Learning Revenue-Maximizing Auctions With Differentiable Matching

Michael J. Curry **Uro Lyi** **Tom Goldstein** **John P. Dickerson**
University of Maryland

Abstract

We propose a new architecture to approximately learn incentive compatible, revenue-maximizing auctions from sampled valuations. Our architecture uses the Sinkhorn algorithm to perform a differentiable bipartite matching which allows the network to learn strategyproof revenue-maximizing mechanisms in settings not learnable by the previous RegretNet architecture. In particular, our architecture is able to learn mechanisms in settings without free disposal where each bidder must be allocated exactly some number of items. In experiments, we show our approach successfully recovers multiple known optimal mechanisms and high-revenue, low-regret mechanisms in larger settings where the optimal mechanism is unknown.

1 INTRODUCTION

Auctions have been held for millennia, since at least Classical antiquity (Krishna, 2009), and have played an important complementary role to set-price sales and bargaining to exchange goods. In recent decades, the advent of computation has resulted in a surge of large-scale fielded auctions in a variety of important industries, such as broadcasting (Leyton-Brown, Milgrom, and Segal, 2017), advertising (Edelman, Ostrovsky, and Schwarz, 2007), electricity markets (Cramton, 2017), and many others (Milgrom, 2017; Roth, 2018). These developments have made auctions not only of theoretical interest, but also of great practical importance.

Auction *design* is the problem faced by an auctioneer who, given uncertain knowledge of the demands of auction participants, wishes to set the rules of the auction to ensure, via proper incentive structure, a desirable outcome. In the usual theoretical model of auctions (Parsons, Rodriguez-Aguilar, and Klein, 2011), the bidders have some private valuation of the items, but the distribution from which these valuations are drawn is common knowledge. The auctioneer solicits bids from participants, awards the items up for sale to the winners, and charges some amount of money to each. Bidders may, however, choose to strategically lie about their valuation given their knowledge of the auction rules and anticipated behavior of other participants, resulting in a Bayes-Nash equilibrium which may be very hard for the auction designer to predict.

To avoid this problem, an auctioneer may simply wish to design a strategyproof (or truthful) mechanism in which participants are incentivized to truthfully reveal their valuations. The distribution of bids is then simply the valuation distribution itself, so it becomes easy to predict the results of the auction in expectation. While satisfying the strategyproofness constraint, the auctioneer can additionally optimize total utility, their own revenue, or other desirable properties.

If the auctioneer's goal is to maximize the welfare of all participants while maintaining strategyproofness, the Vickrey-Clarke-Groves (VCG) mechanism always gives a solution (Vickrey, 1961; Clarke, 1971; Groves, 1973). By contrast, if the auctioneer wishes to maximize revenue, strategyproof mechanisms are much harder to find. Some revenue-maximizing strategyproof mechanisms are known in limited cases: when selling a single item, the Myerson auction is strategyproof and maximizes revenue (Myerson, 1981), and for selling multiple items to one agent, some results are known (Daskalakis, Deckelbaum, and Tzamos, 2015; Kash and Frongillo, 2016; Pavlov, 2011; Manelli and Vincent, 2006). But even for the simple case of selling *two* items to *two* agents, there has been almost no progress, with the major exception of a breakthrough for the special case where items take on at most two discrete valuations (Yao, 2017).

The theoretical difficulty of devising revenue-maximizing strategyproof auctions has resulted in a

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

number of attempts to approximate them. Recently, Duetting et al. (2019) presented a method, RegretNet, for learning approximately incentive compatible mechanisms given samples from the bidder valuations. They parameterize the auction as a neural network, and learn to maximize revenue, and maintain strategyproofness, by gradient descent. This learning approach to auction design can replicate some known-optimal auctions, and can learn good auctions in settings where the optimal auction is not known. It has been extended in a variety of ways (Curry et al., 2020; Tacchetti et al., 2019; Golowich, Narasimhan, and Parkes, 2018; Kuo et al., 2020; Feng, Narasimhan, and Parkes, 2018; Rahme, Jelassi, and Weinberg, 2021; Rahme, Jelassi, Bruna, et al., 2020; Peri et al., 2021).

The RegretNet architecture has architectural features to ensure that no item is over-allocated, and that each bidder receives the correct amount of goods. This is accomplished through a combination of softmax and minimum operations, depending on the exact bidder demand constraints.

However, this approach only works when some ad-hoc combination of these operations is sufficient to enforce the constraints. This is not always possible. Consider the case where the auctioneer needs to ensure that every participant receives exactly k items. Such equality constraints cannot be enforced using the min-of-softmax approach.

We observe that an auction allocation for bidders whose utilities are linear functions of their valuations amounts to a matching assigning items to bidders. Using this observation, we present an alternative approach which explicitly applies matching constraints on the output, using the Sinkhorn algorithm (Cuturi, 2013) to solve a discrete matching problem as part of an end-to-end differentiable architecture.

This approach is among the first attempts to use the techniques of differentiable optimization (Agrawal et al., 2019) for learned mechanism design – concurrent work (Liu et al., 2021) uses a differentiable sorting operator to learn generalized second-price auctions. By simply modifying the constraints in our matching problem, our proposed approach can produce feasible allocations without changes to the overall network architecture for many different bidder demand constraints. Our approach successfully recovers known optimal mechanisms in multiple settings including settings (such as exactly-one-demand auctions) in which RegretNet would be unable to guarantee that its output allocations adhere to the constraints.

2 RELATED WORK

Auction Design and Learning Auction mechanisms are functions from bids to allocations and payments; if one assumes sample access to the valuation distribution, it is natural to treat auction design as a learning problem, and there has been much work in this area.

One thread of work has been learning-theoretic, determining the sample complexity for various known families of auctions to estimate properties like revenue (M.-F. F. Balcan, Sandholm, and Vitercik, 2016; Cole and Roughgarden, 2014; Morgenstern and Roughgarden, 2016) or incentive compatibility (M.-F. Balcan, Sandholm, and Vitercik, 2019).

Another thread of work, sometimes called "differentiable economics", represents auction mechanisms using general parametric function approximators, and attempts to optimize them using gradient descent. (Duetting et al., 2019) introduce several neural network architectures to find revenue-maximizing auctions, including RochetNet, which works for single-agent auctions and is strategyproof by constructions, and RegretNet, which represents auctions as a general neural network and includes a term in the loss function to enforce strategyproofness.

Further work has built directly on both RochetNet and RegretNet (Kuo et al., 2020; Curry et al., 2020; Rahme, Jelassi, and Weinberg, 2021; Rahme, Jelassi, Bruna, et al., 2020). Others have taken a similar approach but applied to different mechanism design problems, including finding welfare-maximizing auctions (Tacchetti et al., 2019) and facility location (Golowich, Narasimhan, and Parkes, 2018). Other applications of gradient-based methods to problems in mechanism design include (Heidekrüger et al., 2021; Weissteiner and Seuken, 2020).

Single-agent auction learning The case of selling multiple items to a single agent is reasonably well understood. Rochet (1987) gives a characterization of strategyproof single-agent mechanisms – their utility functions must be monotone and convex. Based on this characterization, Duetting et al. (2019) design RochetNet, a network architecture for single-agent mechanism learning which is guaranteed to be perfectly strategyproof. Shen, Tang, and Zuo (2019) also provide network architectures for the single-agent setting which can be made strategyproof by construction. The authors of both works are able to learn mechanisms and then prove them optimal using the theoretical framework of Daskalakis, Deckelbaum, and Tzamos (2015).

We wish to explicitly contrast these architectures with our own approach. In a single-agent setting, they work better than a RegretNet-style architecture, and as mentioned can always guarantee perfect strategyproofness. In this work, however, we focus on general architectures which can work in both single- and multi-agent settings.

Optimal Transport and the Sinkhorn Algorithm Optimal transport (Kantorovich, 1942) is the problem of moving one set of masses to another mass while minimizing some cost function. In its infinite dimensional form, where it amounts to finding the cost-minimizing joint distribution between two marginal continuous probability distributions, it has wide applications in pure mathematics (Villani, 2003) as well as machine learning (Arjovsky, Chintala, and Bottou, 2017; Genevay, Peyré, and Cuturi, 2018). Interestingly, Daskalakis, Deckelbaum, and Tzamos (2015) and Kash and Frongillo (2016) use the mathematical tools of infinite-dimensional optimal transport in the context of mechanism design theory.

Our work here is not directly related. By contrast, we use the discrete form of optimal transport, which is essentially a formulation of minimum cost bipartite matching (in our case, between agents and items). This discrete problem can be numerically solved in a number of ways, see Peyré, Cuturi, et al. (2019) for an overview. In particular, we focus on the Sinkhorn algorithm (Cuturi, 2013): a fast, GPU-parallelizable iterative method for solving the entropy-regularized version of the optimal transport problem, which can be used as a differentiable bipartite matching operation (Mena et al., 2018; Grover et al., 2019; Tay et al., 2020; Emami and Ranka, 2018; Cuturi, Teboul, and Vert, 2019). We use the Sinkhorn algorithm to compute matchings between agents and items.

Differentiable optimization and deep learning Recently, there has been broad interest in mixing convex optimization problems with deep learning. For many convex optimization problems, the derivative of the optimal solution with respect to parameters of the objective or constraints is well defined, so it is possible to define a neural network layer that will output a feasible, optimal solution to some optimization problem. This is useful if one wants to use optimization with deep learning in a "predict and optimize" pipeline (Ferber et al., 2020; Wilder, Dilkina, and Tambe, 2019), to enforce that network outputs satisfy some constraints, or if interesting operations can be formulated in terms of optimization problems (Gould, Hartley, and Campbell, 2019).

One family of approaches (e.g., Agrawal et al., 2019; Amos and Kolter, 2017) involves solving the convex optimization problem using standard solvers, then using the implicit function theorem to compute the gradient for the backward pass. In a contrasting approach, when using an iterative method to solve the optimization problem, it is also possible to use automatic differentiation to simply backpropagate through the numerical operations performed by the solver. We employ this latter approach with the aforementioned Sinkhorn algorithm in order to compute feasible matchings in a differentiable manner

3 DIFFERENTIABLE ECONOMICS AND COMBINATORIAL OPTIMIZATION

General auction setting We consider an auction setting with n bidders and m items. Each bidder i has a private valuation function, $v_i : [0,1]^m \to \mathbb{R}_{\geq 0}$, that maps any subset of the items to a real number. We assume that v_i is drawn from a known distribution F_i, but the realized valuation v_i is private and unknown to the auctioneer. Each bidder i then submits their bids $b_i \in \mathbb{R}^m$. Let $b = (b_1, b_2, ..., b_n)$ be the bids from all bidder. The auction mechanism is then a combination of an allocation mechanism and payment mechanism, $(g(b), p(b))$. The g outputs an allocation $(g_1, g_2, ..., g_n)$ where g_{ij} is the probability of allocating item j to bidder i. The payment mechanism p outputs $(p_1, p_2, ..., p_n)$ where p_i is how much bidder i is charged. Each bidder then receives some utility $u_i(v_i, b) = v_i(g_i(b)) - p_i(b)$.

Quasilinear utilities, strategyproofness, and individual rationality Allowing a distinct valuation for every combination of items may result in a combinatorial explosion. A simplifying assumption is that a bidder may have a single valuation for each item. Then their valuation v_i is simply a vector in $\mathbb{R}^m$, and their utility is simply $u_i(v_i, b) = \langle v_i, g_i(b) \rangle - p_i(b)$.

Since a bidder's true valuation is private they are free to strategically report bids to maximize their utility. Thus, a we require our mechanism to be **strategyproof** or **dominant-strategy incentive compatible (DSIC)**, meaning each bidder's utility is maximized by reporting truthfully i.e. $u_i(v_i, (v_i, b_{-i})) \geq u_i(v_i, (v_i', b_{-i})$ for any other v_i' where b_{-i} are all the bids except the ith bidder. **Regret** is defined by the following equation: $\text{rgt}_i(v_i, b) = \max_{v_i'} u_i(v_i, (v_i', b_{-i})) - u_i(v_i, b)$, and represents the utility the bidder could have gained from lying in their bid. We can then also say that when a mechanism is DSIC, the regret for truthful bidding should be 0 for every bidder i with truthful valuation v_i, for any opponent bids b_{-i}.

Another desirable property for an auction for it to be

(ex post) **individually rational**. This means each bidder receives a non-negative utility from the auction i.e. $u_i(v_i, (v_i, b_{-i})) \geq 0$ for all bidders i, all possible valuations v_i, and all other bids b_{-i}. Without this guarantee bidders might choose not to participate in the auction at all because they are concerned about being left worse off than they were before the auction.

3.1 Bidder Demand Types

Here we define common bidder demand constraints which are used in the experiments.

The value of a bundle of goods for a **unit-demand** bidder is equal to the maximum value of a single item in the bundle: $v_i(S) = \max_{j \in S} v_i(j)$. In this sort of auction, there is no need to consider allocations of more than one item to each bidder since those provide that same utility as only allocating the most desirable item within that bundle to that bidder. Thus, by restricting allocations to allocate at most one item per bidder, we can again treat bidder utility as quasilinear. More generally, there can be k-unit-demand auctions where a bidder's value of a bundle of goods is the sum of the top k items in the bundle (H. Zhang and Conitzer, 2020).

We also consider **exactly-one demand**, where each bidder must be assigned exactly one item. In this setting, the bidder cannot be assigned more or fewer than one item and receives the value of the assigned item. Additionally, this demand-type can be extended to a more general exactly-k demand setting where each bidder must be allocated k items.

In all these settings, in general allocations might be randomized so that bidders can receive items with different probabilities. This corresponds to allowing fractional allocations, so that in the unit-demand and exactly-one-demand settings, the one item the bidder receives can be a mixture of fractions of multiple items.

3.2 RegretNet

The RegretNet architecture consists of two neural networks, the allocation network (denote it g) which outputs a matrix representing the allocation of each item to each agent and payment network (denote it p) which outputs the payments for each bidder (Duetting et al., 2019).

RegretNet guarantees individual rationality by making the payment network use a sigmoid activation, outputting a value $\tilde{p} \in [0, 1]$. The final payment is then $\left(\sum_j v_{ij} g_{ij}\right) \tilde{p}$, ensuring that the utility of each bidder is non-negative.

(Duetting et al., 2019) present network architectures to learn under additive, unit-demand and combinatorial valuations. Their architecture utilizes a traditional feed-forward neural network with modifications in the output layer to ensure a valid allocation matrix. In the additive setting, the allocation probabilities for each item must be at most one: this is enforced by taking a row-wise softmax on the network outputs. For unit-demand auctions, each bidder wants at most one item, so the network takes a row-wise softmax of one matrix and a column-wise softmax of another to ensure item allocation is less than one. The final output is then the elementwise min of these two, ensuring that both item allocation and unit-demand constraints are enforced.

For training, the RegretNet architecture uses gradient descent on an augmented Lagrangian which includes terms to maximize payment while also containing terms to enforce the constraint that regret should be zero:

$$\mathcal{L}_{\boldsymbol{\lambda}}(\boldsymbol{v}) = -\sum_i p_i(\boldsymbol{v}) + \sum_i \lambda_i \widehat{\mathrm{rgt}}_i(\boldsymbol{v}) + \frac{\rho}{2} \sum_i \left(\widehat{\mathrm{rgt}}_i(\boldsymbol{v})\right)^2 \quad (1)$$

Here $\widehat{\mathrm{rgt}}_i$ denotes an empirical estimate of regret produced by running gradient ascent on player i's portion of the network input to maximize their utility, computing their possible benefit from strategically lying. During training, the Lagrange multipliers λ are gradually maximized, incentivizing the minimization of regret.

4 BIPARTITE MATCHING FOR AUCTIONS

The auction allocation problem is equivalent to finding a minimum-cost bipartite matching between bidders and items, for some cost matrix. We solve this bipartite matching problem by formulating it as an optimal transport problem. Using the Sinkhorn algorithm, we are able to solve this optimal transport problem in an end-to-end differentiable way, yielding a valid matching that our network can use to learn an optimal allocation mechanism.

4.1 Matching Linear Program

For marginal vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ – here representing constraints for agents and items respectively, each with a dummy component that represents no item or no agent – the optimal transport problem is as follows.

$$\min_{\boldsymbol{P}} \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} \boldsymbol{P}_{ij} \boldsymbol{C}_{ij} \text{ s.t. } \boldsymbol{P} \mathbb{1}_{m+1} = \boldsymbol{a},\ \mathbb{1}_{n+1}^T \boldsymbol{P} = \boldsymbol{b} \quad (2)$$

By specifying $\boldsymbol{a}$ and $\boldsymbol{b}$, we can specify the demand constraints of the problem:

- k-demand: $\boldsymbol{a}_i = k$, $\boldsymbol{a}_{n+1} = m$, $\boldsymbol{b}_j = 1$, $\boldsymbol{b}_{m+1} = kn$

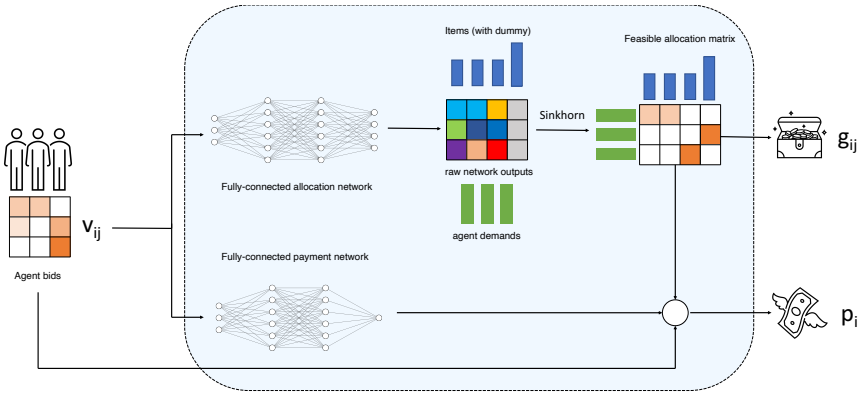


Figure 1: A schematic showing the Sinkhorn-based mechanism network. The agents' bids (here, assumed to be truthful) on each item act as input to a feedforward network, whose output is used as the cost matrix $\boldsymbol{C}_{ij}$ for the Sinkhorn algorithm. Marginals, specified separately, ensure that the Sinkhorn output is a feasible allocation. The payment network charges a fraction of the value of items that each agent wins.

- exactly-k-demand: $\boldsymbol{a}_i = k$, $\boldsymbol{a}_{n+1} = m - kn$, $\boldsymbol{b}_j = 1$, $\boldsymbol{b}_{m+1} = 0$

Here the $n+1$ and $m+1$ values of the marginals represent the dummy agent and item respectively. Concretely, the marginal constraint for agents represents how many items that agent must receive; if there is a dummy item, it can be fully or partially met by assigning that dummy item, representing receiving nothing. Likewise, the constraints for items represent how much of each item must be allocated; if there are dummy agents, these constraints can be met by allocating to the dummy agent (representing allocation to nobody).

More complex constraints, such as different k for different agents, can also be accommodated. Also, note in the exactly-k-demand settings there must be at least kn items, so $m \geq kn$. An optimal solution to the matching LP will always be a permutation matrix (Birkhoff, 1946), or more generally on one of the extreme points of the constraint set.

4.2 Entropy-Regularized Optimal Transport and the Sinkhorn Algorithm

The problem in Equation (2) can be solved as a linear program. However, Cuturi (2013) observed that by adding an entropy penalty to the problem, it can be solved in a practical and scalable way using the matrix-scaling Sinkhorn algorithm. The new objective of the problem is

$$\min_{\boldsymbol{P}} \sum_{i=1}^{N}\sum_{j=1}^{M} \boldsymbol{P}_{ij}\boldsymbol{C}_{ij} + \epsilon \sum_{i=1}^{N}\sum_{j=1}^{M} \boldsymbol{P}_{ij} \log \boldsymbol{P}_{ij}$$

Given marginals and a cost matrix, Cuturi (2013) provides an iterative algorithm to solve this problem. It can be implemented in a variety of ways; we use the following stabilized log-domain updates (Peyré, Cuturi, et al., 2019):

$$\boldsymbol{f}_i = -\epsilon \log \left(\sum_j \exp\left(-\frac{C_{ij} - \boldsymbol{g}_j}{\epsilon} \right) \right) + \epsilon \log \boldsymbol{a}$$

$$\boldsymbol{g}_j = -\epsilon \log \left(\sum_i \exp\left(-\frac{C_{ij} - \boldsymbol{f}_i}{\epsilon} \right) \right) + \epsilon \log \boldsymbol{b}$$

where $\boldsymbol{P}_{i,j} = e^{\boldsymbol{f}_i/\epsilon} e^{-C_{ij}/\epsilon} e^{\boldsymbol{g}_j/\epsilon}$. In addition to enabling the use of the iterative algorithm, the regularization term makes the problem strongly convex. Thus, the derivative of the optimal solution with respect to the cost matrix is well-defined. To approximate this derivative, we choose to unroll several iterations of the Sinkhorn updates (which consist only of differentiable operations) and use automatic differentiation.

The ϵ parameter controls the strength of the regularization. If it is large the output $\boldsymbol{P}$ will be nearly uniform; as it goes to zero we recover the original problem and $\boldsymbol{P}$ will be a permutation matrix. For our purposes, we set ϵ large enough to avoid vanishing gradients, but small enough that the output allocations are still nearly permutation matrices.

4.3 Network Architecture

Our architecture, like RegretNet, uses a pair of networks to compute allocations and payments. We denote the allocation and payment networks g^w and p^w respectively, where w are the network weights.

The main distinction from RegretNet is in the allocation mechanism. Like RegretNet we utilize a traditional feedforward neural network that takes all given bids as input. We use the output of the network as the cost matrix $\boldsymbol{C}_{ij}$ to the Sinkhorn algorithm (if there are dummy agents or items, they receive cost 0). Marginals are specified separately depending on the problem setting.

Finally, the output of the Sinkhorn algorithm (after truncating the row and column for dummy variables) provides the final probabilities g_{ij}^w of allocating item j to bidder i.

As in RegretNet, the final payment for bidder i is $\left(\sum_j v_{ij} g_{ij}\right) \tilde{p}_i^w$ where $\tilde{p}_i^w \in [0,1]$ is the ith output from a feedforward payment network. This ensures that the mechanism is individually rational as the bidders will never be charged more than their utility gained from the allocated items.

4.4 Settings That RegretNet Cannot Represent

RegretNet provides three different architectures depending on bidder demand type and valuation functions. For its unit-demand and combinatorial architectures, it employs a "min-of-softmax" approach to ensure both that agents receive at most 1 item or bundle, and items are not overallocated.

We particularly emphasize the case of exactly-k demand, which describes the situation where the auction designer is obligated to ensure that every participant receives k items, no matter what they bid. This captures, for instance, an offline version of the classic Display Ads setting (Feldman et al., 2009), with a special case of no-free-disposal requirements, where every ad buyer is contractually guaranteed a certain number of ad slots. For multiple agents and multiple items, RegretNet cannot represent this demand type via the min-of-softmax approach.

(In some special cases, ad-hoc changes could be made to the original RegretNet architecture to support settings beyond additive and unit-demand. For example, to enforce "at-most-k" demand, one can just scale the unit-demand outputs by a factor of k. In the special single-agent case, it is possible to produce exactly-k demand by removing the dummy item which allows for the possibility of allocating no item.)

By contrast, in multi-agent settings with exactly-k demand, the Sinkhorn-based architecture can ensure feasible outputs by simply specifying the correct marginals. Our network architecture is also independent of the demand type since only the marginals change rather than the entire network output structure.

Algorithm 1 Sinkhorn allocation procedure

Input: Bid $\boldsymbol{v} \in \mathbb{R}^{m \times n}$, allocation network output $f(\boldsymbol{v};\theta)$, marginals $\boldsymbol{a} \in \mathbb{R}^{n+1}$ and $\boldsymbol{b} \in \mathbb{R}^{m+1}$, Sinkhorn epsilon schedule $\epsilon_1, \cdots, \epsilon_T$, tolerance t
Output: Feasible allocation matrix $\boldsymbol{g}_{ij} \in \mathbb{R}^{m \times n}$
Initialize: $\boldsymbol{f} = 0^{n+1}, \boldsymbol{g} = 0^{m+1}, \boldsymbol{C} = f(\boldsymbol{v};\theta)$
for ϵ in $\epsilon_0, \cdots, \epsilon_T$ **do**
  $\boldsymbol{P}_{ij} = e^{\boldsymbol{f}_i/\epsilon} e^{-\boldsymbol{C}_{ij}/\epsilon} e^{\boldsymbol{g}_j/\epsilon}$
  while $\max_i |\sum_j \boldsymbol{P}_{ij} - \boldsymbol{a}_i| / \boldsymbol{a}_i \geq t$ **do**
    $\boldsymbol{f}_i = -\epsilon \log\left(\sum_j \exp\left(-\frac{\boldsymbol{C}_{ij} - \boldsymbol{g}_j}{\epsilon}\right)\right) + \epsilon \log \boldsymbol{a},\ \boldsymbol{g}_j = -\epsilon \log\left(\sum_i \exp\left(-\frac{\boldsymbol{C}_{ij} - \boldsymbol{f}_i}{\epsilon}\right)\right) + \epsilon \log \boldsymbol{b}$
    $\boldsymbol{P}_{ij} = e^{\boldsymbol{f}_i/\epsilon} e^{-\boldsymbol{C}_{ij}/\epsilon} e^{\boldsymbol{g}_j/\epsilon}$
  end while
end for
$\boldsymbol{g}_{ij} = \boldsymbol{P}_{ij}$ for $i = 1..n, j = 1..m$

4.5 Optimization and Training

Like RegretNet, our training objective is Equation (1). For each batch we estimate the empirical regret, $\widehat{\mathrm{rgt}}_i(w)$ by performing gradient ascent for each bidder on the network inputs, approximating a misreport for each bidder.

In our experiments, the network for both the payment and allocation network had 2 hidden layers of 128 nodes with Tanh activation functions, optimized using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 10^{-3}. The training set consisted of $2^{19} = 524,288$ valuation profiles. We used batch sizes of 4,096 and ran a total of 100 epochs. For each batch, we computed the best misreport using 25 loops of gradient ascent with a learning rate 0.1. We also incremented ρ every two batches and and updated the Lagrange multipliers, λ, every 100 batches. Experiments were all run on single 2080Ti GPUs, on either a compute node or workstation with 32GB RAM, using PyTorch (Paszke et al., 2019).

For evaluation, we used 1,000 testing examples and optimized the misreports of these examples for 1,000 iterations with learning rate 0.1. For each of the 1,000 samples we use ten random initialization points for the misreport optimization and use the maximum regret from these. At both train and test time we used the Sinkhorn algorithm with an ϵ parameter of 0.05, and a stopping criterion of a relative tolerance of 0.01. Additionally we used an ϵ schedule (as suggested by Schmitzer (2019) and Cuturi, Teboul, and Vert (2019)) of 10 steps from 1 down to the final value of 0.05. Pseudocode for computing the allocation network output is given in algorithm 1.

5 EXPERIMENTS

5.1 Optimal single-agent mechanisms

We start by recovering two known optimal mechanisms in the single bidder case. We emphasize that if the only goal were to learn single-agent auctions, architectures such as RochetNet (Duetting et al., 2019) or that of (Shen, Tang, and Zuo, 2019) would work better. Our architecture works for both single-agent and multi-agent auctions. Following previous work (Duetting et al., 2019; Rahme, Jelassi, Bruna, et al., 2020; Rahme, Jelassi, and Weinberg, 2021), we use these single-agent experiments as a test case to make sure it recovers some known optimal mechanisms, before continuing on to multi-agent settings where optimal auctions are not known and where RochetNet or (Shen, Tang, and Zuo, 2019) cannot work.

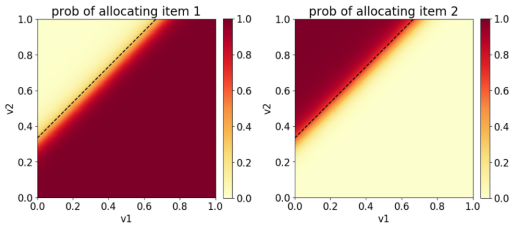
Unit-demand We first recover the optimal mechanism in the unit-demand single-agent two-item setting with item values drawn from $U[0, 1]$. The optimal mechanism, also approximately recovered in Duetting et al. (2019), is from (Pavlov, 2011): it is to offer each good for a price of $\frac{\sqrt{3}}{3}$. The learned allocation probabilities are shown in Figure 2b with the boundary of the optimal analytic mechanism denoted by a dotted line. The x and y axis are the valuation of the bidder for item one and two respectively. The color represents the allocation probability output by the learned mechanism with the darker color corresponding to higher probability. Quantitatively, the learned mechanism has small regret and slightly higher than the optimal mechanism likely due to the small amount of regret.

Exactly-one demand The second optimal mechanism was a deterministic mechanism in the same single agent, two-item setting with valuations on $U[0, 1]$. However, this time the agent will be allocated exactly one item (instead of at most one). Kash and Frongillo (2016) shows that the optimal mechanism is to offer one of the items for free and the other for a price of $\frac{1}{3}$.

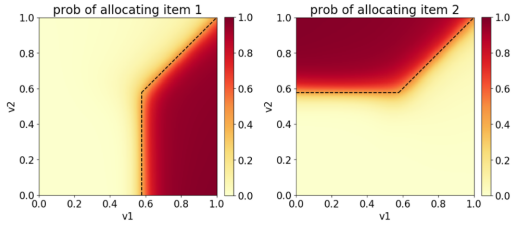
The boundary of this optimal mechanism is shown in Figure 2a as the black dotted line. The revenue in Figure 1 is slightly higher than the optimal, again likely due to the presence of small regret.

5.2 Multi-agent setting

We now experiment in settings where the optimal strategyproof mechanism is unknown. Specifically, we study 2 bidders and either 3 or 15 items where valuations for each item are drawn from $U[0, 1]$. A typical analytic baseline is a separate Myerson auction for each item, but this only works in additive settings. Instead, as a baseline, we compute the revenue from an (Vickrey, 1961; Clarke, 1971; Groves, 1973). The VCG auction is strategyproof, and maximizes welfare rather than revenue. It works by charging each bidder the harm they cause to other bidders, which in the settings we test may be very small or even close to zero. Table 2 contains the revenues for both mechanism (unsurprisingly higher for the learned auction) as well as the average observed regret on the testing sample, while figures 3a and 3b show training plots for the two agent, 3 item, exactly-one demand case.



(a) Exactly-One



(b) Unit-Demand

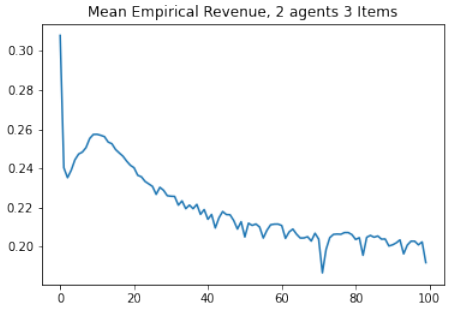
Figure 2: Heatmaps of learned mechanisms for single-bidder two-item with $v_1, v_2 \sim U[0, 1]$ are shown in (a)-(b). The optimal mechanism boundaries are the black dotted lines.

5.3 Comparison To RegretNet

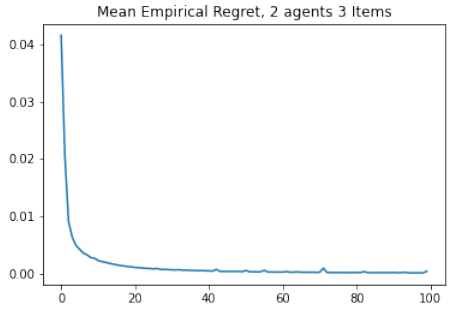
RegretNet is capable of representing unit-demand auctions, but not the exactly-one setting. We compare performance to it in this case; results are shown in Table 2. (Runtimes are significantly faster than those reported in Duetting et al. (2019) likely due to our much larger batch sizes.) For the one-agent, two item setting, both architectures approximate the optimal mechanism so performance is similar. Performance is also similar for the 2 agent, 3 item setting. In the 15-item setting, revenue with the Sinkhorn architecture is somewhat smaller. In all cases the computational cost of RegretNet is significantly lower, as it requires only two softmax operations instead of many iterations of the Sinkhorn algorithm.

Setting	Rev	Regret	Opt Rev	Train Time
Unit-demand	0.397 (0.248)	< 0.001 (< 0.001)	0.393	3h25m
RegretNet Unit-Demand	0.381 (0.258)	< 0.001 (< 0.001)	0.393	18m34s
Exactly-one	0.079 (0.127)	0.001 (0.001)	0.069	1h21m

Table 1: Table of mean revenue and regret from learned mechanism in test set alongside revenue of optimal mechanism for 1 agent, 2 items, with item valuations distributed independently on $U[0, 1]$.



(a) Mean revenue for exactly-one demand, 2 agent 3 item setting



(b) Mean empirical regret for exactly-one demand, 2 agent 3 item setting

Setting	Agents	Items	Rev	Regret	VCG Rev	Time
Unit	2	3	.876 (.322)	.001 (.001)	.048	6h20m
RegretNet Unit	2	3	.878 (.337)	$< .001$ ($< .001$)	.048	20m28s
Unit	2	15	.886 (.113)	.001 ($< .001$)	.002	3h10m
RegretNet Unit	2	15	.999 (.123)	$< .001$ ($< .001$)	.002	25m45s
Exactly-1	2	3	.194 (.064)	.004 (.008)	.049	2h38m
RegretNet Exactly-1	2	3	N/A	N/A	.049	N/A
Exactly-1	2	15	.571 (.030)	.003 (.015)	.002	1h29m
RegretNet Exactly-1	2	15	N/A	N/A	.002	N/A

Table 2: Table of mean revenue and regret from learned mechanism in test set along with revenue from VCG auction, for 2 agents with valuations distributed independently on $U[0, 1]$

6 STRENGTHS, LIMITATIONS, AND POTENTIAL IMPACTS

Structurally, our architecture has the benefit of remaining the same for any quasi-linear bidder utilities and constraints, with only the Sinkhorn marginals changing. The use of the Sinkhorn algorithm allows us to enforce these constraints explicitly and even tackle new settings, such as exactly-k demand, that the existing RegretNet architecture would be unable to handle.

Our Sinkhorn-based architecture has a higher computational cost than the comparable RegretNet architecture, as the Sinkhorn algorithm requires many iterations. The Sinkhorn algorithm also requires two additional hyperparameters, ϵ and a tolerance before the algorithm terminates.

We find that the ϵ parameter plays a crucial role in the training of the algorithm. If the ϵ is too small, training fails, likely due to vanishing gradients. However, with too large of an ϵ the mechanism becomes too smooth and unable to approximate the sharp boundaries that tend to show up in optimal mechanisms leading to suboptimal revenue. (For further discussion of ϵ, see Appendix A in the supplemental material.) Our architecture has an inductive bias towards deterministic allocations. This may pose a limitation where revenue-maximizing mechanisms are nondeterministic, but it may be an advantage if they are deterministic or if determinism is desirable. By setting ϵ small enough it is possible to ensure near-determinism in allocations. However, because the mechanisms are trained under a higher ϵ, there are regions where the price charged becomes too high, increasing regret. Because training directly with small ϵ is difficult, we cannot directly guarantee that we train deterministic mechanisms.

Following previous work that uses neural networks to approximate optimal auctions (e.g. Duetting et al., 2019; Shen, Tang, and Zuo, 2019; Rahme, Jelassi, Bruna, et al., 2020; Rahme, Jelassi, and Weinberg, 2021), we train on synthetic data. While in principle deep-learning-based methods could be used to train on truthful bids from real-world bidders, and doing so would be extremely interesting, such data is hard to come by. We see learned auctions as, at least in part, a tool for pushing theory forward, so using synthetic data remains interesting when it is drawn from valuation distributions for which analytic solutions have remained out of reach.

Our work, like most work in differentiable economics, uses sample-based approximations to estimate revenue and regret. It would be desirable to understand the error due to sampling. Some generalization bounds in previous mechanism design papers (such as the quantile regret from Duetting et al. (2019)) apply directly to our case; others (such as the techniques in M.-F. Balcan, Sandholm, and Vitercik (2019)) do not but might be extended.

There are clear ethical implications of revenue-maximizing auction design, in terms of potential impact on bidders and society as a whole. We do release code which can in principle be used for revenue-maximizing mechanism design. However, since our work is still closely to the theoretical models discussed above, we do not see any direct ethical implications – it is very unlikely that deep-learned auctions will be directly deployed in the near future.

7 CONCLUSION AND FUTURE WORK

We have presented a new architecture for learned auctions which uses the Sinkhorn algorithm to perform a differentiable matching operation to compute an allocation. Our architecture works for a variety of bidder demand constraints by encoding them into the marginals used in the Sinkhorn algorithm. This new architecture allows the network to guarantee valid allocations in settings where RegretNet could not.

We show that our approach successfully recovers optimal mechanisms in settings where optimal mechanisms are known, and achieves good revenue and low regret in larger settings. Future work might include extending the Sinkhorn architecture to more directly support randomized allocations, further computational improvements, or further extensions to other mechanism design problems where the allocation decision can be expressed as a matching.

Acknowledgments. Curry and Dickerson were supported in part by NSF CAREER Award IIS-1846237, NSF D-ISN Award #2039862, NSF Award CCF-1852352, NIH R01 Award NLM-013039-01, NIST MSE Award #20126334, DARPA GARD #HR00112020007, DoD WHS Award #HQ003420F0035, and a Google Faculty Research Award. Goldstein was supported by the ONR MURI program, the AFOSR MURI Program, the National Science Foundation (DMS-1912866), the JP Morgan Faculty Award, and the Sloan Foundation. We thank Ahmed Abdelkader, Kevin Kuo, and Neehar Peri for comments on earlier drafts of this work, and Jason Hartline for helpful commentary.

References

Agrawal, Akshay et al. (2019). "Differentiable Convex Optimization Layers". In: *Neural Information Processing Systems (NeurIPS)*.

Amos, Brandon and J Zico Kolter (2017). "Optnet: Differentiable optimization as a layer in neural networks". In: *International Conference on Machine Learning (ICML)*.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). "Wasserstein Generative Adversarial Networks". In: *International Conference on Machine Learning (ICML)*.

Balcan, Maria-Florina, Tuomas Sandholm, and Ellen Vitercik (2019). "Estimating Approximate Incentive Compatibility". In: *Economics and Computation (EC)*.

Balcan, Maria-Florina F, Tuomas Sandholm, and Ellen Vitercik (2016). "Sample complexity of automated mechanism design". In: *Advances in Neural Information Processing Systems*, pp. 2083–2091.

Birkhoff, Garrett (1946). "Tres observaciones sobre el algebra lineal". In: *Univ. Nac. Tucuman, Ser. A* 5, pp. 147–154.

Clarke, Edward H (1971). "Multipart pricing of public goods". In: *Public choice*, pp. 17–33.

Cole, Richard and Tim Roughgarden (2014). "The sample complexity of revenue maximization". In: *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pp. 243–252.

Cramton, Peter (2017). "Electricity market design". In: *Oxford Review of Economic Policy* 33.4, pp. 589–612.

Curry, Michael et al. (2020). "Certifying Strategyproof Auction Networks". In: *Neural Information Processing Systems (NeurIPS)*.

Cuturi, Marco (2013). "Sinkhorn distances: Lightspeed computation of optimal transport". In: *Neural Information Processing Systems (NeurIPS)*.

Cuturi, Marco, Olivier Teboul, and Jean-Philippe Vert (2019). "Differentiable Ranking and Sorting using Optimal Transport". In: *Neural Information Processing Systems (NeurIPS)*.

Daskalakis, Constantinos, Alan Deckelbaum, and Christos Tzamos (2015). "Strong Duality for a Multiple-Good Monopolist". In: *Economics and Computation (EC)*.

Duetting, Paul et al. (2019). "Optimal Auctions through Deep Learning". In: *International Conference on Machine Learning (ICML)*.

Edelman, Benjamin, Michael Ostrovsky, and Michael Schwarz (2007). "Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords". In: *American Economic Review* 97.1, pp. 242–259.

Emami, Patrick and Sanjay Ranka (2018). "Learning permutations with sinkhorn policy gradient". In: *arXiv preprint arXiv:1805.07010*.

Feldman, Jon et al. (2009). "Online Ad Assignment with Free Disposal". In: *Internet and Network Economics*. Springer Berlin Heidelberg, pp. 374–385.

Feng, Zhe, Harikrishna Narasimhan, and David C. Parkes (2018). "Deep Learning for Revenue-Optimal Auctions with Budgets". In: *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*.

Ferber, Aaron et al. (2020). "Mipaal: Mixed integer program as a layer". In: *AAAI Conference on Artificial Intelligence*.

Genevay, Aude, Gabriel Peyré, and Marco Cuturi (2018). "Learning generative models with sinkhorn divergences". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Golowich, Noah, Harikrishna Narasimhan, and David C. Parkes (2018). "Deep Learning for Multi-Facility Location Mechanism Design". In: *International Joint Conference on Artificial Intelligence (IJCAI)*.

Gould, Stephen, Richard Hartley, and Dylan Campbell (2019). "Deep declarative networks: A new hope". In: *arXiv preprint arXiv:1909.04866*.

Grover, Aditya et al. (2019). "Stochastic optimization of sorting networks via continuous relaxations". In: *arXiv preprint arXiv:1903.08850*.

Groves, Theodore (1973). "Incentives in teams". In: *Econometrica: Journal of the Econometric Society*, pp. 617–631.

Heidekrüger, Stefan et al. (2021). "Equilibrium Learning in Combinatorial Auctions: Computing Approximate Bayesian Nash Equilibria via Pseudogradient Dynamics". In: *arXiv preprint arXiv:2101.11946*.

Kantorovich, Leonid V (1942). "On the translocation of masses". In: *Dokl. Akad. Nauk. USSR (NS)*. Vol. 37, pp. 199–201.

Kash, Ian A and Rafael M Frongillo (2016). "Optimal auctions with restricted allocations". In: *Proceedings of the 2016 ACM Conference on Economics and Computation*, pp. 215–232.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Krishna, Vijay (2009). *Auction theory*. Academic press.

Kuo, Kevin et al. (2020). "ProportionNet: Balancing Fairness and Revenue for Auction Design with Deep Learning". In: *arXiv preprint arXiv:2010.06398*.

Leyton-Brown, Kevin, Paul Milgrom, and Ilya Segal (2017). "Economics and computer science of a radio spectrum reallocation". In: *Proceedings of the National Academy of Sciences (PNAS)* 114.28, pp. 7202–7209.

Liu, Xiangyu et al. (2021). "Neural Auction: End-to-End Learning of Auction Mechanisms for E-Commerce Advertising". In: *arXiv preprint arXiv:2106.03593*.

Manelli, Alejandro M. and Daniel R. Vincent (2006). "Bundling as an optimal selling mechanism for a

multiple-good monopolist". In: *J. Econ. Theory* 127.1, pp. 1–35.

Mena, Gonzalo et al. (2018). "Learning latent permutations with gumbel-sinkhorn networks". In: *arXiv preprint arXiv:1802.08665*.

Milgrom, Paul (2017). *Discovering prices: auction design in markets with complex constraints*. Columbia University Press.

Morgenstern, Jamie and Tim Roughgarden (2016). "Learning simple auctions". In: *Conference on Learning Theory*, pp. 1298–1318.

Myerson, Roger B (1981). "Optimal auction design". In: *Mathematics of Operations Research* 6.1, pp. 58–73.

Parsons, Simon, Juan A Rodriguez-Aguilar, and Mark Klein (2011). "Auctions and bidding: A guide for computer scientists". In: *ACM Computing Surveys (CSUR)* 43.2, pp. 1–59.

Paszke, Adam et al. (2019). "Pytorch: An imperative style, high-performance deep learning library". In: *arXiv preprint arXiv:1912.01703*.

Pavlov, Gregory (2011). "Optimal mechanism for selling two goods". In: *The BE Journal of Theoretical Economics* 11.1.

Peri, Neehar et al. (2021). "PreferenceNet: Encoding Human Preferences in Auction Design with Deep Learning". In: *arXiv preprint arXiv:2106.03215*.

Peyré, Gabriel, Marco Cuturi, et al. (2019). "Computational optimal transport: With applications to data science". In: *Foundations and Trends® in Machine Learning* 11.5-6, pp. 355–607.

Rahme, Jad, Samy Jelassi, Joan Bruna, et al. (2020). "A Permutation-Equivariant Neural Network Architecture For Auction Design". In: *arXiv preprint arXiv:2003.01497*.

Rahme, Jad, Samy Jelassi, and S Matthew Weinberg (2021). "Auction learning as a two-player game". In: *International Conference on Learning Representations (ICLR)*.

Rochet, Jean-Charles (1987). "A necessary and sufficient condition for rationalizability in a quasi-linear context". In: *Journal of mathematical Economics* 16.2, pp. 191–200.

Roth, Alvin E (2018). "Marketplaces, markets, and market design". In: *American Economic Review* 108.7, pp. 1609–58.

Schmitzer, Bernhard (2019). "Stabilized sparse scaling algorithms for entropy regularized transport problems". In: *SIAM Journal on Scientific Computing* 41.3, A1443–A1481.

Shen, Weiran, Pingzhong Tang, and Song Zuo (2019). "Automated mechanism design via neural networks". In: *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*, pp. 215–223.

Tacchetti, Andrea et al. (2019). "A neural architecture for designing truthful and efficient auctions". In: *arXiv preprint arXiv:1907.05181*.

Tay, Yi et al. (2020). "Sparse sinkhorn attention". In: *International Conference on Machine Learning (ICML)*.

Vickrey, William (1961). "Counterspeculation, auctions, and competitive sealed tenders". In: *The Journal of finance* 16.1, pp. 8–37.

Villani, Cédric (2003). *Topics in optimal transportation*. American Mathematical Soc.

Weissteiner, Jakob and Sven Seuken (2020). "Deep Learning—Powered Iterative Combinatorial Auctions". In: *AAAI Conference on Artificial Intelligence*.

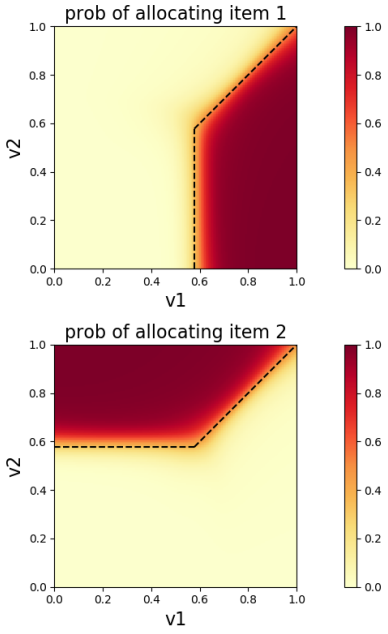
Wilder, Bryan, Bistra Dilkina, and Milind Tambe (2019). "Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization". In: *AAAI Conference on Artificial Intelligence*.

Yao, Andrew Chi-Chih (2017). "Dominant-strategy versus bayesian multi-item auctions: Maximum revenue determination and comparison". In: *Economics and Computation (EC)*.

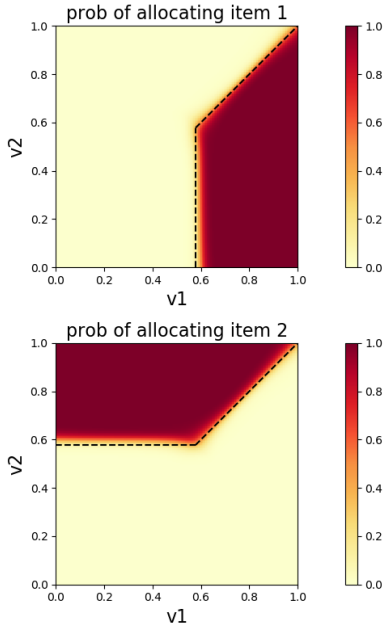
Zhang, Hanrui and Vincent Conitzer (2020). "Learning the Valuations of a k-demand Agent". In: *International Conference on Machine Learning (ICML)*.

A Effect of Sinkhorn Epsilon

Effect of Sinkhorn ϵ Figures 4b and 4a highlights the effect of the Sinkhorn parameters on the final mechanism. The lower the ϵ in the Sinkhorn algorithm the sharper the boundary becomes, as with less entropy regularization, the optimal matching is closer to deterministic. However, we found that very small values of epsilon led to problems during training; the learned mechanism would choose to never allocate items, likely due to vanishing gradients. One can make an almost-perfectly-deterministic mechanism by decreasing ϵ at test time, but this can increase regret as the learned payments no longer agree with the allocations.



(a) $\epsilon = 0.05$



(b) $\epsilon = 0.02$

Figure 4: The following figures illustrate the effect of Sinkhorn temperature on allocation mechanism (c) higher Sinkhorn ϵ value and (d) lower Sinkhorn ϵ value.