omega_i \to \mathbb{R}$. We assume that for all $i \in N$ and $\theta_i \in \Theta_i$ there exists an outcome $o \in \Omega$ with $v_i(\theta_i, o_i) = 0$.

A *(direct) mechanism* is a pair (g, p) of an *outcome rule* $g : \Theta \to \times_{i \in N} \Omega_i$ and a *payment rule* $p : \Theta \to \mathbb{R}^n_{\geq 0}$. The intuition is that the agents reveal to the mechanism a type profile $\theta \in \Theta$, possibly different from their true types, and the mechanism chooses outcome $g(\theta)$ and charges each agent i a payment of $p_i(\theta) = (p(\theta))_i$. We assume *quasi-linear preferences*, so the *utility* of agent i with type $\theta_i \in \Theta_i$ given a profile $\theta' \in \Theta$ of revealed types is $u_i(\theta', \theta_i) = v_i(\theta_i, g_i(\theta')) - p_i(\theta')$, where $g_i(\theta) = (g(\theta)_i)$ denotes the outcome for agent i. A crucial property of mechanism (g, p) is that its outcome rule is *feasible*, i.e., that $g(\theta) \in \Omega$ for all $\theta \in \Theta$.

Outcome rule g satisfies *consumer sovereignty* if for all $i \in N$, $o_i \in \Omega_i$, and $\theta'_{-i} \in \Theta_{-i}$, there exists $\theta'_i \in \Theta_i$ such that $g_i(\theta'_i, \theta'_{-i}) = o_i$; and *reachability of the null outcome* if for all $i \in N$, $\theta_i \in \Theta_i$, and $\theta'_{-i} \in \Theta_{-i}$, there exists $\theta'_i \in \Theta_i$ such that $v_i(\theta_i, g_i(\theta'_i, \theta'_{-i})) = 0$.

Mechanism (g, p) is *dominant strategy incentive compatible*, or *strategyproof*, if each agent maximizes its utility by reporting its true type, irrespective of the reports of the other agents, i.e., if for all $i \in N$, $\theta_i \in \Theta_i$, and $\theta' = (\theta'_i, \theta'_{-i}) \in \Theta$, $u_i((\theta_i, \theta'_{-i}), \theta_i) \geq u_i((\theta'_i, \theta'_{-i}), \theta_i)$; it satisfies *individual rationality* (IR) if agents reporting their true types are guaranteed non-negative utility, i.e., if for all $i \in N$, $\theta_i \in \Theta_i$, and $\theta'_{-i} \in \Theta_{-i}$, $u_i((\theta_i, \theta'_{-i}), \theta_i) \geq 0$. Observe that given reachability of the null outcome, strategyproofness implies individual rationality.

A mechanism (g, p) is strategyproof if and only if the payment of an agent is independent of its reported type and the chosen outcome simultaneously maximizes the utility of all agents, i.e., if for every type profile $\theta \in \Theta$,

$$p_i(\theta) = t_i(\theta_{-i}, g_i(\theta)) \qquad \text{for all } i \in N, \text{ and} \tag{1}$$

$$g_i(\theta) \in \arg\max_{o'_i \in \Omega_i} \big(v_i(\theta_i, o'_i) - t_i(\theta_{-i}, o'_i)\big) \qquad \text{for all } i \in N, \tag{2}$$

for a *price function* $t_i : \Theta_{-i} \times \Omega_i \to \mathbb{R}$. This simple characterization is crucial for our approach, providing the basic insight into how to utilize the discriminant function of a classifier as a payment rule. The first property is the *agent-independent* property: conditioned on reports of others, and an outcome, an agent's payment is independent of its own report. The second property is the *agent-optimizing* property: the outcome should maximize an agent's utility given these agent-independent prices and its reported valuation. The first property is necessary since if it is violated, there exists some type vector θ where an agent i can misreport its type and receive the same outcome but a lower payment. The second property is necessary since if it is violated, there exists some type vector θ where an agent i receives a preferred outcome and payment pair by misreporting its type.

Strategyproofness can also be characterized in regard to necessary and sufficient properties of outcome rules alone, and especially through *monotonicity* properties. These properties characterize those outcome rules for which there exists a payment rule such that the outcome rule and payment rule form a strategyproof mechanism [Saks and Yu 2005; Ashlagi et al. 2010]. These monotonicity properties constrain the space of outcome rules for which it is possible to learn a payment rule that provides full strategyproofness to a designed mechanism.

We quantify the degree of strategyproofness of a mechanism in terms of the *regret* experienced by an agent when revealing its true type. The *ex post regret* of agent $i \in N$ in mechanism (g, p), given true type $\theta_i \in \Theta_i$ and reported types $\theta'_{-i} \in \Theta_{-i}$ of the other agents, is

$$rgt_i(\theta_i, \theta'_{-i}) = \max_{\theta'_i \in \Theta_i} u_i\big((\theta'_i, \theta'_{-i}), \theta_i\big) - u_i\big((\theta_i, \theta'_{-i}), \theta_i\big).$$

This is the maximum gain in utility the agent could achieve through a misreport θ'_i, holding the reports of others fixed. Analogously, the *ex post violation of individual rationality* of agent $i \in N$ in mechanism (g, p), given true type $\theta_i \in \Theta_i$ and reported types $\theta'_{-i} \in \Theta_{-i}$ of the other agents, is

$$irv_i(\theta_i, \theta'_{-i}) = |\min(u_i((\theta_i, \theta'_{-i}), \theta_i), 0)|.$$

This quantity is zero when there is no violation of individual rationality (IR) for the agent at this type profile, but positive when the agent's utility is negative for the outcome and payment.

We consider situations where type profiles θ are drawn from a distribution with probability density function, $D : \Theta \to \mathbb{R}$, such that $D(\theta) \geq 0$ and $\int_{\theta \in \Theta} D(\theta) = 1$. Given such a distribution, and assuming that all agents report their true types, the *expected ex post regret* of agent $i \in N$ in mechanism (g, p) is $\mathbb{E}_{\theta \sim D}[rgt_i(\theta_i, \theta_{-i})]$.

Outcome rule g is *agent symmetric* if for every permutation π of agents N, and all types $\theta, \theta' \in \Theta$ such that $\theta_i = \theta'_{\pi(i)}$ for all $i \in N$, $g_i(\theta) = g_{\pi(i)}(\theta')$ for all $i \in N$. This specifically requires that $\Theta_i = \Theta_j$ and $\Omega_i = \Omega_j$ for all $i, j \in N$. Similarly, type distribution D is *agent symmetric* if $D(\theta) = D(\theta')$, for every permutation π of N, and all types $\theta, \theta' \in \Theta$ such that $\theta_i = \theta'_{\pi(i)}$ for all $i \in N$. Given agent symmetry, a price function $t_1 : \Theta_{-1} \times \Omega_i \to \mathbb{R}$ for agent 1 can be used to generate the payment rule p for a mechanism (g, p), with

$$p(\theta) = \big(t_1(\theta_{-1}, g_1(\theta)), t_1(\theta_{-2}, g_2(\theta)), \ldots, t_1(\theta_{-n}, g_n(\theta))\big),$$

so that the expected ex post regret is the same for every agent.

We assume agent symmetry going forward, which precludes outcome rules that break ties based on agent identity, but obviates the need to train a separate classifier for each agent while also providing some benefits in terms of simplifying the presentation of our results.[2]

3. PAYMENT RULES FROM MULTI-CLASS CLASSIFIERS

A *multi-class classifier* is a function $h : X \to Y$, where X is an input domain and Y is a discrete output domain. One could imagine, for example, a multi-class classifier that labels a given image as a dog, cat, or some other animal. In the context of mechanism design, we will be interested in classifiers that take as input a type profile and output an outcome. What distinguishes this from an outcome rule is that we will impose restrictions on the form the classifier can take.

Classification typically assumes an underlying target function $h^* : X \to Y$, and the goal is to learn a classifier h that minimizes disagreements with h^* on an input distribution D_X on X, based only on a finite set of *training data* $\{(x^1, y^1), \ldots, (x^\ell, y^\ell)\} =$

[2]Technically, agent symmetric outcome rules would need to either break ties using randomization or by not allocating anything to agents that are tied. In the former case, the outcome rule would then map to a distribution over outcomes rather than a single outcome. The relation between multi-class classification and mechanism design still holds in this setting, but perfect classification is no longer possible because the outcome rule is not deterministic. We adopt agent symmetry to simplify the presentation of our results, but without agent symmetry, we can still use the same methods to train a separate classifier for each agent based on an agent-specific outcome rule. We also assume agent symmetry and train a single classifier for all agents in the experimental results as ties occur with negligible probability for the settings and outcome rules we study.

$\{(x^1, h^*(x^1)), \ldots, (x^\ell, h^*(x^\ell))\}$ with $x^1, \ldots, x^\ell$ drawn from D_X. This may be challenging because the amount of training data is limited, or because h is restricted to some hypothesis class $\mathcal{H}$ with a certain simple structure, e.g., linear threshold functions. If $h(x) = h^*(x)$ for all $x \in X$, we say that h is a *perfect classifier* for h^*.

We consider classifiers that are defined in terms of a *discriminant function* $f : X \times Y \to \mathbb{R}$, such that

$$h(x) \in \arg\max_{y \in Y} f(x, y)$$

for all $x \in X$. More specifically, we will be concerned with *linear* discriminant functions of the form

$$f_w(x, y) = w^T \psi(x, y)$$

for a weight vector $w \in \mathbb{R}^m$ and a *feature map* $\psi : X \times Y \to \mathbb{R}^m$, where $m \in \mathbb{N} \cup \{\infty\}$. The function ψ maps input and output into an m-dimensional space, which allows non-linear features to be expressed. In general, we allow w to have infinite dimension, while requiring the inner product between w and $\psi(x, y)$ to remain well-defined. Computationally, the infinite-dimensional case is handled through kernels, as described in Section 4.1.1.

3.1. Mechanism Design as Classification

Given an outcome rule g and access to a distribution D over type profiles, our goal is to design a payment rule p that gives the mechanism (g, p) the best possible incentive properties, in the sense of expected regret.

Assuming agent symmetry, we focus on a partial outcome rule $g_1 : \Theta \to \Omega_1$ and train a classifier to predict the outcome to agent 1. To train a classifier, we generate examples by drawing a type profile $\theta \in \Theta$ from distribution D and applying outcome rule g to obtain the target class $g_1(\theta) \in \Omega_1$.

We impose a special structure on the hypothesis class. A classifier $h_w : \Theta \to \Omega_1$ is *admissible* if it is defined in terms of a discriminant function f_w of the form

$$f_w(\theta, o_1) = w_1 v_1(\theta_1, o_1) + w_{-1}^T \psi(\theta_{-1}, o_1) \tag{3}$$

for weights w such that $w_1 \in \mathbb{R}_{>0}$ and $w_{-1} \in \mathbb{R}^m$, and a feature map $\psi : \Theta_{-1} \times \Omega_1 \to \mathbb{R}^m$ for $m \in \mathbb{N} \cup \{\infty\}$. The first term of $f_w(\theta, o_1)$ only depends on the type of agent 1, and increases in its valuation for outcome o_1, while the remaining terms ignore θ_1 entirely.

This restriction to admissible discriminant functions is crucial because it allows us to directly infer agent-independent prices from the discriminant function of a trained classifier. For this, define the *associated price function* of an admissible classifier h_w, as

$$t_w(\theta_{-1}, o_1) = -\frac{1}{w_1} w_{-1}^T \psi(\theta_{-1}, o_1),$$

where we again focus on agent 1 for concreteness. By agent symmetry, we obtain the mechanism (g, p_w) corresponding to classifier h_w, by defining payment rule,

$$p_w(\theta) = \big(t_w(\theta_{-1}, g_1(\theta)), t_w(\theta_{-2}, g_2(\theta)), \ldots, t_w(\theta_{-n}, g_n(\theta))\big).$$

Even requiring admissibility, the hope is that appropriate choices for the feature map ψ can produce rich function spaces, and thus ultimately useful payment rules. Moreover, this admissibility structure can be adopted in the context of structural support vector machines, as discussed in Section 4.1.

3.2. Example: Single-Item Auction

Before proceeding further, we illustrate the ideas developed so far in the context of a single-item auction. In a single-item auction, the type of each agent is a single number, corresponding to its value for the item, and there are two possible allocations from the point

of view of an agent: one where it receives the item, and one where it does not. Formally, $\Theta = \mathbb{R}^n$ and $\Omega_1 = \{0, 1\}$ (agent 1 is allocated, or it is not).

Consider a setting with three agents and a training set:

$$(\theta^1, o_1^1) = ((1, 3, 5), 0), \quad (\theta^2, o_1^2) = ((5, 4, 3), 1), \quad (\theta^3, o_1^3) = ((2, 3, 4), 0),$$

and note that this training set is consistent with an *optimal* outcome rule, i.e., one that assigns the item to an agent with maximum value.

Our goal is to learn an admissible classifier,

$$h_w(\theta) = \arg\max_{o_1 \in \{0,1\}} f_w(\theta, o_1) = \arg\max_{o_1 \in \{0,1\}} w_1 v_1(\theta_1, o_1) + w_{-1}^T \psi(\theta_{-1}, o_1),$$

that performs well on the training set. Since there are only two possible outcomes, the outcome chosen by h_w is simply the one with the larger discriminant. A classifier that is perfect on the training data must therefore satisfy the following constraints:

$$\begin{aligned}
w_1 \cdot 0 + w_{-1}^T \psi((3, 5), 0) &> w_1 \cdot 1 + w_{-1}^T \psi((3, 5), 1), \\
w_1 \cdot 5 + w_{-1}^T \psi((4, 3), 1) &> w_1 \cdot 0 + w_{-1}^T \psi((4, 3), 0), \\
w_1 \cdot 0 + w_{-1}^T \psi((3, 4), 0) &> w_1 \cdot 2 + w_{-1}^T \psi((3, 4), 1).
\end{aligned}$$

This can, for example, be achieved by setting $w_1 = 1$, and

$$w_{-1}^T \psi((\theta_2, \theta_3), o_1) = \begin{cases} -\max(\theta_2, \theta_3) & \text{if } o_1 = 1 \text{ and} \\ 0 & \text{if } o_1 = 0. \end{cases}$$

Recalling our definition of the price function as $t_w(\theta_{-1}, o_1) = -(1/w_1) w_{-1}^T \psi(\theta_{-1}, o_1)$, we see that this choice of w and ψ corresponds to the second-price payment rule.

In practice, we are limited to hypotheses that are linear in features $\psi((\theta_2, \theta_3), o_1)$, and should not expect that the classifier is exact on the training data or generally on the distribution of inputs. Nevertheless, we will see in Section 4.1.1 that through the use of kernels we can adopt choices of ψ that allow for rich, non-linear discriminant functions.

3.3. Perfect Classifiers and Implementable Outcome Rules

We now formally establish a connection between mechanism design and multi-class classification.

THEOREM 3.1. *Let (g, p) be a strategyproof mechanism with an agent symmetric outcome rule g, and let t_1 be the corresponding price function. Then, a perfect admissible classifier h_w for partial outcome rule g_1 exists if* $\arg\max_{o_1 \in \Omega_1} (v_1(\theta_1, o_1) - t_1(\theta_{-1}, o_1)))$ *is unique for every type profile θ.*

PROOF. By the first characterization of strategyproof mechanisms, g must select an outcome that maximizes the utility of agent 1 at the current prices, i.e.,

$$g_1(\theta) \in \arg\max_{o_1 \in \Omega_1} (v_1(\theta_i, o_1) - t_1(\theta_{-1}, o_1)).$$

Consider the admissible discriminant $f_{(1,1)}(\theta, o_1) = v_1(\theta_1, o_1) - t_1(\theta_{-1}, o_1)$, which uses the price function t_1 as its feature map. Clearly, the corresponding classifier $h_{(1,1)}$ maximizes the same quantity as g_1, and the two must agree if there is a unique maximizer. □

The relationship also works in the opposite direction: a perfect, admissible classifier h_w for outcome rule g can be used to construct a payment rule that turns g into a strategyproof mechanism.

THEOREM 3.2. *Let g be an agent symmetric outcome rule, $h_w : \Theta \to \Omega_1$ an admissible classifier, and p_w the payment rule corresponding to h_w. If h_w is a perfect classifier for the partial outcome rule g_1, then mechanism (g, p_w) is strategyproof.*

We prove this result by expressing the regret of an agent in mechanism (g, p_w) in terms of the discriminant function f_w. Let $\Omega_i(\theta_{-i}) \subseteq \Omega_i$ denote the set of partial outcomes for agent i that can be obtained under g given reported types θ_{-i} from all agents but i, keeping the dependence on g silent for notational simplicity.

LEMMA 3.3. *Suppose that agent 1 has type θ_1 and that the other agents report types θ_{-1}. Then the regret of agent 1 for bidding truthfully in mechanism (g, p_w) is*

$$\frac{1}{w_1}\big(\max_{o_1 \in \Omega(\theta_{-1})} f_w(\theta, o_1) - f_w(\theta, g_1(\theta))\big).$$

PROOF. We have

$$\begin{aligned} rgt_1(\theta) &= \max_{\theta_1' \in \Theta_1}\big(v_1(\theta_1, g_1(\theta_1', \theta_{-1})) - p_{w,1}(\theta_1', \theta_{-1})\big) - \big(v_1(\theta_1, g_1(\theta)) - p_{w,1}(\theta)\big) \\ &= \max_{o_1 \in \Omega_1(\theta_{-1})}\big(v_1(\theta_1, o_1) - t_w(\theta_{-1}, o_1)\big) - \big(v_1(\theta_1, g_1(\theta)) - t_w(\theta_{-1}, g_1(\theta))\big) \\ &= \max_{o_1 \in \Omega_1(\theta_{-1})}\big(v_1(\theta_1, o_1) + \frac{1}{w_1} w_{-1}^T \psi(\theta_{-1}, o_1)\big) \\ &\qquad - \big(v_1(\theta_1, g_1(\theta)) + \frac{1}{w_1} w_{-1}^T \psi(\theta_{-1}, g_1(\theta))\big) \\ &= \frac{1}{w_1}\big(\max_{o_1 \in \Omega_1(\theta_{-1})} f_w(\theta, o_1) - f_w(\theta, g_1(\theta))\big). \end{aligned}$$

□

PROOF OF THEOREM 3.2. If h_w is a perfect classifier, then the discriminant function f_w satisfies $\arg\max_{o_1 \in \Omega_1} f_w(\theta, o_1) = g_1(\theta)$ for every $\theta \in \Theta$. Since $g_1(\theta) \in \Omega_1(\theta_{-1})$, we thus have that $\max_{o_1 \in \Omega_1(\theta_{-1})} f_w(\theta, o_1) = f_w(\theta, g_1(\theta))$. By Lemma 3.3, the regret of agent 1 for bidding truthfully in mechanism (g, p_w) is always zero, which means that the mechanism is strategyproof. □

It bears emphasis that classifier h_w is only used to derive the payment rule p_w, while the outcome is still selected according to g.

We might ask whether classifier h_w could be used to obtain an agent symmetric outcome rule g_w, and, since h_w is a perfect classifier for itself, a strategyproof mechanism (g_w, p_w). In particular, for each agent i, the outcome rule g_w would be defined to select the outcome o_i^* that maximizes, $f_w(\theta, o_i) = w_i v_i(\theta_i, o_i) + w_{-i}^T \psi(\theta_{-i}, o_i)$. But the problem is that this need not be feasible: there need not be a set of outcomes, $o^* = (o_1^*, \ldots, o_n^*)$, such that this outcome is itself feasible. For example, in the context of an auction, the outcome rule g_w implied by the trained classifier might seek to give the same item to the more than one agent.

The mechanism that we adopt, namely (g, p_w), has in some sense the opposite problem—it is guaranteed to be feasible because outcome rule g is feasible, but is only strategyproof if h_w is a perfect classifier for g. While the learned payment rule, p_w, always satisfies the agent-independent property (1), the agent-maximizing property (2) (the second requirement for strategyproofness) is violated when $h_w(\theta) \neq g_1(\theta)$.

3.4. Approximate Classification and Approximate Strategyproofness

A perfect admissible classifier for outcome rule g provides a payment rule for a strategyproof mechanism. We now show that this result extends gracefully to situations where no such payment rule is available, by relating the *expected* ex post regret of a mechanism (g, p) to a measure of the generalization error of a classifier for outcome rule g.

Fix a feature map ψ, and denote by $\mathcal{H}_\psi$ the space of all admissible classifiers with this feature map. The *discriminant loss* of a classifier $h_w \in \mathcal{H}_\psi$ with respect to a type profile θ

and an outcome $o_1 \in \Omega_1$ is given by,

$$\Delta_w(o_1, \theta) = \frac{1}{w_1}\big(f_w(\theta, h_w(\theta)) - f_w(\theta, o_1)\big).$$

Intuitively the discriminant loss measures how far, in terms of the normalized discriminant, h_w is from predicting the correct outcome for type profile θ, assuming the correct outcome is o_1. Note that $\Delta(o_1, \theta) \geq 0$ for all $o_1 \in \Omega_1$ and $\theta \in \Theta$, and $\Delta(o_1, \theta) = 0$ if $o_1 = h_w(\theta)$. In addition, $h_w(\theta) = h_{w'}(\theta)$ does not imply that $\Delta_w(o_1, \theta) = \Delta_{w'}(o_1, \theta)$ for all $o_1 \in \Omega_1$: even if two classifiers predict the same outcome, one of them may still be closer to predicting the correct outcome o_1.

The *generalization error* of classifier $h_w \in \mathcal{H}_\psi$ with respect to a type distribution D and a partial outcome rule $g_1 : \Theta \to \Omega_1$, is given by

$$R_w(D, g) = \int_{\theta \in \Theta} \Delta_w\big(g_1(\theta), \theta\big) D(\theta) d\theta.$$

The following result establishes a connection between the generalization error and the expected ex post regret of the corresponding mechanism.

THEOREM 3.4. *Consider an outcome rule g, a space $\mathcal{H}_\psi$ of admissible classifiers, and a type distribution D. Let $h_{w^*} \in \mathcal{H}_\psi$ be a classifier that minimizes generalization error with respect to D and g among all classifiers in $\mathcal{H}_\psi$. Then the following holds:*

(1) If g satisfies consumer sovereignty, then (g, p_{w^*}) minimizes expected ex post regret with respect to D among all mechanisms (g, p_w) corresponding to classifiers $h_w \in \mathcal{H}_\psi$.
(2) Otherwise, (g, p_{w^*}) minimizes an upper bound on expected ex post regret with respect to D amongst all mechanisms (g, p_w) corresponding to classifiers $h_w \in \mathcal{H}_\psi$.

PROOF. For the second property, observe that

$$\begin{aligned}
\Delta_w(g_1(\theta), \theta) &= \frac{1}{w_1}\big(f_w(\theta, h_w(\theta)) - f_w(\theta, g_1(\theta))\big) \\
&= \frac{1}{w_1}\big(\max_{o_1 \in \Omega_1} f_w(\theta, o_1) - f_w(\theta, g_1(\theta))\big) \\
&\geq \frac{1}{w_1}\big(\max_{o_1 \in \Omega(\theta_{-1})} f_w(\theta, o_1) - f_w(\theta, g_1(\theta))\big) = rgt_1(\theta),
\end{aligned}$$

where the last equality holds by Lemma 3.3. If g satisfies consumer sovereignty, then the inequality holds with equality, and the first property follows as well. □

Minimization of expected regret itself, rather than an upper bound, can also be achieved even in the absence of consumer sovereignty (which holds for all the outcome rules studied in this paper) if the learner has access to the set of available outcomes, $\Omega_1(\theta_{-1})$, that are achievable for every $\theta_{-1} \in \Theta_{-1}$.

4. A SOLUTION USING STRUCTURAL SUPPORT VECTOR MACHINES

In this section we discuss the method of *structural support vector machines* (structural SVMs) [Tsochantaridis et al. 2005; Joachims et al. 2009]. In particular, we show how structural SVMs can be adapted for the purpose of learning classifiers with admissible discriminant functions.

4.1. Structural SVMs

Given an input space X, a discrete output space Y, a target function $h^* : X \to Y$, and a set of *training examples* $\{(x^1, h^*(x^1)), \ldots, (x^\ell, h^*(x^\ell))\} = \{(x^1, y^1), \ldots, (x^\ell, y^\ell)\}$, structural SVMs learn a multi-class classifier h that given input $x \in X$ selects an output $y \in Y$ to

maximize $f_w(x,y) = w^T\psi(x,y)$. For a given feature map ψ, the training problem is to find a vector w for which h_w has low generalization error.

For those readers familiar with SVMs (for binary classification), structural SVMs apply similar insights to solve a multi-class classification problem. While SVMs try to find a boundary that separates the positive examples from the negative examples and maximizes the minimum distance (or margin) to any example, structural SVMs try to find weights that separate the discriminant function values for the correct class from the discriminant function values for all other classes as much as possible. By maximizing this re-defined notion of margin, structural SVMs attempt to learn weights which induce discriminant functions with low generalization error.

Given examples $\{(x^1,y^1),\ldots,(x^\ell,y^\ell)\}$, training is achieved by solving the following convex optimization problem:

$$\min_{w,\xi\geq 0} \frac{1}{2}w^Tw + \frac{C}{\ell}\sum_{k=1}^{\ell}\xi^k \qquad \text{(Training Problem 1)}$$
$$\text{s.t. } w^T\big(\psi(x^k,y^k) - \psi(x^k,y)\big) \geq \mathcal{L}(y^k,y) - \xi^k \quad \text{for all } k=1,\ldots,\ell,\ y\in Y$$
$$\xi^k \geq 0 \quad \text{for all } k=1,\ldots,\ell.$$

The goal is to find a weight vector w and slack variables ξ^k such that the objective function is minimized while satisfying the constraints. The learned weight vector w parameterizes the discriminant function f_w, which in turn defines the classifier h_w. The kth constraint states that the value of the discriminant function on (x^k,y^k) should exceed the value of the discriminant function on (x^k,y) by at least $\mathcal{L}(y^k,y)$, where $\mathcal{L}$ is a loss function that penalizes misclassification, with $\mathcal{L}(y,y) = 0$ and $\mathcal{L}(y,y') \geq 0$ for all $y,y' \in Y$. the loss function is optional (since it can be set to 0 everywhere), but is a useful tool to tune the classifiers that are learned. We generally use a 0/1 loss function, but consider an alternative in Section 4.2.2 to improve ex post IR properties. Positive values for the slack variables ξ^k allow the weight vector to violate some of the constraints.

The other term in the objective, the squared norm of the weights, penalizes larger weight vectors. Without this, scaling up the weight vector w can arbitrarily increase the margin between $f_w(x^k,y^k)$ and $f_w(x^k,y)$, and make the constraints easier to satisfy. Smaller values of w, on the other hand, increases the ability of the learned classifier to generalize by decreasing the propensity to over-fit to the training data.

Parameter $C \geq 0$ is a *regularization parameter*: larger values of C encourage small ξ^k and larger w, such that more points are classified correctly, but with a smaller margin (and thus perhaps with less generalization power).

4.1.1. The Feature Map and the Kernel Trick. Given a feature map ψ, the *feature vector* $\psi(x,y)$ for $x \in X$ and $y \in Y$ provides an alternate representation of the input-output pair (x,y). It is useful to consider feature maps ψ for which $\psi(x,y) = \phi(\chi(x,y))$, where $\chi : X\times Y \to \mathbb{R}^s$ for some $s \in \mathbb{N}$ is an *attribute map* that combines x and y into a single *attribute vector*, $\chi(x,y)$, which compactly represents the pair. Given this, function $\phi : \mathbb{R}^s \to \mathbb{R}^m$, for $m > s$, maps the attribute vector to a higher-dimensional space and can introduce additional non-linear interactions between attributes. In this way, SVMs can achieve non-linear classification in the attribute space.

What is commonly described as "feature engineering" occurs in our setting through a combination of designing the attribute map χ and the function ϕ.

The use of kernels allows for a large (even unbounded) m, because $\psi(x,y)$ only appears in the dual of Training Problem 1 within an inner product of the form $\langle\psi(x,y),\psi(x',y')\rangle$, or, for a decomposable feature map, $\langle\phi(q),\phi(q')\rangle$ where $q = \chi(x,y)$ and $q' = \chi(x',y')$ (see [Joachims et al. 2009] for a complete derivation of the dual). For computational

tractability it suffices that this inner product can be computed efficiently, and the kernel "trick" is to choose ϕ such that $\langle \phi(q), \phi(q') \rangle = K(q, q')$ for a simple closed-form function K, which is known as the *kernel*.

Two common kernels are the *polynomial kernel* K_{polyd}, which is parameterized by degree $d \in \mathbb{N}^+$, and the *radial basis function (RBF) kernel* K_{RBF}, which is parameterized by $\gamma = 1/(2\sigma^2)$ for $\sigma \in \mathbb{R}^+$:

$$K_{polyd}(q, q') = (q \cdot q')^d,$$
$$K_{RBF}(q, q') = \exp\left(-\gamma \left(\|q\|^2 + \|q'\|^2 - 2q \cdot q'\right)\right).$$

Both polynomial and RBF kernels use the standard inner product of their arguments, so their efficient computation requires only that $\chi(x, y) \cdot \chi(x, y')$ can be computed efficiently. In our experimental results we adopt the RBF kernel for part of our study on combinatorial auctions, but develop our other experimental results without making use of the kernel trick.

4.1.2. Dealing with an Exponentially Large Output Space. Training Problem 1 has $\Omega(|Y|\ell)$ constraints, where Y is the output space and ℓ the number of training instances, and enumerating all of them is computationally prohibitive when Y is large. Joachims et al. [2009] address this issue for structural SVMs through constraint generation: starting from an empty set of constraints, this technique iteratively adds a constraint that is maximally violated by the current solution until the violation is below a desired threshold ϵ'. Joachims et al. show that this will happen after no more than $O(\frac{C}{\epsilon'})$ iterations, each of which requires $O(\ell)$ (resp. $O(\ell^2)$) time and memory if linear (resp. polynomial or RBF) kernels are used.

However, this approach assumes the existence of an *efficient separation oracle*, which given a weight vector w, an input x^k, and a target y^k, finds an output $y \in \arg\max_{y' \in Y} f_w(x^k, y') + \mathcal{L}(y^k, y')$. The subproblem solved by this separation oracle is referred to as the *separation problem*. The existence of such an oracle remains an open question in application to multi-minded combinatorial auctions; see Section 5.1.3 for additional discussion.

Sometimes the separation problem can be written as a polynomially sized linear program of a particular form. We will see this in the context of succinct, graphical representations of agent valuations in the combinatorial auction domain. In this case, we can modify Training Problem 1 so that constraint generation is not needed, even when the output space is exponential in the problem size [Taskar et al. 2004]. Indeed, adopting the approach of Taskar et al. [2004], suppose that we can write $\max_{y' \in Y} f_w(x^k, y') + \mathcal{L}(y^k, y)$ as a linear program of the form:

$$\begin{aligned} \max \quad & wBz \\ \text{subject to} \quad & z \geq 0, Az \leq b, \end{aligned} \tag{4}$$

where A, B, b are functions of x^k. Assuming that this program is feasible and bounded, we have a dual linear program that attains the same objective value:

$$\begin{aligned} \min \quad & b^T z' \\ \text{subject to} \quad & z' \geq 0, A^T z' \geq (wB)^T. \end{aligned} \tag{5}$$

In this case, we can rewrite Training Problem 1 by replacing the many constraints for a single training example with a single constraint that uses a max function:

$$\begin{aligned} \min_{w, \xi \geq 0} \ & \frac{1}{2} w^T w + \frac{C}{\ell} \sum_{k=1}^{\ell} \xi^k \\ \text{s.t.} \ & w^T \psi(x^k, y^k) + \xi^k \geq \max_{y \in Y} \left(w^T \psi(x^k, y) + \mathcal{L}(y^k, y)\right) \quad \text{for all } k = 1, \ldots, \ell \\ & \xi^k \geq 0 \quad \text{for all } k = 1, \ldots, \ell. \end{aligned}$$

We can now apply the LP formulation for finding the maximum value of $f_w(x^k, y) + \mathcal{L}(x, y)$.

$$\begin{aligned}
\min_{w,\xi\geq 0}\ & \frac{1}{2}w^T w + \frac{C}{\ell}\sum_{k=1}^{\ell}\xi^k \\
\text{s.t. }\ & w^T\psi(x^k, y^k) + \xi^k \geq \max_{z\geq 0, A^k z\leq b^k} wB^k z \quad \text{for all } k = 1, \ldots, \ell \\
& \xi^k \geq 0 \quad \text{for all } k = 1, \ldots, \ell.
\end{aligned}$$

By LP duality, we can replace the max linear program with a min linear program.

$$\begin{aligned}
\min_{w,\xi\geq 0}\ & \frac{1}{2}w^T w + \frac{C}{\ell}\sum_{k=1}^{\ell}\xi^k \\
\text{s.t. }\ & w^T\psi(x^k, y^k) + \xi^k \geq \min_{z\geq 0, (A^k)^T z\geq (wB^k)^T} (b^k)^T z \quad \text{for all } k = 1, \ldots, \ell \\
& \xi^k \geq 0 \quad \text{for all } k = 1, \ldots, \ell.
\end{aligned}$$

We can now drop the min on the right hand side and add the constraints under the min directly into the linear program. This is valid since the only place z occurs is on the right hand side of these constraints. As a result, even without explicitly minimizing, the optimization will choose a value of z that allows for the most flexibility in the left hand side of these constraints.

We therefore have a single, succinct primal convex program even though the number of original constraints was exponentially large:

$$\begin{aligned}
\min_{w,\xi\geq 0}\ & \frac{1}{2}w^T w + \frac{C}{\ell}\sum_{k=1}^{\ell}\xi^k \\
\text{s.t. }\ & w^T\psi(x^k, y^k) + \xi^k \geq (b^k)^T z^k, z^k \geq 0, (A^k)^T z^k \geq (wB^k)^T \quad \text{for all } k = 1, \ldots, \ell \\
& \xi^k \geq 0 \quad \text{for all } k = 1, \ldots, \ell.
\end{aligned}$$

We apply these ideas in Section 5.2 to combinatorial auctions where agents have succinct, graph-based value representations. This allows us to have a scalable training problem even though the winner determination problem remains still NP-hard.

Though we work directly with the features $\psi(x^k, y^k)$ in our experiments, it is still possible to use kernels in conjunction with the succinct formulation of the convex program. This would require working with the dual of the succinct primal convex program.

4.1.3. Required Information. In summary, the use of structural SVMs requires specification of the following:

(1) The input space X, the discrete output space Y, and examples of input-output pairs.
(2) An attribute map $\chi : X \times Y \to \mathbb{R}^s$. This function generates an attribute vector that combines the input and output data into a single object.
(3) A kernel function $K(q, q')$, typically chosen from a well-known set of candidates, e.g., polynomial or RBF. The kernel implicitly calculates the inner product $\langle\phi(q), \phi(q')\rangle$, e.g., between a mapping of the inputs into a high dimensional space.
(4) If the space Y is prohibitively large, a routine that allows for efficient separation, i.e., a function that computes $\arg\max_{y\in Y} f_w(x, y)$ for a given w, x, or a compact representation of the separation problem, enabling a succinct formulation of the training problem in the form of convex optimization.

In addition, the user needs to stipulate particular training parameters, such as the regularization parameter C, and the kernel parameter γ if the RBF kernel is being used.

4.2. Structural SVMs for Mechanism Design

We now specialize structural SVMs such that the learned discriminant function will provide a payment rule, for a given symmetric outcome function g and distribution D. In this application, the input domain X is the space of type profiles Θ, and the output domain Y is the space Ω_1 of outcomes for agent 1.

We construct training data by sampling $\theta \sim D$ and applying g to these inputs: $\{(\theta^1, g_1(\theta^1)), \ldots, (\theta^\ell, g_1(\theta^\ell))\} = \{(\theta^1, o_1^1), \ldots, (\theta^\ell, o_1^\ell)\}$. For admissibility of the learned hypothesis $h_w(\theta) = \arg\max_{o_1 \in \Omega_1} w^T \psi(\theta, o_1)$, we require that

$$\psi(\theta, o_1) = (v_1(\theta_1, o_1), \psi'(\theta_{-1}, o_1))$$

For this reason, we must use an attribute map $\chi' : \Theta_{-1} \times \Omega_1 \to \mathbb{R}^s$ rather than $\chi : \Theta \times \Omega_1 \to \mathbb{R}^s$, and the kernel ϕ' we specify will only be applied to the output of χ'. This results in the following more specialized training problem:

$$\min_{w, \xi \geq 0} \frac{1}{2} w^T w + \frac{C}{\ell} \sum_{k=1}^{\ell} \xi^k \qquad \text{(Training Problem 2)}$$

$$\text{s.t. } (w_1 v_1(\theta_1^k, o_1^k) + w_{-1}^T \psi'(\theta_{-1}^k, o_1^k)) - (w_1 v_1(\theta_1^k, o_1) + w_{-1}^T \psi'(\theta_{-1}^k, o_1)) \geq \mathcal{L}(o_1^k, o_1) - \xi^k$$
$$\text{for all } k = 1, \ldots, \ell,\ o_1 \in \Omega_1$$

$$\xi^k \geq 0 \quad \text{for all } k = 1, \ldots, \ell.$$

If $w_1 > 0$ then the weights w together with the feature map ψ' define a price function $t_w(\theta_{-1}, o_1) = -(1/w_1) w_{-1}^T \psi'(\theta_{-1}, o_1)$ that can be used to define payments $p_w(\theta)$, as described in Section 3.1. In this case, we can also relate the regret in the induced mechanism (g, p_w) to the classification error as described in Section 3.3.

THEOREM 4.1. *Consider training data $\{(\theta^1, o_1^1), \ldots, (\theta^\ell, o_1^\ell)\}$. Let g be an outcome function such that $g_1(\theta^k) = o_1^k$ for all k. Let w, ξ^k be the weight vector and slack variables output by Training Problem 2, with $w_1 > 0$. Consider corresponding mechanism (g, p_w). For each type profile θ^k in the training data,*

$$rgt_1(\theta^k) \leq \frac{1}{w_1} \xi^k$$

PROOF. Consider input θ^k. The constraints in the training problem impose that for every outcome $o_1 \in \Omega_1$,

$$w_1 v_1(\theta_1^k, o_1^k) + w_{-1}^T \psi'(\theta_{-1}^k, o_1^k) - \big(w_1 v_1(\theta_1^k, o_1) + w_{-1}^T \psi'(\theta_{-1}^k, o_1)\big) \geq \mathcal{L}(o_1^k, o_1) - \xi^k$$

Rearranging,

$$\xi^k \geq \mathcal{L}(o_1^k, o_1) + \big(w_1 v_1(\theta_1^k, o_1) + w_{-1}^T \psi'(\theta_{-1}^k, o_1)\big) - \big(w_1 v_1(\theta_1^k, o_1^k) + w_{-1}^T \psi'(\theta_{-1}^k, o_1^k)\big)$$
$$\Rightarrow \xi^k \geq \mathcal{L}(o_1^k, o_1) + f_w(\theta^k, o_1) - f_w(\theta^k, o_1^k)$$

This inequality holds for every $o_1 \in \Omega_1$, so

$$\begin{aligned} \xi^k &\geq \max_{o_1 \in \Omega_1} \big(\mathcal{L}(o_1^k, o_1) + f_w(\theta^k, o_1) - f_w(\theta^k, o_1^k)\big) \\ &\geq \max_{o_1 \in \Omega_1} \big(f_w(\theta^k, o_1) - f_w(\theta^k, o_1^k)\big) \\ &\geq w_1 rgt_1(\theta^k), \end{aligned}$$

where the second inequality holds because $\mathcal{L}(o_1^k, o_1) \geq 0$, and the final inequality follows from Lemma 3.3. This completes the proof. □

We choose not to enforce $w_1 > 0$ explicitly in Training Problem 2, because adding this constraint leads to a dual problem that references ψ' outside of an inner product, and thus makes computation of all but linear or low-dimensional polynomial kernels prohibitively expensive. Instead, in our experiments we simply discard hypotheses where the result of training is $w_1 \leq 0$. This is sensible since the discriminant function value should increase as an agent's value increases, and negative values of w_1 typically mean that the training parameter C or the kernel parameter γ (if the RBF kernel is used) are poorly chosen.

Looking forward to our experiments, this requirement of positive w_1 did not present a practical concern. For example, for multi-minded combinatorial auctions, $1049/1080 > 97\%$ of the trials had positive w_1 for the trained classifier, and for the egalitarian assignment problem all of the trained classifiers had $w_1 > 0$.

4.2.1. Payment Normalization. One issue with the framework as stated is that the payments p_w computed from the solution to Training Problem 2 could be negative. We solve this problem by normalizing payments, using a *baseline outcome* o_b. If there exists a null outcome o', such that $v_1(\theta_1, o') = 0$ for every θ_1, then this outcome provides the baseline. Otherwise, we adopt as the baseline outcome the outcome o_b with the lowest price to agent 1 for a given set of types of other agents. For this, let $t_w(\theta_{-1}, o_1)$ be the price function corresponding to the solution w to Training Problem 2. Adopting the baseline outcome o_b, the *normalized payments* $t'_w(\theta_{-1}, o_1)$, are defined as

$$t'_w(\theta_{-1}, o_1) = \max(0, t_w(\theta_{-1}, o_1) - t_w(\theta_{-1}, o_b)).$$

Even when the baseline outcome is defined as that with the lowest price, it is still only a function of the types of other agents θ_{-1}, and so the prices t'_w remain a function of θ_{-1} and o_1 and are still agent independent.

4.2.2. Individual Rationality Violation. Even after normalization, the learned payment rule p_w may not satisfy individual rationality (IR). Recall that this requires that an agent's payment is no greater than its reported value for the outcome. We offer three solutions to this problem, which can also be used in combination.

Payment offsets. One way to reduce IR violations is to make an additional adjustment to prices, across all type reports, designed to reduce the prices. In particular, for a given offset $\mathit{off} > 0$, and given normalized prices t'_w, we can then further adjust prices by the offset to obtain final prices $t''_w(\theta_{-1}, o_1) = \max(0, t'_w(\theta_{-1}, o_1) - \mathit{off})$. The effect is to leave the price on the baseline outcome unchanged (since its price was already normalized to zero), but to apply the offset where possible to other outcomes.

Although the use of a payment offset decreases the IR violation it might increase regret because of the non-linearity in taking the max with zero. For instance, suppose there are only two outcomes o_{11}, o_{12}, where o_{12} is the null outcome. Suppose agent 1 values o_{11} at 5 and receives the null outcome if he reports truthfully. Suppose further that payments t_w are 7 for o_{11} and 0 for the null outcome. With no payment offset, the agent experiences no regret, since he receives utility 0 from the null outcome, but negative utility from o_{11}. However, if the payment offset is greater than 2, the agent's regret becomes positive (assuming consumer sovereignty), because he could have reported differently and received o_{11} and received positive utility.

Adjusting the loss function $\mathcal{L}$. We incur an IR violation when there is a null outcome o_{null} (for example allocating no items to an agent in a combinatorial auction), such that $g_1(\theta) \neq o_{null}$ and $f_w(\theta, o_{null}) > f_w(\theta, g_1(\theta))$ for some type θ; i.e., the discriminant value of the null outcome is greater than that for the actual outcome selected by the outcome rule. This happens because the discriminant $f_w(\theta, o_1)$ is a scaled version of the agent's utility for outcome o_1 under payments p_w. If the utility for the null outcome is greater than the utility for $g_1(\theta)$, and the payment on null outcomes are zero, then the payment $t_w(\theta_{-1}, g_1(\theta))$ must

be greater than $v_1(\theta_1, g_1(\theta))$ (so that the discriminant value $f_w(\theta, g_1(\theta)) < 0$), causing an IR violation.

Recognizing this, we can discourage these types of errors by modifying the constraints of Training Problem 2: when $o_1^k \neq o_{null}$ and $o_1 = o_{null}$, we can increase $\mathcal{L}(o_1^k, o_1)$ to heavily penalize misclassifications of this type. With a larger $\mathcal{L}(o_1^k, o_1)$, a larger ξ^k will be required if $f_w(\theta, o_1^k) < f_w(\theta, o_{null})$. As with payment offsets, this technique will decrease IR violations but is not guaranteed to eliminate all of them. In our experimental results, we refer to this as the *null loss fix*, and the null loss refers to the value we choose for $\mathcal{L}(o_1^k, o_{null})$ where outcome $o_1^k \neq o_{null}$.

Deallocation. In settings that have a null outcome and are *downward closed* (i.e., settings where a feasible outcome o remains feasible if o_i is replaced with the null outcome), we can also choose to modify the function g to allocate the null outcome whenever the price function t_w creates an IR violation. This reduces ex post regret, and in particular ensures ex post IR for all instances. On the other hand, the total value to the agents necessarily decreases under the modified allocation, and we begin to deviate from the intended outcome rule. In our experimental results, we refer to this as the *deallocation fix*.

5. APPLYING THE FRAMEWORK

In this section, we discuss the application of our framework to three domains: multi-minded combinatorial auctions, combinatorial auctions with k-wise dependent valuations, and egalitarian welfare in the assignment problem.

5.1. Multi-Minded Combinatorial Auctions

A combinatorial auction allocates items $\{1, \ldots, r\}$ among n agents, such that each agent receives a possibly empty subset of the items. The outcome space Ω_i for agent i is the set of all subsets of the r items, and the type of agent i can be represented by a vector $\theta_i \in \Theta_i = \mathbb{R}^{2^r}$ that specifies its value for each possible bundle. The set of possible type profiles is then $\Theta = \mathbb{R}^{2^r n}$, and the value $v_i(\theta_i, o_i)$ of agent i for bundle o_i is equal to the entry in θ_i corresponding to o_i.

We require that valuations are monotone, such that $v_i(\theta_i, o_i) \geq v_i(\theta_i, o_i')$ for all $o_i, o_i' \in \Omega_i$ with $o_i' \subseteq o_i$, and normalized such that $v_i(\theta_i, \emptyset) = 0$. Assuming agent symmetry and adopting the view of agent 1, the partial outcome rule $g_1 : \Theta \to \Omega_1$ specifies the bundle $g_1(\theta)$ allocated to agent 1. We require feasibility of outcome rules, so that no item is allocated more than once.

In a multi-minded CA, each agent is interested in at most κ bundles for some constant κ. The special case where $\kappa = 1$ is the well studied problem of *single-minded* CAs. We choose to study multi-minded CAs rather than single-minded CAs because they provide an example for which truthful, algorithmic mechanism design is not well understood. We choose to study multi-minded CAs in particular, as an example of a multi-parameter mechanism design problem, because the valuation profiles and thus the training data can be represented in a compact way. In the case of multi-minded CAs, this is by explicitly writing down the valuations for the bundles in which each agent is interested. In addition, the inner products between valuation profiles, which are required to apply the kernel trick, can be computed in constant time.

5.1.1. Attribute Maps. To apply structural SVMs to multi-minded CAs, we need to specify an appropriate attribute map χ'. The approach that we take in choosing an attribute map is to recognize that the attributes should expose to the classifier enough information about the valuations of agents 2 through n to allow the discriminant function to accurately rank the different bundles that could be allocated to agent 1. In particular, the classifier is accurate when the discriminant function assigns the highest score to the bundle that is allocated to agent 1 by the outcome rule.

Conceptually, we find it helpful in this exercise in feature engineering to conceptualize an outcome rule as maximizing some objective function $\tilde{f}(\theta, o)$, for type profile $\theta \in \Theta$ and outcome $o \in \Omega$, where this objective function might approximate social welfare, or approximate max-min value. Given this, and the structure of discriminant function (3), then the attribute map should expose attributes that allow for the accurate estimation of the optimal objective value, when the outcome is restricted to assign bundle o_1 to agent 1.[3] In this sense, a good attribute map represents, perhaps in summary form, the valuation functions of other agents given that bundle o_1 has been assigned to agent 1.

With this in mind, we adopt two simple attribute maps $\chi_1' : \Theta_{-1} \times \Omega_1 \to \mathbb{R}^{2^r(2^r(n-1))}$ and $\chi_2' : \Theta_{-1} \times \Omega_1 \to \mathbb{R}^{2^r(n-1)}$ in our experiments, defined as follows:

$$\chi_1'(\theta_{-1}, o_1) = \begin{bmatrix} 0 \\ \cdots \\ 0 \\ \theta_{-1} \\ 0 \\ \cdots \\ 0 \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}0\\\cdots\\0\end{matrix}}\right\} dec(o_1)(2^r(n-1)) \\ \\ \left.\vphantom{\begin{matrix}0\\\cdots\\0\end{matrix}}\right\} (2^r - dec(o_1) - 1)(2^r(n-1)) \end{matrix}, \quad \chi_2'(\theta_{-1}, o_1) = \begin{bmatrix} \theta_2 \setminus o_1 \\ \theta_3 \setminus o_1 \\ \cdots \\ \theta_n \setminus o_1 \end{bmatrix},$$

where $dec(o_1) = \sum_{j=1}^{r} 2^{j-1} \mathbf{I}_{j \in o_1}$ is a decimal index of bundle o_1, where $\mathbf{I}_{j \in o_1} = 1$ if $j \in o_1$ and $\mathbf{I}_{j \in o_1} = 0$ otherwise, and $\theta_i \setminus o_1$ denotes the valuation function that is obtained by setting the value on all bundles that are non-disjoint with o_1 to zero.

Attribute map χ_1' stacks the vector θ_{-1} (with $2^r(n-1)$ entries), which represents the valuations of all agents except agent 1, with zero vectors of the same dimension, where the position of θ_{-1} is determined by the index of bundle o_1.

We view attribute map χ_1' as providing a baseline, since no effort is made to encode the effect of assigning bundle o_1 to agent 1 on the valuations of the other agents. Rather, the choice of bundle o_1 is encoded only indirectly, through the position of valuations θ_{-1} in the attribute vector. An undesirable side effect is that two training instances in which bundle o_1 differs only slightly will have completely distinct sets of non-zero attributes. We might expect this to reduce the generalization power of the classifier.

In comparison, attribute map χ_2' is designed to encode very explicitly the effect of assigning bundle o_1 to agent 1 on the valuations of other agents. In particular, this attribute vector stacks vectors $\theta_i \setminus o_1$, which are obtained from valuation type θ_i by setting the entries for all bundles that share one or more items with bundle o_1 to zero. This captures the fact that another agent cannot be allocated a bundle that intersects with o_1.

Both χ_1' and χ_2' are defined for a particular number of items and agents, and in our experiments we train a different classifier for each number of agents and items. An attractive alternative to adopt in practice would be to pad out items and agents by setting bids to zero, allowing a single classifier to be trained.

5.1.2. Efficient Computation of Inner Products. Efficient computation of inner products is possible for both χ_1', χ_2'. A full discussion of the approach that we take for this is provided in Appendix A.

[3]In the single-item example in Section 3.2, we could have obtained an exact classifier by setting $w_{-1}^T \psi((\theta_2, \theta_3), o_1) = 0$ if $o_1 = 1$ and $\max(\theta_2, \theta_3)$ if $o_1 = 0$, and then obtaining the second-price payment rule by first normalizing the payment rule as described in Section 4.2.1. In this way, the discriminant rule $f_w(\theta, o_1)$ would be exactly the objective value for the optimal assignment that is constrained to respect assignment o_1 to agent 1.

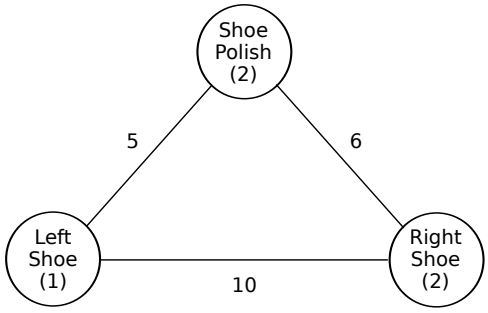

Fig. 1. An example of a 2-wise dependent valuation. The values listed in the nodes give the agent's weights for the corresponding items. Each item has some small value on its own, but complementarities exist between pairs of items which give added utility to the agent. Note that while this graph is complete, this is not necessary. Absent edges are assumed to have weight 0.

5.1.3. Dealing with an Exponentially Large Output Space. Recall that Training Problems 1 and 2 have constraints for every training example (θ^k, o_1^k) and every possible bundle of items $o_1 \in \Omega_1$. For CAs, there will be exponentially many such bundles. In lieu of an efficient separation oracle, a workaround exists when the discriminant function ensures that the induced prices weakly increase as items are added to a bundle. Given this property of *item monotonicity*, it suffices to include constraints for bundles that have a strictly larger value to the agent than any of their respective subsets. Coupled with the assumption that valuations in CAs are monotone, and the admissibility property of the discriminant function, no other bundles can have a greater discriminant value than these bundles.

But we are not able to impose item monotonicity directly on the training problem with a small number of constraints.[4] For this reason, the baseline experimental results in Section 6 do not assume item monotonicity, and instead use an inefficient separation oracle, that simply iterates over all possible bundles $o_1 \in \Omega_1$.

An alternative that we have also studied is to optimistically assume item monotonicity, and only include the constraints associated with bundles that are explicit in agent valuations. We also present experimental results that test this optimistic approach, and while there is a degradation in performance, results are mostly comparable. This provides a useful approach to scaling up training for representation languages such as the XOR representation adopted for multi-minded CAs for which it is simple to identify the small set of bundles that are candidates for maximizing the discriminant function (= agent utility.)

5.2. Combinatorial Auctions with Positive k-wise Dependent Valuations

We also study combinatorial auctions where agents have positive k-wise dependent valuations [Conitzer et al. 2005]. This setting allows us to apply the ideas discussed in Section 4.1.2 to attain a polynomial time training formulation despite the exponential size of Ω_1.

When an agent has a k-wise dependent valuation, the agent's valuation is described by a hypergraph $G = (R, E)$ where R is a set of nodes and E is a set of hyperedges of size at most k. The nodes in the graph correspond to the items being auctioned (which is why we use $j \in R$ to refer to both nodes and items), and the hyperedges to groups of these items. These

[4]For polynomial kernels and certain attribute maps, a possible sufficient condition for item monotonicity is to force the weights w_{-1} to be negative. However, as with the discussion of enforcing $w_1 > 0$ directly, these weight constraints do not dualize conveniently and results in the dual formulation to no longer operate on inner products $\langle \psi'(\theta_{-1}, o_1), \psi'(\theta'_{-1}, o'_1) \rangle$. As a result, we would be forced to work in the primal, and incur extra computational overhead that increases polynomially with the kernel degree d. We have performed some preliminary experiments with polynomial kernels, but we have not looked into reformulating the primal to enforce item monotonicity.

nodes and hyperedges are each assigned weights $z_1(j)$ and $z_1(e)$ respectively. An agent's value for a subset of items $o_1 \in \Omega_1$ is the sum of the weights of nodes and hyperedges contained in o_1, i.e., $\sum_{j \in R, j \in o_1} z_1(j) + \sum_{e \in E, e \subseteq o_1} z_1(e)$. Figure 1 gives a pictorial view of a simple 2-wise valuation over 3 items.

A *positive k-wise dependent valuation* adds the restriction that hyperedge weights are positive. This restriction is required for our results, and is also studied by Abraham et al. [2012]. This forces agent valuations to be supermodular, i.e., $v_i(\theta_i, o_1 \cap o_2) + v_i(\theta_i, o_1 \cup o_2) \geq v_i(\theta_i, o_1) + v_i(\theta_i, o_2)$ for all sets of items $o_1, o_2 \in \Omega_1$. When we have multiple agents, we use $z_i(j)$ and $z_i(e)$ to denote the weights that agent i assigns to nodes j and edges e. For convenience, $z_i(e)$ for an edge not in the agent's edge set is defined to be 0. If we are given the agent's type θ_i, then it can be convenient to write $z_i(\theta_i, j)$ or $z_i(\theta_i, e)$ to represent the weights in the agent's underlying graph when its valuation function is θ_i.

Though these valuations are very different from the multi-minded valuations we discussed earlier, the winner determination problem for positive k-wise dependent valuations is still NP-hard when $k = 2$ and hence for any values of $k > 1$ [Conitzer et al. 2005]. Because the winner determination problem is NP-hard, we seek to learn a payment rule for a greedy allocation algorithm.

Going forward, we specialize to the case of $k = 2$, which represents the case where the agent's hypergraph is just a graph. For this case, it is possible to make Training Problem 1 tractable by carefully choosing the attribute map. We discuss extensions to $k > 2$ at the end of Section 5.2.3.

5.2.1. A Greedy Algorithm. We first introduce a simple greedy algorithm, that tries to find an allocation with good welfare. We use this greedy algorithm both in defining the attribute map, and as an outcome rule in our experimental results.

Let $R = \{1, \ldots, r\}$ denote the set of all items. Given some subset of items $R' \subseteq R$, the greedy algorithm orders the items by index and assigns the items incrementally. At each step, the algorithm computes the gain in welfare of assigning the item to each agent and chooses the agent that provides the maximum gain in welfare. Note that if an item j has been assigned to an agent i, then when considering the assignment for item k the gain in welfare of assigning it to agent i includes agent i's node weight for item k as well as agent i's edge weight for edge (j, k) (if the edge exists in the agent's valuation graph). We let $\mathsf{GREEDY}_i(R')$ denote agent i's allocation when this greedy algorithm is run on R'.

5.2.2. A Concrete Example. To clarify the construction, we introduce a simple example where agents have 2-wise dependent valuations. Consider a setting where we have 3 agents and 3 items. We denote the agents and items using indices $1, 2, 3$ but the association should be clear from context. The agents have the following 2-wise dependent valuations:

$$z_1(1) = 1, z_1(2) = 4, z_1(3) = 2, z_1(1,2) = 4$$
$$z_2(1) = 2, z_2(2) = 6, z_2(3) = 2, z_2(2,3) = 3, z_2(1,3) = 6$$
$$z_3(1) = 5, z_3(2) = 3, z_3(3) = 1, z_3(1,2) = 2, z_3(1,3) = 7$$

Applied to this example, the greedy algorithm first considers the assignment of item 1. Agent 3 has the highest value, so 1 goes to agent 3. We then consider item 2. The gain in giving this to agent 1 is 4, the gain to agent 2 is 6, and the gain to agent 3 is $3 + 2 = 5$ (for agent 3, we add in both $z_3(2)$ and $z_3(1,2)$ since 1 was given to agent 3). As a result, agent 2 has the highest gain and we give the item to agent 2. Then for item 3, the gains are 2, $2 + 3 = 5$, and $1 + 7 = 8$ respectively. As a result, item 3 is assigned to agent 3.

5.2.3. Attribute Map. Before defining our attribute map, we provide some intuition for why we use our particular attribute map. Our goal is to apply the techniques described in Section

4.1.2, which require us to write $\max_{o_1' \in \Omega_1} f_w(\theta, o_1') + \mathcal{L}(o_1', o_1)$ as a linear program. Going forward, we assume that $\mathcal{L}(o_1', o_1)$ is 0 everywhere to simplify the presentation.[5]

At first glance, Ω_1 has size that is exponential in the number of items being allocated. It may be possible to write an integer program that solves this maximization, but we require a linear program to apply the ideas of Section 4.1.2. We rely on a result of Taskar et al. [2004] for Markov Random Fields, which, when translated to our setting, shows that if a single agent has what we call a *semi-positive* 2-wise dependent valuation, then it is possible to find the bundle of items that maximizes the agent's utility. Of course if an agent's valuation is positive 2-wise dependent, then the value maximizing bundle is always the set of all items. A semi-positive 2-wise dependent valuation allows an agent to have a weight for not receiving an item or not receiving any item of a given pair of items, and the weights associated with receiving and not receiving a single item can be negative (though the weights for receiving both items in a pair or not receiving any item in a pair must be non-negative). More concretely, a semi-positive 2-wise dependent valuation can be written as:

$$v(\theta_1, o_1) = \sum_{j \in o_1} z_1(j) + \sum_{\substack{1 \le j_1 < j_2 \le r, \\ \{j_1, j_2\} \subseteq o_1}} z_1(j_1, j_2) + \sum_{j \notin o_1} z_1'(j) + \sum_{\substack{1 \le j_1 < j_2 \le r, \\ \{j_1, j_2\} \cap o_1 = \emptyset}} z_1'(j_1, j_2),$$

where $z_1(j)$ and $z_1'(j)$ can take on negative values, and $z_1(j_1, j_2)$ and $z_1'(j_1, j_2)$ are non-negative. While z_1 is active when an item or a pair of items is included in o_1, z_1' is active when an item is not in o_1 or neither of a pair of items is in o_1. This flexibility allows for richer valuation functions which are not permitted with positive 2-wise valuations.

If an agent's valuation is semi-positive 2-wise dependent, then though Ω_1 is of exponential size in the number of items, there is a polynomially sized linear program that finds the bundle o_1 that maximizes $v(\theta_1, o_1)$.

Our goal in defining an attribute map χ_3', is therefore to convert $f_w(\theta, o_1')$ into a semi-positive 2-wise dependent valuation. It is informative to write out $f_w(\theta, o_1')$ when specialized to the case where agents have positive 2-wise dependent valuations:

$$\begin{aligned} f_w(\theta, o_1) &= w_1 v(\theta_1, o_1) - w_{-1}^T \chi_3'(\theta_{-1}, o_1) \\ &= w_1 \sum_{j \in R} \left(z_1(\theta_1, j)\, \mathbb{I}(j \in o_1) + \sum_{1 \le j_1 < j_2 \le r} z_1(\theta_1, (j_1, j_2))\, \mathbb{I}(\{j_1, j_2\} \subseteq o_1) \right) - w_{-1}^T \chi_3'(\theta_{-1}, o_1). \end{aligned}$$

The part representing agent 1's value for o_1 already has the structure of a positive 2-wise dependent valuation (after all, we have assumed that all agents have this valuation structure). What remains is to design χ_3' so that the entire expression, when summed together, resembles a semi-positive 2-wise dependent valuation. If we can accomplish this, then it will be possible to write the maximization $\max_{o_1 \in \Omega_1} f_w(\theta_1, o_1)$ as a linear program. The trick is to construct the attribute map in a way such that the right hand side of the above expression decomposes into a sum over individual terms, each of which corresponds to a weight for a node or an edge in a semi-positive 2-wise dependent valuation.

In addition to ensuring this structure for our attribute map, we still want the attribute map to capture the effect of an assignment of bundle o_1 to agent 1 on the valuations of the other agents, so that the total value of the outcome rule constrained to allocate o_1 to agent 1 can be estimated. In this case, we use the greedy outcome rule in an explicit way to define our attribute map.

[5]As will become clear in the discussion below, we only require that $\mathcal{L}(o_1', o_1)$ be expressed as a sum of products where each product consists of a weight multiplied by: a.) an indicator of whether o_1 contains a given subset of items; or b.) an indicator of whether o_1 does not intersect a given subset of items. As a result, we can adjust the null loss by using an indicator for o_1 not intersecting the entire set of items.

The attribute map $\chi_3'(\theta_{-1}, o_1)$ maps from $\Theta_{-1} \times \Omega_1 \to \mathbb{R}^{2r+r(r-1)}$. For each possible item $j \in \{1, \ldots, r\}$ that can be in bundle o_1, we include two entries in the attribute vector $\chi_3'(\theta_{-1}, o_1)$:

$$\mu_j(0) \cdot \mathbb{I}(j \notin o_1), \quad \mu_j(1) \cdot \mathbb{I}(j \in o_1),$$

where $\mathbb{I}$ is an indicator variable. $\mu_j(0)$ is designed to represent the "gain" to others of not allocating item j to agent 1. We calculate this directly as $z_i(j)$ in the case that some agent $i \neq 1$ is allocated item j in $\mathsf{GREEDY}_i(R)$, otherwise we set the value $\mu_j(0)$ to 1. Value $\mu_j(1)$ does the opposite, approximating the "cost" to others of allocating item j to agent 1. We calculate this as $\mathsf{welfare}(\mathsf{GREEDY}^{-1}(R)) - \mathsf{welfare}(\mathsf{GREEDY}^{-1}(R \setminus \{j\})))$, where $\mathsf{welfare}$ returns the total value of the assignment and GREEDY^{-1} is GREEDY with agent 1 removed.

We also include two entries in attribute vector $\chi_3'(\theta_{-1}, o_1)$ for each pair of items j_1, j_2:

$$\mu_{j_1,j_2}(0) \cdot \mathbb{I}(\{j_1, j_2\} \cap o_1 = \emptyset), \quad \mu_{j_1,j_2}(1) \cdot \mathbb{I}(\{j_1, j_2\} \subseteq o_1),$$

where $\mu_{j_1,j_2}(0)$ and $\mu_{j_1,j_2}(1)$ indicate the value to the other agents when agent 1 is not assigned either j_1 or j_2 and assigned both j_1 and j_2 respectively. Note that we do not specify values for the cases where exactly one of j_1, j_2 is contained in o_1 as these types of weights are not permitted in a semi-positive 2-wise dependent valuation.

The values for μ_{j_1,j_2} are obtained by considering the allocation of GREEDY^{-1} when applied to all items, and considering whether items j_1 and j_2 are assigned to the same agent i. If they are not, then they are set to zero. Otherwise, they are set to $-z_i(j_1, j_2)$. The negative sign here is important for tractability since only non-negative edge weights are permitted in a semi-positive 2-wise dependent valuation. The intuition for why we do not make $\mu_{j_1,j_2}(1)$ equal to the "cost" of allocating items j_1, j_2 is that if $\{j_1, j_2\} \subseteq o_1$ then this cost is already accounted for in $\mu_{j_1}(1)$ and $\mu_{j_2}(1)$. In fact, the cost is double-counted since in both $\mathsf{GREEDY}^{-1}(R \setminus \{j_1\})$ and $\mathsf{GREEDY}^{-1}(R \setminus \{j_2\})$ no agents can derive value from edge (j_1, j_2) since one of the items is missing in both cases.

Returning to our example from Section 5.2.2, recall that when run on all agents, the greedy algorithm gives items 1 and 3 to agent 3 and item 2 to agent 2. The total welfare in this case is 19. The total value to agents 2 and 3 is also 19 since agent 1 does not receive any items. In this case, we then have the following attribute values:

— $\mu_1(0)$: Agent 3 receives item 1, so this is set to $z_3(1) = 5$.
— $\mu_2(0)$: Agent 2 receives item 2, so this is set to $z_2(2) = 6$.
— $\mu_3(0)$: Agent 3 receives item 3, so this is set to $z_3(3) = 1$.
— $\mu_1(1)$: We consider the greedy allocation where item 1 cannot be allocated. The greedy algorithm gives item 2 to agent 2, and then item 3 to agent 2 as well (since the gain will be 5 versus 1). As a result, the total welfare is 11. The welfare difference for the other agents is 8, so $\mu_1(1) = 8$.
— $\mu_2(1)$: Without item 2, the greedy algorithm gives item 1 to agent 3 and then item 3 to agent 3 as well (gain of 8 for agent 3 versus gain of 2 for agent 1). The total welfare is 13, so $\mu_2(1) = 19 - 13 = 6$.
— $\mu_3(1)$: Without item 3, item 1 goes to agent 3 and item 2 goes to agent 2. The total welfare is 11, so $\mu_3(1) = 19 - 11 = 8$.
— $\mu_{1,2}(0), \mu_{1,2}(1)$: The original allocation allocates items 1 and 2 to different agents, so these values are 0.
— $\mu_{1,3}(0), \mu_{1,3}(1)$: Items 1 and 3 are allocated to agent 3, so this is set to $-z_3((1,3)) = -7$.
— $\mu_{2,3}(0), \mu_{2,3}(1)$: The original allocation allocates items 2 and 3 to different agents, so these values are 0.

A useful way to think of this attribute map χ_3' is that it modifies agent 1's 2-wise valuation, while still maintaining the structure of a semi-positive 2-wise dependent valuation.

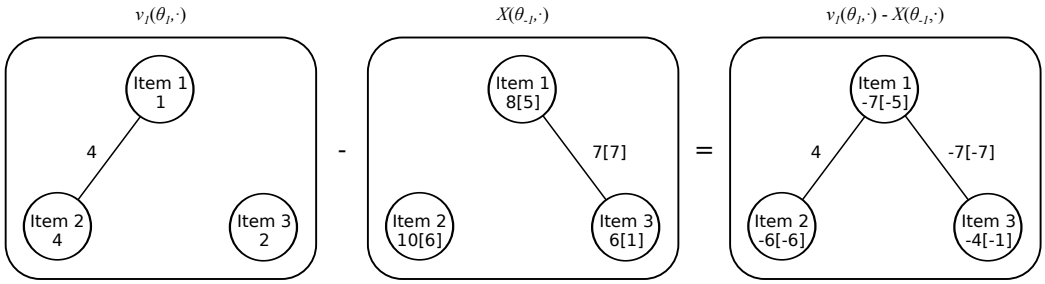

Fig. 2. A pictorial representation of the attribute map χ_3' for our concrete example. We make the attribute map χ_3' resemble a 2-wise dependent valuation (with weights for not being assigned a node and not being assigned any of the items for an edge) so that when combined with agent 1's valuation, we have a modified single-agent problem. The weights are not shown here, but they would be multiplied by the values in χ_3' as illustrated in the graph.

Pictorially, we can think of this as combining agent 1's valuation graph with the valuation graph induced by the feature map. See Figure 2 for an illustration. While we specialize to the case of positive 2-wise dependent valuations, the same approach should be applicable to positive k-wise dependent valuations. Semi-positive k-wise dependent valuations, where an agent can have weights for hyperedges, can also be tractably optimized, and a similar approach can be applied where the attribute vector is carefully chosen so that $f_w(\theta_1, o_1)$ looks like a semi-positive k-wise dependent valuation.

5.2.4. A Tractable Training Problem. We have now given the attribute map χ_3', with an eye on making $f_w(\theta_1, o_1)$ look like a semi-positive 2-wise dependent valuation. Before proving the main result of this section, namely that we can solve $\max_{o_1 \in \Omega_1} f_w(\theta_1, o_1)$ in polynomial time, we need to impose positivity constraints on certain elements of the vector w. This restriction of the space of possible weights enables us to make sure that $f_w(\theta_1, o_1)$ is a semi-positive 2-wise dependent valuation at the possible loss of some pricing accuracy. It also prevents us from using the kernel trick over the positive-restricted weights, although we can still use kernels on the rest. In the present analysis we choose not to add this complexity and work with a linear kernel only.

Before proving our theorem, we formally show that we can find an agents' maximum value if the agent as a semi-positive 2-wise dependent valuation.

LEMMA 5.1. *If an agent has a semi-positive 2-wise dependent valuation, then it is possible to find the agent's value-maximizing bundle in time polynomial in r, the number of items in the auction.*

PROOF. Our proof relies on a connection between semi-positive 2-wise dependent valuations and Markov networks. Finding the value maximizing bundle for an agent with a semi-positive 2-wise dependent valuation is equivalent to finding a *maximum a posteriori* assignment on a particularly defined Markov network. Consider a Markov network with a node for each item and edges between pairs of nodes. Let each node be a binary variable, and let the node potentials be defined based on the coefficients in the above maximization. The potential values for setting a node j to be 0 or 1 be $\exp(z_1(j))$ and $\exp(z_1'(j))$ and the potential values for an edge j_1, j_2 being set to $0,0$ and $1,1$ be $\exp(z_1(j_1, j_2))$, $\exp(z_1'(j_1, j_2))$ respectively. The potential values for setting edges to $0,1$ and $1,0$ are set to $1 = \exp(0)$. Finding the value-maximizing bundle for an agent with a semi-positive 2-wise dependent valuation defined by z_1 and z_1' is equivalent finding a *maximum a posteriori* assignment in the Markov network we have defined.

Applying this connection and using the ideas of Taskar et al. [2004], we see that our maximization problem can be solved by the following integer program. The integer pro-

gram has a binary variable corresponding to each of the indicator variables in the above maximization.

$$\begin{aligned}
\max \quad & \sum_{j\in R} z_1(j)I_{j,1} + \sum_{j\in R} z_1'(j)I_{j,0} + \sum_{1\le j_1<j_2\le r} z_1(j_1,j_2)I_{j_1,j_2,1} + \sum_{1\le j_1<j_2\le r} z_1'(j_1,j_2)I_{j_1,j_2,0} \\
\text{s.t.} \quad & I_{j,0} + I_{j,1} = 1 \quad \text{for all } j \in R \\
& I_{j_1,j_2,p} \le I_{j_1,p}, I_{j_1,j_2,p} \le I_{j_2,p} \quad \text{for all } 1 \le j_1 < j_2 \le r, p \in \{0,1\} \\
& I_{j,p} \in \{0,1\} \quad \text{for all } j \in R, p \in \{0,1\} \\
& I_{j_1,j_2,p} \in \{0,1\} \quad \text{for all } 1 \le j_1 < j_2 \le r, p \in \{0,1\}.
\end{aligned}$$

The first set of constraints ensures that exactly one of $I_{j,0}$ and $I_{j,1}$ is active. The second set of constraints ensures that $I_{j_1,j_2,p}$ is active if and only if $I_{j_1,p}$ and $I_{j_2,p}$ are active. Note that the if direction follows because the objective coefficients of $I_{j_1,j_2,p}$ are non-negative, so the second set of constraints will be tight if $I_{j_1,p}$ and $I_{j_2,p}$ are both set to 1. Therefore, the value of the objective corresponds to the single agent's value for o, where o consists of the items j for which $I_{j,1}$ is set to one.

To complete the proof, we apply Theorem 3.1 from Taskar et al. [2004] to show that the LP relaxation of this integer program is integral if $z_1(j_1,j_2)$ and $z_1'(j_1,j_2)$ are non-negative. □

THEOREM 5.2. *When agents have positive 2-wise dependent valuations and we use attribute map χ_3' without a kernel, then we can solve the structural SVM training problem (with added constraints on the weight vector w discussed above) in time polynomial in r, the number of items in the auction and n, the number of agents.*

PROOF. To prove this theorem, we just need to show that $f_w(\theta_1, o_1)$ can be written as $v'(\theta_1, o_1)$ where v' is a semi-positive 2-wise dependent valuation.

We observe that $\chi_3'(\theta_{-1}, o_1)$ is a vector with $2r + r(r-1)$ elements. Therefore, the weight vector w_{-1} will have the same number of elements. We index elements of these vectors using notation similar to the notation we use for $\chi_3'(\theta_{-1}, o_1)$. That is, we let $w_j(p)$ correspond to the attribute term that includes $\mu_j(p)$, where $p \in \{0,1\}$. Similarly, we let $w_{j_1,j_2}(p)$ correspond to the attribute term that includes $\mu_{j_1,j_2}(p)$, where $p \in \{0,1\}$.

In the primal formulation of Training Problem 1, we add the constraints that $w_{j_1,j_2}(p) \ge 0$ for $p \in \{0,1\}$ and all j_1, j_2. While not strictly necessary, we also impose that $w_1 = 1$ (as we are working with the primal formulation, the enforcement of such a constraint is available to us; alternatively, we could forgo this constraint and operate in the dual, enabling the use of kernels over the unconstrained attributes of the feature map).

$$\begin{aligned}
f_w(\theta, o_1) &= v(\theta_1, o_1) - w_{-1}^T \chi_3'(\theta_{-1}, o_1) \\
&= \sum_{j\in R} z_1(\theta_1, j)\mathbb{I}(j \in o_1) + \sum_{1\le j_1<j_2\le r} z_1(\theta_1, (j_1,j_2))\mathbb{I}(\{j_1,j_2\} \subseteq o_1) - \\
&\quad \sum_{j\in R} (w_j(0)\mu_j(0)\mathbb{I}(j \notin o_1) + w_j(1)\mu_j(1)\mathbb{I}(j \in o_1)) - \\
&\quad \sum_{1\le j_1<j2\le r} (w_{j_1,j_2}(0)\mu_{j_1,j_2}(0)\mathbb{I}(\{j_1,j_2\} \cap o_1 = \emptyset) - \\
&\quad \sum_{1\le j_1<j2\le r} (w_{j_1,j_2}(0)\mu_{j_1,j_2}(0) + w_{j_1,j_2}(1)\mu_{j_1,j_2}(1)\mathbb{I}(\{j_1,j_2\} \subseteq o_1)) \\
&= \sum_{j\in R} w_j(0)\mu_j(0)\mathbb{I}(j \notin o_1) + \sum_{j\in R} \left(z_1(\theta_1, j) + w_j(1)\mu_j(1)\right)\mathbb{I}(j \in o_1) +
\end{aligned}$$

$$\sum_{1\leq j_1<j_2\leq r}(-w_{j_1,j_2})\mu_{j_1,j_2}(0)\mathbb{I}(\{j_1,j_2\}\cap o_1=\emptyset)+$$
$$\sum_{1\leq j_1<j_2\leq r}(z_1(\theta_1,(j_1,j_2))-w_{j_1,j_2}\mu_{j_1,j_2}(1))\mathbb{I}(\{j_1,j_2\}\subseteq o_1).$$

Because we impose the constraints that w_{j_1,j_2} are non-negative, the coefficients of the indicator variables for the edges (j_1,j_2) will be positive. Indeed, as we defined in Section 5.2.3, $\mu_{j_1,j_2}(0)=\mu_{j_1,j_2}(1)\leq 0$. Combining this with the assumption that $z_1(\theta_1,(j_1,j_2))\geq 0$ and our constraint that $w_{j_1,j_2}\geq 0$, we see that the coefficients of the indicator variables for the edges (j_1,j_2) are positive. Having argued that the coefficients of the indicator variables for edges are positive, it is straightforward to conclude that there exists a semi-positive 2-wise dependent v' such that $f_w(\theta_1,o_1)=v'(\theta_1,o_1)$.

We can now draw a connection to Markov networks. The k-wise Maximization Problem is equivalent to finding a *maximum a posteriori* assignment on a particularly defined Markov network. Consider a Markov network with a node for each item and edges between pairs of nodes. Let each node be a binary variable, and let the node potentials be defined based on the coefficients in the above maximization. The potential values for setting a node j to be 0 or 1 are $\exp(w_j(0)\mu_j(0))$ and $\exp(z_1(\theta_1,j)+w_j(1)\mu_j(1))$ respectively. The potential values for an edge j_1,j_2 being set to $0,0$ and $1,1$ are $\exp(-w_{j_1,j_2}\mu_{j_1,j_2}(0)$ and $\exp(z_1(\theta_1,(j_1,j_2))-w_{j_1,j_2}\mu_{j_1,j_2}(1))$ respectively. The potential values for setting edges to $0,1$ and $1,0$ are set to $1=\exp(0)$. Solving the k-wise Maximization Problem is equivalent to finding a *maximum a posteriori* assignment in the Markov network we have defined.

Applying this connection and using the ideas of Taskar et al. [2004], we see that our maximization problem can be solved by the following integer program. The integer program has a binary variable corresponding to each of the indicator variables in the above maximization.

$$\begin{aligned}
\max\quad & \sum_{j\in R} w_j(0)\mu_j(0)I_{j,0}+\sum_{j\in R}(z_1(\theta_1,j)+w_j(1)\mu_j(1))I_{j,1}+\\
& \sum_{1\leq j_1<j_2\leq r}(-w_{j_1,j_2}(0)\mu_{j_1,j_2}(0))I_{j_1,j_2,0}+\\
& \sum_{1\leq j_1<j_2\leq r}(z_1(\theta_1,(j_1,j_2))-w_{j_1,j_2}(1))I_{j_1,j_2,1}\\
\text{s.t.}\quad & I_{j,0}+I_{j,1}=1 \quad \text{for all } j\in R\\
& I_{j_1,j_2,p}\leq I_{j_1,p}, I_{j_1,j_2,p}\leq I_{j_2,p} \quad \text{for all } 1\leq j_1<j_2\leq r, p\in\{0,1\}\\
& I_{j,p}\in\{0,1\} \quad \text{for all } j\in R, p\in\{0,1\}\\
& I_{j_1,j_2,p}\in\{0,1\} \quad \text{for all } 1\leq j_1<j_2\leq r, p\in\{0,1\}.
\end{aligned}$$

The first set of constraints ensures that exactly one of $I_{j,0}$ and $I_{j,1}$ is active. The second set of constraints ensures that $I_{j_1,j_2,p}$ is active if and only if $I_{j_1,p}$ and $I_{j_2,p}$ are active. Note that the if direction follows because the objective coefficients of $I_{j_1,j_2,p}$ are non-negative, so the second set of constraints will be tight if $I_{j_1,p}$ and $I_{j_2,p}$ are both set to 1. Therefore, the value of the objective corresponds to $f_w(\theta,o)$, where o consists of the items j for which $I_{j,1}$ is set to one.

To complete the proof, we apply Theorem 3.1 of Taskar et al. [2004] to show that the LP relaxation of this integer program is integral. □

5.3. The Assignment Problem

In the assignment problem, we are given a set of n agents and a set $\{1,\dots,n\}$ of items, and wish to assign each item to exactly one agent. The outcome space of agent i is thus $\Omega_i = \{1,\dots,n\}$, and its type can be represented by a vector $\theta_i \in \Theta_i = \mathbb{R}^n$. The set of possible type profiles is $\Theta = \mathbb{R}^{n^2}$.

We consider an outcome rule that maximizes *egalitarian welfare* in a lexicographic manner: first, the minimum value of any agent is maximized; if more than one outcome achieves the minimum, the second lowest value is maximized, and so forth. This outcome rule can be computed by solving a sequence of integer programs. As such, our focus in this application is not on studying the framework for the setting of tractable outcome rules, but rather for understanding its performance on an objective that is far from welfare maximizing. We continue to assume agent symmetry, and adopt the view of agent 1.

To complete our specification of the structural SVM framework for this application, we need to again define an attribute map. In this case, we follow the same approach as the definition of attribute map χ_2' for the multi-minded combinatorial auction application. The attribute map, $\chi_4'(\theta_{-1}, j)$, where the second argument indexes the item assigned to agent 1, is constructed as,

$$\chi_4'(\theta_{-1}, j) = (\theta_2[-j], \theta_3[-j], \dots, \theta_n[-j]) \in \mathbb{R}^{(n-1)^2},$$

where $\theta_i[-j]$ denotes the vector obtained by removing the jth entry from valuation type θ_i. The attribute map reflects the effect of assigning item j to agent 1 on the valuations of the other agents, capturing the fact that the item cannot be assigned to any other agent. For this set of experiments, we choose not to apply non-linear kernels on top of this attribute vector in order to evaluate the effect of a simple feature map.

6. EXPERIMENTAL EVALUATION

We perform a series of experiments to test our theoretical framework. To run our experiments, we use the SVM^{struct} package [Joachims et al. 2009], which allows for the use of custom kernel functions, attribute maps, and separation oracles.

6.1. Setup

We begin by briefly discussing our experimental methodology, performance metrics, and optimizations used to speed up the experiments.

6.1.1. Methodology. For each of the settings we consider, we generate three data sets: a training set, a validation set, and a test set. The training set is used as input to Training Problem 2, which in turn yields classifiers h_w and corresponding payment rules p_w. For each choice of the parameter C of Training Problem 2, and the parameter γ if the RBF kernel is used, a classifier h_w is learned based on the training set and evaluated based on the validation set. The classifier with the highest accuracy on the validation set is then chosen and evaluated on the test set. During training, we take the perspective of agent 1, and so a training set size of ℓ means that we train an SVM on ℓ examples. Once a partial outcome rule has been learned, however, it can be used to infer payments for all agents. We exploit this fact during testing, and report performance metrics across all agents for a given instance in the test set.

6.1.2. Metrics. We employ three metrics to measure the performance of the learned classifiers. These metrics are computed over the test set $\{(\theta^k, o^k)\}_{k=1}^{\ell}$.

Classification accuracy. Classification accuracy measures the accuracy of the trained classifier in predicting the outcome. Each instance of the ℓ instances has n agents, so in total

we measure accuracy over $n\ell$ instances:[6]

$$accuracy = 100 \cdot \frac{\sum_{k=1}^{\ell} \sum_{i=1}^{n} \mathbb{I}(h_w(\theta_i^k, \theta_{-i}^k) = o_i^k))}{n\ell}.$$

Ex post regret. We measure ex post regret by summing over the ex post regret experienced by all agents in each of the ℓ instances in the dataset, i.e.,

$$regret = \frac{\sum_{k=1}^{\ell} \sum_{i=1}^{n} rgt_i(\theta_i^k, \theta_{-i}^k)}{n\ell}.$$

Individual rationality violation. This metric measures the fraction of individual rationality violation across all agents:

$$ir\text{-}violation = \frac{\sum_{k=1}^{\ell} \sum_{i=1}^{n} \mathbb{I}(irv_i(\theta_i^k, \theta_{-i}^k) > 0)}{n\ell}.$$

Expected individual rationality violation. This metric measures the expected amount of individual rationality violation across all agents:

$$exp\text{-}ir\text{-}viol = \frac{\sum_{k=1}^{\ell} \sum_{i=1}^{n} \min(u_i(\theta_i^k, \theta_{-i}^k), 0)}{n\ell}$$

We also measure *exp-cond-ir-viol* which only averages over agents with negative utility.

Individual rationality violation percentiles. ir-viol-95 measures the threshold at which 95% of agents have utility at least the negative of this value. So if this metric is 0.2, this means that 95% of agents have utility at least -0.2. Similarly, we measure *cond-ir-viol-95*, which provides the same threshold but limited to users with negative utility.

6.1.3. Optimizations. In the case of multi-minded CAs, we first map the inputs θ_{-1} into a smaller space, which allows us to learn more effectively with smaller amounts of data.[7] For this step, we use *instance-based normalization*, which normalizes the values in θ_{-1} by the highest observed value and then rescales the computed payment appropriately, and *sorting*, which orders agents based on bid values.

Before passing examples θ to the learning algorithm or learned classifier, they are normalized by a positive multiplier so that the value of the highest bid by agents other than agent 1 is exactly 1, before passing it to the learning algorithm or classifier. The values and the solution are then transformed back to the original scale before computing the payment rule p_w. This technique of *instance-based normalization* leverages the observation that agent 1's allocation depends on the relative values of the other agent's reports, so that scaling all reports by a factor does not affect the outcome chosen. We apply this to multi-minded CAs and the assignment problem, but not to our experiments on CAs with positive k-wise dependent valuations.

In the sorting step, instead of choosing an arbitrary ordering of agents in θ_{-i}, we choose a specific ordering based on the maximum value the agent reports. For example, in a single-item setting, this amounts to ordering agents by their value. In the multi-minded CA setting, agents are ordered by the value they report for their most desired bundle. The intuition behind sorting is that we can again decrease the space of possible θ_{-i} reports the learner sees and learn more quickly. In the single-item case, we know that the second price payment

[6]For a given instance θ, there are actually many ways to choose (θ_i, θ_{-i}) depending on the ordering of all agents but agent i. We discuss a technique we refer to as sorting in Section 6.1.3, which will choose a particular ordering. When this technique is not used, for example in application to the assignment problem, we fix an ordering of the other agents for each agent i, and use the same ordering across all instances.

[7]The barrier to using more data is not the availability of the data itself, but the time required for training, because training time scales quadratically in the size of the training set due to the use of non-linear kernels.

rule only depends on the maximum value across all other agents, and sorting places this value in the first coordinate of θ_{-i}. We apply sorting to the assignment problem by ordering agents by their maximum value for any item. We do not apply sorting to our experiments with k-wise dependent valuations in CAs.[8]

6.2. Single-Item Auction

As a sanity check, we first perform experiments in application to a single-item auction with the efficient outcome rule, where the agent with the highest bid receives the item. For the distribution D on value profiles, we simply draw each agent's value independently from a uniform distribution on $[0, 1]$. The outcome rule g allocates the item to the agent with the highest value. We use a training set size of 300, and validation and test set sizes of 1000. We use an RBF kernel and parameters $C \in \{10^4, 10^5\}$ and $\gamma \in \{0.01, 0.1, 1\}$.

In this case, we know that the associated payment function that makes (g, p) strategyproof is the second price payment rule.

The results reported in Table I and Figure 3 are for attribute maps χ_1' and χ_2', which can be applied to this setting by observing that single-item auctions are a special case of multi-minded CAs. In particular, letting $\underline{0}$ be the 0 vector of dimension $n-1$, $\chi_1'(\theta_{-1}, o_1) = (\theta_{-1}, \underline{0})$ if $o_1 = \emptyset$ and $\chi_1'(\theta_{-1}, o_1) = (\underline{0}, \theta_{-1})$ if $o_1 = \{1\}$ and $\chi_2'(\theta_{-1}, o_1) = \theta_{-1}$ if $o_1 = \emptyset$ and $\chi_2'(\theta_{-1}, o_1) = \underline{0}$ if $o_1 = \{1\}$. For both choices of the attribute map we obtain excellent accuracy and very close approximation to the second-price payment rule. This shows that the framework is able to automatically learn the payment rule of Vickrey's auction.[9]

Table I. Performance metrics for single-item auction.

n	accuracy		regret		ir-violation	
	χ_1	χ_2	χ_1	χ_2	χ_1	χ_2
2	99.7	93.1	0.000	0.003	0.00	0.07
3	98.7	97.6	0.000	0.000	0.01	0.00
4	98.4	99.1	0.000	0.000	0.00	0.01
5	97.3	96.6	0.001	0.001	0.02	0.00
6	97.6	97.4	0.000	0.001	0.00	0.02

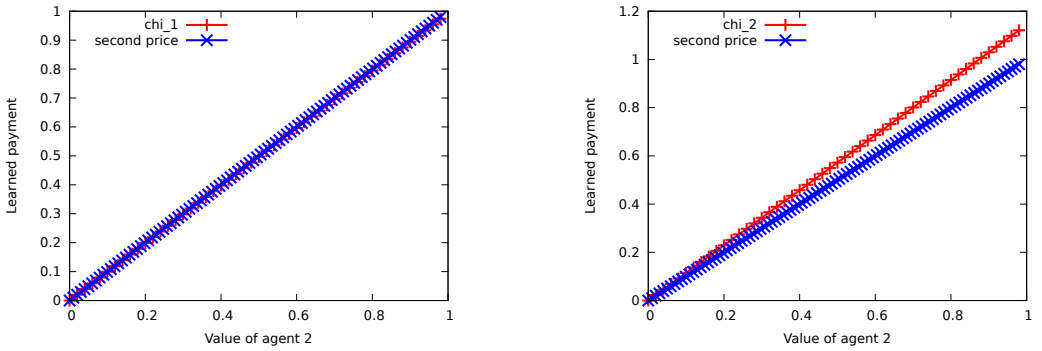

Fig. 3. Learned payment rule vs. second-price payment rule for single-item auction with 2 agents, for χ_1' (left) and χ_2' (right).

[8]Sorting may be effective in this domain as well, but we did not try this out in our experiments.

[9]Given that we apply sorting, the good performance is not surprising since the payment rule is a linear function of the sorted valuations. While we do not report on it here, we observe similar performance even if we do not apply sorting.

6.3. Multi-Minded CAs

6.3.1. Type Distribution. Recall that in a multi-minded setting, there are r items, and each agent is interested in exactly $\kappa > 1$ bundles. For each bundle, we use the following procedure to determine which items are included in the bundle. We first assign an item to the bundle uniformly at random. Then with probability α, we add another random item (chosen uniformly from the remaining items), and with probability $(1 - \alpha)$ we stop. We continue this procedure until we stop or have exhausted the items. This procedure is inspired by Sandholm's decay distribution for the single-minded setting [Sandholm 2002], and we use $\alpha = 0.75$ to be consistent with that setting, where this parameter value generated harder instances of the winner determination problem.

Once the bundle identities have been determined, we sample values for these bundles. Let c be an r-dimensional vector with entries chosen uniformly from $(0, 1]$. For each agent i, let d_i be an r-dimensional vector with entries chosen uniformly from $(0, 1]$. Each entry of c denotes the common value of a specific item, while each entry of d_i denotes the private value of a specific item for agent i. The value of bundle S_{ij} is then given by

$$v_{ij} = \min_{S_{ij'} \leq S_{ij}} \left(\frac{\langle S_{ij'}, \beta c + (1 - \beta) d_i \rangle}{r} \right)^{\zeta}$$

for parameters $\beta \in [0, 1]$ and $\zeta > 1$. The inner product in the numerator corresponds to a sum over values of items, where common and private values for each item are respectively weighted with β and $(1 - \beta)$. The denominator normalizes all valuations to the interval $(0, 1]$. Parameter ζ controls the degree of complementarity among items: $\zeta > 1$ implies that goods are complements, whereas $\zeta < 1$ means that goods are substitutes. Choosing the minimum over bundles $S_{ij'}$ contained in S_{ij} finally ensures that the resulting valuations are monotonic.

6.3.2. Outcome Rules. We use two outcome rules in our experiments on multi-minded CAs. For the *optimal* outcome rule, the payment rule p_{vcg} makes the mechanism (g_{opt}, p_{vcg}) strategyproof. Under this payment rule, agent i pays the externality it imposes on other agents. That is,

$$p_{vcg,1}(\theta) = \left(\max_{o \in \Omega} \sum_{i \neq 1} v_i(\theta_i, o_i) \right) - \sum_{i \neq 1} v_i(\theta_i, g_i(\theta)).$$

The second outcome rule with which we experiment is a generalization of the greedy outcome rule for single-minded CA [Lehmann et al. 2002]. Our generalization of the greedy rule is as follows. Let θ be the agent valuations and $o_i(j)$ denote the jth bundle desired by agent i. For each bundle $o_i(j)$, assign a score $v_i(\theta_i, o_i(j))/\sqrt{|o_i(j)|}$, where $|o_i(j)|$ indicates the total items in bundle $o_i(j)$. The greedy outcome rule orders the desired bundles by this score, and takes the bundle $o_i(j)$ with the next highest score as long as agent i has not already been allocated a bundle and $o_i(j)$ does not contain any items already allocated. While this greedy outcome rule has an associated payment rule that makes it strategyproof in the single-minded case, it is not implementable in the multi-minded case, as evidenced by the example in Appendix B.

6.3.3. Description of Experiments. We experiment with training sets of sizes 100, 300, and 500, and validation and test sets of size 1000. All experiments we report on are for a setting with 5 agents, 5 items, and 3 bundles per agent, and use $\beta = 0.5$, the RBF kernel, and parameters $C \in \{10^4, 10^5\}$ and $\gamma \in \{0.01, 0.1, 1\}$.

6.3.4. Basic Results. Table II presents the basic results for multi-minded CAs with optimal and greedy outcome rules, respectively. For both outcome rules, we present the results for

Table II. Results for multi-minded CA with training set size 500.

		Optimal outcome rule									Greedy outcome rule								
		accuracy			regret			ir-violation			accuracy			regret			ir-violation		
n	ζ	p_{vcg}	χ_1	χ_2	p_{vcg}	χ_1	χ_2	p_{vcg}	χ_1	χ_2	p_{vcg}	χ_1	χ_2	p_{vcg}	χ_1	χ_2	p_{vcg}	χ_1	χ_2
2	0.5	100	70.7	91.9	0	0.014	0.002	0.0	0.06	0.03	50.9	59.1	40.6	0.079	0.030	0.172	0.22	0.12	0.33
3	0.5	100	54.5	75.4	0	0.037	0.017	0.0	0.19	0.10	55.4	57.9	54.7	0.070	0.030	0.088	0.18	0.21	0.36
4	0.5	100	53.8	67.7	0	0.042	0.031	0.0	0.22	0.18	61.1	58.2	57.9	0.056	0.033	0.056	0.14	0.20	0.31
5	0.5	100	15.8	67.0	0	0.133	0.032	0.0	0.26	0.19	64.9	61.3	63.0	0.048	0.027	0.042	0.13	0.19	0.24
6	0.5	100	61.1	68.2	0	0.037	0.032	0.0	0.22	0.20	66.6	63.8	63.8	0.041	0.034	0.045	0.12	0.20	0.24
2	1.0	100	84.5	93.4	0	0.008	0.001	0.0	0.08	0.02	87.8	86.6	84.0	0.007	0.005	0.008	0.04	0.06	0.09
3	1.0	100	77.1	83.5	0	0.012	0.005	0.0	0.13	0.09	85.3	86.7	85.7	0.006	0.006	0.006	0.04	0.07	0.05
4	1.0	100	74.6	81.1	0	0.014	0.009	0.0	0.16	0.12	82.4	86.5	84.2	0.006	0.006	0.007	0.05	0.08	0.08
5	1.0	100	73.4	77.4	0	0.018	0.011	0.0	0.19	0.12	82.7	85.8	84.9	0.007	0.009	0.009	0.04	0.10	0.10
6	1.0	100	75.0	77.7	0	0.020	0.013	0.0	0.20	0.16	80.0	87.4	88.1	0.006	0.007	0.005	0.04	0.08	0.07
2	1.5	100	91.5	96.9	0	0.004	0.000	0.0	0.06	0.02	94.7	91.1	91.7	0.002	0.002	0.002	0.02	0.04	0.04
3	1.5	100	91.0	93.4	0	0.004	0.001	0.0	0.05	0.03	97.1	92.8	93.2	0.001	0.002	0.001	0.01	0.02	0.04
4	1.5	100	92.5	94.2	0	0.003	0.001	0.0	0.03	0.04	96.4	91.5	92.1	0.001	0.003	0.002	0.02	0.07	0.07
5	1.5	100	91.7	93.9	0	0.004	0.002	0.0	0.06	0.03	97.5	90.5	91.4	0.001	0.004	0.002	0.01	0.06	0.04
6	1.5	100	91.9	93.7	0	0.003	0.001	0.0	0.05	0.04	98.4	92.2	92.8	0.000	0.003	0.002	0.01	0.06	0.06

p_{vcg} as a baseline. Because p_{vcg} is the strategyproof payment rule for the optimal outcome rule, p_{vcg} always has accuracy 100, regret 0, and IR violation 0 for the optimal outcome rule. The main findings are that our learned payment rule has low regret for the optimal outcome rule and regret that is about the same as or better than the regret of p_{vcg} when the outcome rule is greedy. Given that greedy winner determination is seeking to maximize total welfare it is natural the VCG-based payments would perform reasonably well in this environment.

Across all instances, as expected, accuracy is negatively correlated with regret and ex post IR violation. The degree of complementarity between items, ζ, as well as the outcome rule chosen, has a major effect on the results. Instances with low complementarity ($\zeta = 0.5$) yield payment rules with higher regret, and χ_1' performs better on the greedy outcome rule while χ_2' performs better on the optimal outcome rule. For high complementarity between items the greedy outcome tends to allocate all items to a single agent, and the learned price function sets high prices for small bundles to capture this property. For low complementarity the allocation tends to be split and less predictable. Still, the best classifiers achieve average ex post regret of less than 0.032 (for values normalized to [0,1]) even though the corresponding prediction accuracy can be as low as 67%.

For the greedy outcome rule, the performance of p_{vcg} is comparable for $\zeta \in \{1.0, 1.5\}$ but worse than the payment rule learned in our framework in the case of $\zeta = 0.5$, where the greedy outcome rule becomes less optimal.

6.3.5. Effect of Training Set Size. Table III charts performance as the training set size is varied for the greedy outcome rule. While training data is readily available (we can simply sample from D and run the outcome rule g), training time becomes prohibitive[10] for larger training set sizes. Table III shows that regret decreases with larger training sets, and for a training set size of 500, the best of χ_1' and χ_2' outperforms p_{vcg} for $\zeta = 0.5$ and is comparable to p_{vcg} for $\zeta \in \{1.0, 1.5\}$.

6.3.6. IR Fixes. Tables IV and V summarize our results regarding the various fixes to IR violations, for the particularly challenging case of the greedy outcome rule and $\zeta = 0.5$.

[10]Training took on the order to a few days for the largest problem sizes and training set sizes we considered.

Table III. Effect of training set size on accuracy of learned classifier. Multi-minded CA, greedy outcome rule. Training set size is given in the column labels for χ'_1, χ'_2. p_{vcg} does not have a training set size.

n	ζ	accuracy	100		300		500		regret	100		300		500	
		p_{vcg}	χ'_1	χ'_2	χ'_1	χ'_2	χ'_1	χ'_2	p_{vcg}	χ'_1	χ'_2	χ'_1	χ'_2	χ'_1	χ'_2
2	0.5	50.9	54.3	48.2	57.0	46.9	59.1	40.6	0.079	0.045	0.195	0.032	0.098	0.030	0.172
3	0.5	55.4	50.1	49.8	55.7	54.4	57.9	54.7	0.070	0.054	0.078	0.038	0.082	0.030	0.088
4	0.5	61.1	53.4	56.2	56.4	58.5	58.2	57.9	0.056	0.050	0.059	0.040	0.061	0.033	0.056
5	0.5	64.9	14.2	57.9	61.0	61.8	61.3	63.0	0.048	0.173	0.064	0.038	0.048	0.027	0.042
6	0.5	66.6	58.4	58.8	62.2	63.9	63.8	63.8	0.041	0.039	0.059	0.037	0.049	0.034	0.045
2	1.0	87.8	80.7	80.5	84.4	84.1	86.6	84.0	0.007	0.010	0.010	0.009	0.008	0.005	0.008
3	1.0	85.3	74.9	78.0	83.0	80.6	86.7	85.7	0.006	0.020	0.011	0.009	0.009	0.006	0.006
4	1.0	82.4	78.5	80.1	84.2	83.1	86.5	84.2	0.006	0.015	0.014	0.008	0.009	0.006	0.007
5	1.0	82.7	81.0	81.8	84.3	84.3	85.8	84.9	0.007	0.020	0.014	0.010	0.009	0.009	0.009
6	1.0	80.0	81.8	83.7	87.6	88.3	87.4	88.1	0.006	0.062	0.018	0.008	0.005	0.007	0.005
2	1.5	94.7	83.3	88.1	89.3	89.8	91.1	91.7	0.002	0.008	0.003	0.003	0.002	0.002	0.002
3	1.5	97.1	86.9	87.6	90.3	91.5	92.8	93.2	0.001	0.005	0.004	0.003	0.002	0.002	0.001
4	1.5	96.4	88.4	90.7	89.3	90.8	91.5	92.1	0.001	0.005	0.003	0.004	0.003	0.003	0.002
5	1.5	97.5	87.2	88.5	91.4	90.5	90.5	91.4	0.001	0.006	0.004	0.003	0.003	0.004	0.002
6	1.5	98.4	86.3	86.8	91.4	92.5	92.2	92.8	0.000	0.011	0.007	0.004	0.002	0.003	0.002

Table IV. Impact of payment offset and null loss fix for $\zeta = 0.5$ and greedy outcome rule, training set size 300. All results are for χ'_2, null loss values across columns.

payment offset	accuracy			regret			ir-violation			ir-fix-welfare-avg		
	0.5	1.0	1.5	0.5	1.0	1.5	0.5	1.0	1.5	0.5	1.0	1.5
0	0.639	0.639	0.628	0.058	0.049	0.043	0.279	0.248	0.191	0.345	0.407	0.529
0.05	0.639	0.635	0.606	0.051	0.046	0.046	0.226	0.192	0.142	0.446	0.522	0.635
0.1	0.627	0.613	0.575	0.050	0.049	0.053	0.181	0.144	0.097	0.541	0.622	0.733
0.15	0.613	0.589	0.536	0.053	0.056	0.065	0.135	0.103	0.065	0.636	0.713	0.813
0.2	0.591	0.546	0.489	0.060	0.066	0.080	0.096	0.069	0.042	0.726	0.795	0.873
0.25	0.553	0.506	0.449	0.070	0.080	0.097	0.068	0.047	0.027	0.792	0.856	0.914

Table V. Impact of payment offset and null loss fix for $\zeta = 0.5$ and greedy outcome rule, training set size 300. All results are for χ'_2, null loss values across columns.

payment offset	exp-ir-viol			exp-cond-ir-viol			ir-viol-95			cond-ir-viol-95		
	0.5	1.0	1.5	0.5	1.0	1.5	0.5	1.0	1.5	0.5	1.0	1.5
0	0.049	0.037	0.025	0.177	0.151	0.131	0.293	0.242	0.180	0.463	0.390	0.351
0.05	0.036	0.026	0.017	0.161	0.138	0.117	0.243	0.192	0.130	0.435	0.363	0.323
0.1	0.026	0.018	0.011	0.145	0.125	0.109	0.193	0.142	0.080	0.398	0.336	0.301
0.15	0.018	0.012	0.007	0.136	0.116	0.102	0.143	0.092	0.030	0.385	0.328	0.292
0.2	0.013	0.008	0.004	0.132	0.113	0.096	0.093	0.042	0.000	0.365	0.326	0.272
0.25	0.009	0.005	0.002	0.127	0.104	0.086	0.043	0.000	0.000	0.358	0.315	0.234

The extent of IR violation decreases with larger payment offset and null loss. Regret tends to move in the opposite direction, but there are cases where IR violation and regret both decrease. The three rightmost columns of Table IV list the average ratio between welfare after and before the deallocation fix, across the instances in the test set. With a payment offset of 0, a large welfare hit is incurred if we deallocate agents with IR violations. However, this penalty decreases with increasing payment offsets and increasing null loss. At the most extreme payment offset and null loss adjustment, the IR violation is as low as 2%, and the deallocation fix incurs a welfare loss of only 7%. Table V provides detail on the amount by which IR is violated. The expected amount of IR violation is low, though this becomes

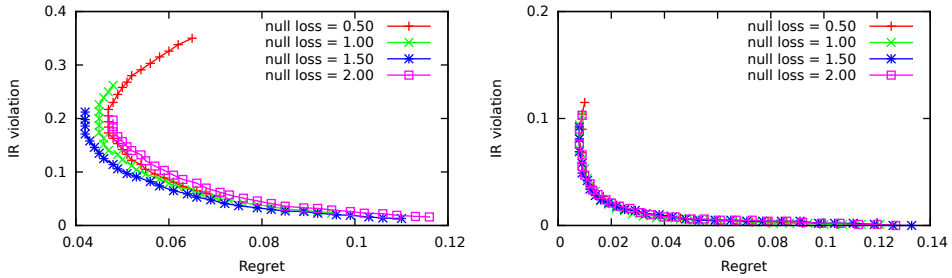

Fig. 4. Impact of payment offset and null loss fix for greedy outcome rule, training set size 300.

Table VI. Comparison of performance with and without optimistically assuming item monotonicity. (i-mon) indicates a payment rule learned by optimistically assuming item monotonicity. Greedy outcome rule. Training set size 300.

n	ζ	accuracy		regret		ir-violation	
		$\chi 2$	$\chi 2$ (i-mon)	$\chi 2$	$\chi 2$ (i-mon)	$\chi 2$	$\chi 2$ (i-mon)
2	0.5	46.9	46.3	0.098	0.232	0.28	0.38
3	0.5	54.4	8.6	0.082	0.465	0.33	0.06
4	0.5	58.5	48.2	0.061	0.811	0.31	0.25
5	0.5	61.8	57.0	0.048	0.136	0.26	0.26
6	0.5	63.9	61.3	0.049	0.078	0.25	0.20
2	1.0	84.1	82.2	0.008	0.010	0.06	0.08
3	1.0	80.6	80.1	0.009	0.010	0.10	0.09
4	1.0	83.1	79.7	0.009	0.012	0.11	0.11
5	1.0	84.3	77.2	0.009	0.020	0.10	0.11
6	1.0	88.3	83.9	0.005	0.013	0.08	0.11
2	1.5	89.8	89.1	0.002	0.003	0.03	0.06
3	1.5	91.5	91.3	0.002	0.003	0.04	0.04
4	1.5	90.8	89.7	0.003	0.003	0.06	0.06
5	1.5	90.5	87.3	0.003	0.005	0.04	0.05
6	1.5	92.5	70.8	0.002	0.081	0.06	0.17

more significant when we condition on agents with IR violations. Larger payment offsets decrease all of these metrics as expected.

Figure 4 shows a graphical representation of the impact of payment offsets and null losses. Each line in the plot corresponds to a payment rule learned with a different null loss, and each point on a line corresponds to a different payment offset. The payment offset is zero for the top-most point on each line, and equal to 0.29 for the lowest point on each line. Increasing the payment offset always decreases the rate of IR violation, but may decrease or increase regret. Increasing null loss lowers the top-most point on a given line, but arbitrarily increasing null loss can be harmful. Indeed, in the figure on the left, a null loss of 1.5 results in a slightly higher top-most point but significantly lower regret at this top-most point compared to a null loss of 2.0. It is also interesting to note that these adjustments have much more impact on the hardest distribution with $\zeta = 0.5$.

6.3.7. Item Monotonicity. Table VI presents a comparison of a payment rule learned with explicit enumeration of all bundle constraints (the default that we have been using for our other results) and a payment rule learned by optimistically assuming item monotonicity (see Section 5.1.3). Performance is affected when we drop constraints and optimistically assume item monotonicity, although the effects are small for $\zeta \in \{1.0, 1.5\}$ and larger for $\zeta = 0.5$.

Table VII. Basic results for valuations with $\rho = 0.1$. Metrics for 10 and 20 items are computed using an approximation based on the tractable separation oracle for our training problem. Metrics are not computed for VCG-based rules because computation requires brute force enumeration over all possible bundles (but can be efficiently computed for the succinctly represented, trained payment rule).

agents	items	accuracy		regret	
		χ	p_{vcg}	χ	p_{vcg}
6	2	94.9	97.3	0.006	0.008
6	4	70.7	82.7	0.020	0.022
6	6	53.1	66.4	0.027	0.031
6	10	28.3	–	0.033	–
6	20	–	–	–	–

Because item monotonicity allows for the training problem to be succinctly specified, we may be able to train on more data, and this seems a very promising avenue for further consideration (perhaps coupled with heuristic methods to add additional constraints to the training problem).

6.4. Combinatorial Auctions with Positive k-wise Dependent Valuations

We experiment with our framework on combinatorial auctions with positive k-wise dependent valuations. We find that our learned payment rules can outperform VCG-based payment rules in terms of regret for settings with large numbers of items, and outperform VCG-based payment rules in terms of the trade-off between IR violation and regret. Because we have an efficient separation oracle as discussed in Section 5.2.4, we are able to train payment rules and compute regret for larger instances.

In order to experiment with positive k-wise dependent valuations in combinatorial auctions, we need a way to generate such valuations. To construct agent i's valuation, we first specify the nodes and edges in a graph (R, E), and then assign weights $z_i(j)$ and $z_i(e)$ over the nodes $j \in R$ and edges $e \in E$. For every possible edge (j_1, j_2), we add the edge to the agents' graph with probability ρ. Value $z_i(j)$ is sampled uniformly at random from $[0, 1]$; the weight for each added edge is also sampled uniformly at random from $[0, 1]$. With this setup, the edge probability parameter ρ lets us generate test instances of varying edge density. So that our regret numbers are comparable across different size instances, we normalize each agent's weights by the expected value for the set of all items.

The outcome rule we use is the greedy outcome rule outlined in Section 5.2.1. We use a training set size of 1000 and validation and test sets of size 500. We compare against a VCG-based payment rule which runs the greedy allocation rule on all agents and on all agents excluding agent i and charges agent i the difference in value to agents other that i in the two allocations.

Tables VII and VIII and Figure 5 compare our learned payment rules (with 0 null loss) to the VCG-based payment rule for $\rho = 0.1$ and $\rho = 0.9$. The learned payment rule has better regret, despite having worse accuracy. However, the learned payment rule incurs significant IR violation. We examine the IR violation issue in Figure 6. Here we implement the two IR fixes of increasing the null loss value and applying payment offsets. We see that across all instances, we can find settings of the null loss for which our IR / regret curve lies beneath that of the VCG-based payment rule, indicating that we have settings which have better regret and lower IR violation. We also see that despite having significant IR failures when we have no payment offset, we can significantly decrease IR violation at the cost of a small amount of regret increase by using a payment offset.

Table VIII. Basic results for valuations with $\rho = 0.9$. Metrics for 10 and 20 items are computed using an approximation based on the tractable separation oracle for our training problem. Metrics are not computed for VCG-based rules because computation requires brute force enumeration over all possible bundles (but can be efficiently computed for the succinctly represented, trained payment rule).

agents	items	accuracy		regret	
		χ	p_{vcg}	χ	p_{vcg}
6	2	79.9	82.1	0.024	0.028
6	4	64.0	51.8	0.038	0.056
6	6	61.6	42.7	0.030	0.062
6	10	61.6	–	0.026	–
6	20	–	–	–	–

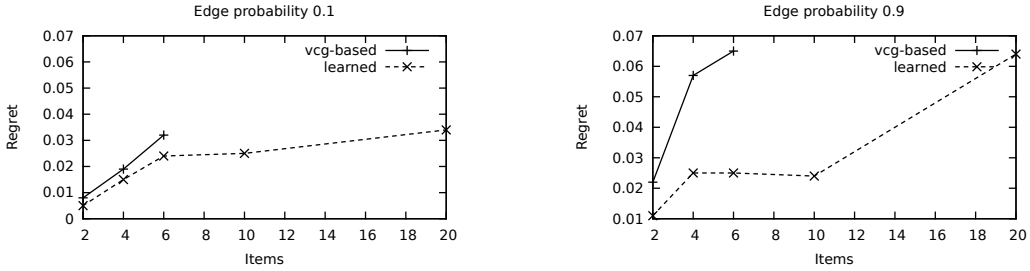

Fig. 5. Regret v. Number of Items for learned payment rule and VCG-based payment rules. For 10 and 20 items, we do not have regret number for the VCG-based rules because computing regret requires enumeration over all possible bundles. In this case, the regret for learned payment rules and 10 and 20 items is an upper bound on the true regret obtained by applying our tractable separation oracle.

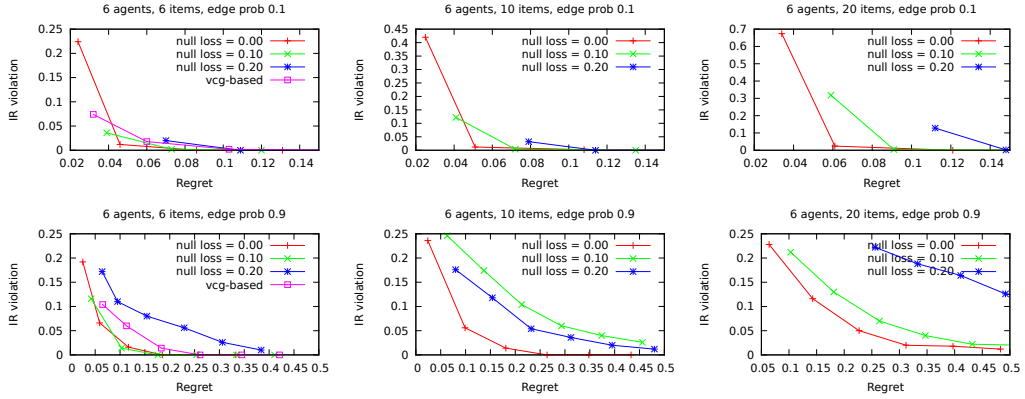

Fig. 6. Regret v. IR Violation trade-off for learned payment rule and VCG-based payment rule for k-wise dependent valuations. We do not have regret numbers for the VCG-based rule and 10 and 20 items because computing regret requires brute force enumeration over all possible bundles. In this case, the regret numbers for the learned payment rule are an upper bound on regret obtained by using our tractable separation oracle.

6.5. The Egalitarian Assignment Problem

In the assignment problem, agents' values for the items are sampled uniformly and independently from $[0, 1]$. We use a training set of size 600, validation and test sets of size 1000, and

Table IX. Results for assignment problem with egalitarian outcome rule

n	accuracy				regret				ir-violation			
	vcg	tot-vcg	eg-vcg	p_w	vcg	tot-vcg	eg-vcg	p_w	vcg	tot-vcg	eg-vcg	p_w
2	64.3	67.5	67.5	89.0	0.018	0.015	0.015	0.023	0.03	0.01	0.01	0.03
3	48.0	52.1	42.5	77.9	0.070	0.077	0.127	0.041	0.06	0.07	0.03	0.04
4	40.6	43.1	30.8	71.0	0.111	0.123	0.199	0.054	0.07	0.09	0.03	0.02
5	32.4	35.3	24.5	63.9	0.157	0.169	0.254	0.071	0.10	0.12	0.03	0.01
6	27.1	29.9	20.0	59.0	0.189	0.208	0.290	0.074	0.10	0.13	0.03	0.01

the RBF kernel with parameters $C \in \{10, 1000, 100000\}$ and $\gamma \in \{0.1, 0.5, 1.0\}$. We find that our learned payment rules have significantly better accuracy and regret than VCG-based payment rules. We explain the improvement over VCG-based payments by observing that the egalitarian rule is not maximizing total welfare, and thus not compatible in this sense with VCG-based ideas.

The performance of the learned payment rules is compared to that of three VCG-based payment rules. For this, let W be the total welfare of all agents other than i under the outcome chosen by g, and W_{eg} be the minimum value any agent other than i receives under this outcome. We consider the following payment rules:

(1) the *vcg* payment rule, where agent i pays the difference between the maximum total welfare of the other agents under any allocation and W;

(2) the *tot-vcg* payment rule, where agent i pays the difference between the total welfare of the other agents under the allocation maximizing egalitarian welfare and W; and

(3) the *eg-vcg* payment rule, where agent i pays the difference between the minimum value of any agent under the allocation maximizing egalitarian welfare and W_{eg}.

The results are shown in Table IX. We see that the learned payment rule p_w yields significantly lower regret than any of the VCG-based payment rules, and average ex post regret less than 0.074 for values normalized to $[0, 1]$. Since we are not maximizing the sum of values of the agents, it is not very surprising that VCG-based payment rules perform rather poorly. The learned payment rule p_w can adjust to the outcome rule, and also achieves a low fraction of ex post IR violation of at most 3%.

7. CONCLUSIONS

We have introduced a new paradigm for computational mechanism design, in which statistical machine learning is adopted to design payment rules for outcome rules, and have shown encouraging experimental results. The mechanism design domain can be multi-parameter, and the outcome rules can be specified algorithmically and need not be designed for objectives that are separable across agents. Central to our approach is to relax incentive compatibility as a hard constraint on mechanism design, adopting in its place the goal of minimizing expected regret while requiring agent-independent prices.

Future directions of interest include: (1) considering alternative learning paradigms, including formulations of the problem as a regression rather than classification problem; (2) developing formulations that can impose constraints on properties of the learned payment rule, concerning for example the core, budgets, or individual-rationality properties; (3) developing methods that learn classifiers that induce feasible outcome rules, so that these learned outcome rules can be used directly; (4) extending the approach to domains without money by developing a structure on discriminant functions appropriate to the incentive considerations facing rational self-interested agents in such domains; (5) investigating the extent to which alternative goals can be achieved through machine learning, such as *regret percentiles* (maximizing the probability that the ex post regret is no greater than some amount $\lambda > 0$), or directly minimizing the expected *interim regret*; (6) explore alternate attribute maps (e.g., it would be interesting to adopt attributes that encode economic concepts such as the

total externality imposed on others by by assigning a bundle of items to agent 1), kernels, and succinct valuation representations.

Acknowledgments

We thank Shivani Agarwal, Ruggiero Cavallo, Vince Conitzer, Amy Greenwald, Jason Hartline, Sébastien Lahaie, and Tim Roughgarden for valuable discussions, and the anonymous referees for helpful feedback. All errors remain our own. This material is based upon work supported in part by the National Science Foundation under grant CCF-1101570, the Deutsche Forschungsgemeinschaft under grant FI 1664/1-1, an EURYI award, an NDSEG fellowship, and an SNF Postdoctoral Fellowship.

REFERENCES

Abraham, I., Babaioff, M., Dughmi, S., and Roughgarden, T. 2012. Combinatorial auctions with restricted complements. In *Proceedings of the 13th ACM Conference on Electronic Commerce*. 3–16.

Ashlagi, I., Braverman, M., Hassidim, A., and Monderer, D. 2010. Monotonicity and implementability. *Econometrica 78*, 5, 1749–1772.

Ausubel, L. M. and Milgrom, P. 2006. The lovely but lonely Vickrey auction. In *Combinatorial Auctions*, P. Cramton, Y. Shoham, and P. Steinberg, Eds. MIT Press, Chapter 1, 17–40.

Bei, X. and Huang, Z. 2011. Bayesian incentive compatibility via fractional assignments. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*. 720–733.

Budish, E. 2011. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy 119*, 1061–1103.

Cai, Y., Daskalakis, C., and Weinberg, S. M. 2012. An algorithmic characterization of multi-dimensional mechanisms. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*. 459–478.

Carroll, G. 2011. A quantitative approach to incentives: Application to voting rules. Tech. rep., MIT.

Conitzer, V. and Sandholm, T. 2002. Complexity of mechanism design. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence*. 103–110.

Conitzer, V., Sandholm, T., and Santi, P. 2005. Combinatorial auctions with k-wise dependent valuations. In *Proceedings of the 20th AAAI Conference on Artificial Intelligence*. 248–254.

Day, R. and Milgrom, P. 2008. Core-selecting package auctions. *International Journal of Game Theory 36,* 3–4, 393–407.

Erdil, A. and Klemperer, P. 2010. A new payment rule for core-selecting package auctions. *Journal of the European Economic Association 8,* 2–3, 537–547.

Guo, M. and Conitzer, V. 2010. Computationally feasible automated mechanism design: General approach and case studies. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*.

Hartline, J. D., Kleinberg, R., and Malekian, A. 2011. Bayesian incentive compatibility via matchings. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*. 734–747.

Joachims, T., Finley, T., and Yu, C.-N. J. 2009. Cutting-plane training of structural SVMs. *Machine Learning 77,* 1, 27–59.

Lahaie, S. 2009. A kernel method for market clearing. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*. 208–213.

Lahaie, S. 2010. Stability and incentive compatibility in a kernel-based combinatorial auction. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. 811–816.

Lavi, R., Mu'alem, A., and Nisan, N. 2003. Towards a characterizatiotowards a characterization of truthful combinatorial auctions. In *Proc. 44th Annual IEEE Symp. on Foundations of Computer Science (FOCS)*. 574–583.

Lavi, R. and Swamy, C. 2005. Truthful and near-optimal mechanism design via linear programming. In *Proceedings of the 46th Symposium on Foundations of Computer Science*. 595–604.

Lehmann, D., O'Callaghan, L. I., and Shoham, Y. 2002. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM 49*, 577–602.

Lubin, B. 2010. Combinatorial markets in theory and practice: Mitigating incentives and facilitating elicitation. Ph.D. thesis, Department of Computer Science, Harvard University.

Lubin, B. and Parkes, D. C. 2009. Quantifying the strategyproofness of mechanisms via metrics on payoff distributions. In *Proceedings of the 25th Annual Conference on Uncertainty in Artificial Intelligence*. 349–358.

Lubin, B. and Parkes, D. C. 2012. Approximate strategyproofness. *Current Science 103,* 9, 1021–1032.

Myerson, R. B. 1981. Optimal auction design. *Mathematics of operations research 6,* 1, 58–73.

Parkes, D. C., Kalagnanam, J., and Eso, M. 2001. Achieving budget-balance with Vickrey-based payment schemes in exchanges. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*. 1161–1168.

Pathak, P. and Sönmez, T. 2013. School admissions reform in Chicago and England: Comparing mechanisms by their vulnerability to manipulation. *American Economic Review 103*, 80–106.

Rastegari, B., Condon, A., and Leyton-Brown, K. 2011. Revenue monotonicity in deterministic, dominant-strategy combinatorial auctions. *Artificial Intelligence 175*, 441–456.

Saks, M. and Yu, L. 2005. Weak monotonicity suffices for truthfulness on convex domains. In *Proceedings of the 6th ACM Conference on Electronic Commerce*. 286–293.

Sandholm, T. 2002. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence 135,* 1-2, 1–54.

Taskar, B., Chatalbashev, V., and Koller, D. 2004. Learning associative markov networks. In *Proceedings of the 21st International Conference on Machine Learning*.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research 6*, 1453–1484.

Yokoo, M., Sakurai, Y., and Matsubara, S. 2004. The effect of false-name bids in combinatorial auctions: new fraud in internet auctions. *Games and Economic Behavior 46,* 1, 174–188.

A. EFFICIENT COMPUTATION OF INNER PRODUCTS

For both χ'_1 and χ'_2, computing inner products reduces to the question of whether inner products between valuation profiles are efficiently computable. For χ'_1, we have that

$$\left\langle \chi'_1(\theta_{-1}, o_1), \chi'_1(\theta'_{-1}, o'_1) \right\rangle = \mathbf{I}_{o_1=o'_1} \sum_{i=2}^{n} \langle \theta_i, \theta'_i \rangle \,,$$

where indicator $\mathbf{I}_{o_1=o_1'} = 1$ if $o_1 = o_1'$ and $\mathbf{I}_{o_1=o_1'} = 0$ otherwise. For χ_2',

$$\left\langle \chi_2'(\theta_{-1}, o_1), \chi_2'(\theta'_{-1}, o_1') \right\rangle = \sum_{i=2}^{n} \left\langle \theta_i \setminus o_1, \theta_i' \setminus o_1 \right\rangle .$$

We next develop efficient methods for computing the inner products $\langle \theta_i, \theta_i' \rangle$ on compactly represented valuation functions. The computation of $\langle \theta_i \setminus o_1, \theta_i' \setminus o_1 \rangle$ can be done through similar methods.

In the single-minded setting, let θ_i correspond to a bundle $S_i \subseteq \{1, \ldots, r\}$ of items with value v_i, and θ_i' correspond to a set $S_i' \subseteq \{1, \ldots, r\}$ of items valued at v_i'.

Each set containing both S_i and S_i' contributes $v_i v_i'$ to $\theta_i^T \theta_i'$, while all other sets contribute 0. Since there are exactly $2^{r-|S_i \cup S_i'|}$ sets containing both S_i and S_i', we have

$$\theta_i^T \theta_i' = v_i v_i' 2^{r-|S_i \cup S_i'|}.$$

This is a special case of the formula for the multi-minded case.

LEMMA A.1. *Consider a multi-minded CA and two bid vectors x_1 and x_1' corresponding to sets $S = \{S_1, \ldots, S_s\}$ and $S' = \{S_1', \ldots, S_t'\}$, with associated values $v_1, \ldots, v_s$ and $v_1', \ldots, v_t'$. Then,*

$$x_1^T x_1' = \sum_{T \subseteq S, T' \subseteq S'} \Big((-1)^{|T|+|T'|} \cdot (\min_{S_i \in T} v_i) \cdot (\min_{S_j' \in T'} v_j') \cdot 2^{r-|(\bigcup_{S_i \in T} S_i) \cup (\bigcup_{S_j' \in T'} S_j')|} \Big). \quad (6)$$

PROOF. The contribution of a particular bundle B' of items to the inner product is $(\max_{S_i \in S, S_i \subseteq B'} v_i) \cdot (\max_{S_j' \in S', S_j' \subseteq B'} v_j')$, and thus

$$x_1^T x_1' = \sum_{B'} \Big((\max_{\substack{S_i \in S \\ S_i \subseteq B'}} v_i) \cdot (\max_{\substack{S_j' \in S' \\ S_j' \subseteq B'}} v_j') \Big).$$

By the maximum-minimums identity, which asserts that for any set $\{x_1, \ldots, x_n\}$ of n numbers, $\max\{x_1, \ldots, x_n\} = \sum_{Z \subseteq X} ((-1)^{|Z|+1} \cdot (\min_{x_i \in Z} x_i))$,

$$\max_{\substack{S_i \in S \\ S_i \subseteq B'}} v_i = \sum_{\substack{T \subseteq S \\ \cup_{S_i \in T} S_i \subseteq B'}} \Big((-1)^{|T|+1} \cdot (\min_{S_i \in T} v_i) \Big) \quad \text{and}$$

$$\max_{\substack{S_j' \in S' \\ S_j' \subseteq B'}} v_j' = \sum_{\substack{T' \subseteq S' \\ \cup_{S_j' \in T'} S_j' \subseteq B'}} \Big((-1)^{|T'|+1} \cdot (\min_{S_j' \in T'} v_j') \Big).$$

The inner product can thus be written as

$$\theta_1^T \theta_1' = \sum_{B'} \sum_{\substack{T \subseteq S, T' \subseteq S' \\ \cup_{S_i \in T} S_i \subseteq B' \\ \cup_{S_j' \in T'} S_j' \subseteq B'}} \Big((-1)^{|T|+|T'|} \cdot (\min_{S_i \in T} v_i) \cdot (\min_{S_j' \in T'} v_j') \Big).$$

Finally, for given $T \subseteq S$ and $T' \subseteq S'$, there exist exactly $2^{r-|(\bigcup_{S_i \in T} S_i) \cup (\bigcup_{S_j' \in T'} S_j')|}$ bundles B' such that $\bigcup_{S_i \in T} S_i \subseteq B'$ and $\bigcup_{S_j' \in T'} S_j' \subseteq B'$, and we obtain

$$\theta_1^T \theta_1' = \sum_{T \subseteq S, T' \subseteq S'} \Big((-1)^{|T|+|T'|} \cdot (\min_{S_i \in T} v_i) \cdot (\min_{S_j' \in T'} v_j') \cdot 2^{m-|(\bigcup_{S_i \in T} S_i) \cup (\bigcup_{S_j' \in T'} S_j')|} \Big). \quad \square$$

If S and S' have constant size, then the sum on the right hand side of (6) ranges over a constant number of sets and can be computed efficiently.

B. GREEDY ALLOCATION RULE IS NOT WEAKLY MONOTONE

Consider a setting with a single agent and four items.

If the valuations θ_1 of the agent are

$$v_1(\theta_1, o_1) = \begin{cases} 20 & \text{if } o_1 = \{1,2,3,4\}, \\ 12 & \text{if } 1 \in o_1 \text{ and } j \notin o_1 \text{ for some } j \in \{2,3,4\}, \text{ and} \\ 0 & \text{otherwise,} \end{cases}$$

then the allocation is $\{1\}$.

If the valuations are θ_1' such that

$$v_1(\theta_1', o_1) = \begin{cases} 12 & \text{if } o_1 = \{1,2,3,4\}, \\ 5 & \text{if } 1 \in o_1 \text{ and } j \notin o_1 \text{ for some } j \in \{2,3,4\}, \text{ and} \\ 0 & \text{otherwise,} \end{cases}$$

then the allocation is $\{1,2,3,4\}$.

We have $v_1(\theta_1', \{1,2,3,4\}) - v_1(\theta_1', \{1\}) < v_1(\theta_1, \{1,2,3,4\}) - v_1(\theta_1, \{1\})$ contradicting weak monotonicity.