LEARNING REVENUE-MAXIMIZING AUCTIONS WITH NEURAL AFFINE MAXIMIZER

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning truthful, revenue-maximizing auctions is a central challenge in automated mechanism design and differentiable economics. Existing learning approaches that guarantee truthfulness typically discretize the outcome space into a finite menu, which suffers from the "curse of dimensionality" in large-scale auctions. In this work, we propose *Neural Affine Maximizer* (NAM), a discretization-free mechanism for learning truthful auctions, as well as a novel algorithm to learn NAMs. NAM guarantees truthfulness by building on affine maximizer auctions (AMAs) while replacing the conventional finite menu with a boosting function over the outcome space. We then parameterize the boosting function with neural networks and derive unbiased gradient estimators to enable first-order optimization. Experiments on instances with up to 15 buyers or 30 goods show that NAM is either competitive with or outperforms existing approaches. Specifically, with 3 buyers, 30 goods, and additive valuations, NAM discovers a truthful auction achieving 10.6% higher revenue and lower training time compared with state-of-the-art baselines.

1 INTRODUCTION

Designing a *truthful* (DSIC, strategy-proof) auction that achieves high expected revenue is a central challenge in auction design. While the single-item case admits an analytical optimum [38], multi-bidder multi-item settings remain notoriously challenging [49], with optima known only in special cases [12, 23, 35, 41, 58]. This motivates automated mechanism design [48, 49] that aims at designing learning algorithms to improve auctions' revenues.

Recently, differentiable economics—a technical paradigm—treats economic design as an end-to-end learning problem with neural networks and gradient-based optimization [46, 54, 60]. Within this paradigm, notable progress has been made in auction learning [10, 19, 55].

However, existing approaches for learning truthful multi-bidder auctions either *discretize* outcome spaces [10, 49, 55] or apply only in restrictive settings (*e.g.*, goods with unlimited supply [33]). The discretization step brings inefficiency in learning large-scale auctions. Specifically, to learn n-buyers, m-goods auctions, Curry et al. [10], Duan et al. [17], Sandholm and Likhodedov [49] represent Affine Maximizer Auctions (AMAs) to discretize the allocation space $(\Delta_n)^m$, where $\Delta_n := \{y \in \mathbb{R}_+^n : \sum_{i\in[n]} y_i \leq 1\}$ represents the space of feasible probabilistic allocations of one good among n buyers. There are $(n+1)^m$ deterministic allocations, growing combinatorially and rendering AMA representations not scalable. Duan et al. [18], Sandholm and Likhodedov [49] reduce the parameter count to $(n+1)\cdot 2^m$ via Virtual Valuation Combinatorial Auctions (VVCAs). However, VVCAs are less expressive than AMAs and the parameter count is still exponential in m.

To resolve the exponential complexity from discretization, we propose **Neural Affine Maximizer** (NAM)—a *discretization-free* mechanism as an alternative to AMA, as well as an algorithm to learn NAMs. Unlike discretized AMAs, NAM replaces the discrete outcome representation with a continuous boosting function $\lambda(\boldsymbol{x}) : (\Delta_n)^m \to \mathbb{R}$ to assign a boost to each outcome $\boldsymbol{x} \in (\Delta_n)^m$. We next parameterize $\lambda(\boldsymbol{x})$ with an Input Convex Neural Network (ICNN, [2]), a neural network that universally approximates convex functions. We prove that NAM preserves the same expressive power as AMAs, and establish a first-order algorithm to learn neural network parameters. Our algorithm

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

design is inspired by Ma et al. [33] to resolve the technical difficulties faced by the elimination of discretization step, which might be of independent interest.

Experiments with up to 15 buyers or 30 goods show that NAM is competitive with or outperforms existing discretization-based approaches consistently across different random seeds and settings. Specifically, on $U([0,1])$ valuations with 3 buyers and 30 goods, NAM yields 10.6% higher revenue with lower runtime compared with state-of-the-art baselines. [1]

2 PRELIMINARY

Auction Model We study a standard multi-buyer, multi-good auction with one seller (the auctioneer), n buyers, and m goods [10, 17, 19]. Buyer $i \in [n]$ is characterized by a private valuation vector $t_i \in \mathcal{T}_i \subseteq \mathbb{R}_+^m$, where the coordinate t_{ij} represents her value for good j. Let $\mathcal{T} \coloneqq \times_{i\in[n]}\mathcal{T}_i$ denote the set of type profiles. We assume $\mathcal{T}$ is compact.

The auctioneer implements a *direct mechanism*: given a reported type profile $\boldsymbol{t}' = \{t_i'\}_{i\in[n]} \in \mathcal{T}$, it returns (i) an outcome $\boldsymbol{x} \in \mathcal{X}$ and (ii) a monetary transfer vector $\boldsymbol{p} = (p_1, \ldots, p_n) \in \mathbb{R}^n$. The feasible outcome space is $\mathcal{X} \coloneqq (\Delta_n)^m$, where $\Delta_n \coloneqq \{y \in \mathbb{R}_+^n : \sum_{i\in[n]} y_i \le 1\}$ is the n-simplex that represents feasible probabilistic allocations of 1 good among n buyers. $\boldsymbol{x}$'s i,j-th element, x_{ij}, represents the probability that good j is allocated to buyer i, and p_i represents the monetary transfer from buyer i to the auctioneer. We write $x_i = (x_{i1}, \ldots, x_{im})$ for buyer i's allocation. Buyer i has an additive, quasi-linear utility that relies on her allocation x_i, monetary transfer p_i, and type t_i, *i.e.*, $u_i(x_i, p_i; t_i) = \langle x_i, t_i\rangle - p_i$.

The auctioneer is revenue-maximizing, *i.e.*, her utility is $\sum_i p_i$, the auction's revenue. Because types are buyers' private information, the auctioneer only knows a prior distribution $\mathcal{F} \in \Delta(\mathcal{T})$ over type profiles and seeks to maximize her expected revenue under $\mathcal{F}$. Overall, an auction instance is specified by $\mathcal{A} = (n, m, \mathcal{F}, \mathcal{T})$, and the auctioneer seeks a truthful (DSIC or strategyproof) and individually rational (IR) direct mechanism that maximizes expected revenue.

Definition 2.1 (Direct Mechanisms)**.** A direct mechanism $M^{\mathrm{DIR}} = \left(\boldsymbol{x}^{\mathrm{DIR}}, \boldsymbol{p}^{\mathrm{DIR}}\right)$ consists of an allocation rule $\boldsymbol{x}^{\mathrm{DIR}} : \mathcal{T} \to \mathcal{X}$ and a payment rule $\boldsymbol{p}^{\mathrm{DIR}} : \mathcal{T} \to \mathbb{R}^n$.[2] Given a reported type profile $\boldsymbol{t}'$, a direct mechanism M^{DIR} implements the outcome $\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t}') \in \mathcal{X}$ and monetary transfer $\boldsymbol{p}^{\mathrm{DIR}}(\boldsymbol{t}') \in \mathbb{R}^n$. For each buyer i with true type t_i, she realizes her allocation $x_i^{\mathrm{DIR}}(\boldsymbol{t}')$ and the monetary transfer $p_i^{\mathrm{DIR}}(\boldsymbol{t}')$, thus yielding utility $u_i(x_i^{\mathrm{DIR}}(\boldsymbol{t}'), p_i^{\mathrm{DIR}}(\boldsymbol{t}'); t_i)$.

Definition 2.2 (Truthful and IR Direct Mechanisms)**.** A direct mechanism $M^{\mathrm{DIR}} = (\boldsymbol{x}^{\mathrm{DIR}}, \boldsymbol{p}^{\mathrm{DIR}})$ is truthful if reporting the true type is a dominant strategy for each buyer i, regardless of the reported types $\boldsymbol{t}'_{-i}$ of all other buyers. M^{DIR} is IR if each buyer i weakly prefers participating in the auction to nonparticipation ($x_i = \mathbf{0}, p_i = 0$, representing no allocation and no payment) in every case. The properties of truthfulness and IR can be expressed by the following two conditions for all $i \in [n]$, buyer i's type $t_i \in \mathcal{T}_i$, and buyer $-i$'s reported types $\boldsymbol{t}'_{-i} \in \mathcal{T}_{-i}$:

$$u_i(x_i^{\mathrm{DIR}}(t_i, \boldsymbol{t}'_{-i}), p_i^{\mathrm{DIR}}(t_i, \boldsymbol{t}'_{-i}); t_i) \ge 0, \tag{IR}$$

$$u_i(x_i^{\mathrm{DIR}}(t_i, \boldsymbol{t}'_{-i}), p_i^{\mathrm{DIR}}(t_i, \boldsymbol{t}'_{-i}); t_i) \ge u_i(x_i^{\mathrm{DIR}}(t'_i, \boldsymbol{t}'_{-i}), p_i^{\mathrm{DIR}}(t'_i, \boldsymbol{t}'_{-i}); t_i). \qquad \forall t'_i \in \mathcal{T}_i \tag{IC}$$

From now on, we refer to truthful and IR direct mechanisms simply as truthful mechanisms when the context is clear. Denote $\mathcal{M}^{\mathrm{DIR}}$ as the set of all direct mechanisms. The auctioneer's goal for an auction problem $\mathcal{A}$ can be stated as solving Program (1).

$$\max_{(\boldsymbol{x}^{\mathrm{DIR}}, \boldsymbol{p}^{\mathrm{DIR}}) = M^{\mathrm{DIR}} \in \mathcal{M}^{\mathrm{DIR}}} \ \mathbb{E}_{\boldsymbol{t}\sim\mathcal{F}}\Big[\sum_{i\in[n]} p_i^{\mathrm{DIR}}(\boldsymbol{t})\Big] \qquad \text{s.t. (IC), (IR).} \tag{1}$$

Affine Maximizer Auctions (AMA) An Affine Maximizer Auction is represented by $M^{\mathrm{AMA}} = \{(\boldsymbol{x}^\kappa, \lambda^\kappa)\}_{\kappa\in[L]}$ where $L \in \mathbb{N}$, $\boldsymbol{x}^\kappa \in \mathcal{X}$, and $\lambda^\kappa \in \mathbb{R}$ for all $\kappa \in [L]$. Here, $\boldsymbol{x}^\kappa$ denotes a candidate outcome and λ^κ denotes the boosting value *w.r.t.* $\boldsymbol{x}^\kappa$.

[1] Further related works are deferred to Appendix A.

[2] The superscript "DIR" abbreviates "direct mechanism".

An AMA M^{AMA} can behave like a direct mechanism, *i.e.*, given $\boldsymbol{t} \in \mathcal{T}$ as input, M^{AMA} can output an outcome $\boldsymbol{x} \in \mathcal{X}$ and a monetary transfer $\boldsymbol{p} \in \mathbb{R}^n$. Specifically, an AMA $M^{\text{AMA}} = \{(\boldsymbol{x}^\kappa, \lambda^\kappa)\}_{\kappa \in [L]}$ can induce direct mechanisms $M^{\text{DIR}} = (\boldsymbol{x}^{\text{DIR}}, \boldsymbol{p}^{\text{DIR}})$ that compute the outcome $\boldsymbol{x}^{\text{DIR}}(\boldsymbol{t})$ and monetary transfer $\boldsymbol{p}^{\text{DIR}}(\boldsymbol{t})$ through the following steps given $\boldsymbol{t} \in \mathcal{T}$ as input:

$$\begin{aligned}
&\text{ASW}(\kappa; \boldsymbol{t}) \coloneqq \sum_i \langle t_i, x_i^\kappa \rangle + \lambda^\kappa, && \text{ASW}_{-i}(\kappa; \boldsymbol{t}) \coloneqq \sum_{j \neq i} \langle t_j, x_j^\kappa \rangle + \lambda^\kappa, \\
&\kappa^*(\boldsymbol{t}) \in \text{argmax}_{\kappa \in [L]} \text{ASW}(\kappa; \boldsymbol{t}), && \kappa^{-i,*}(\boldsymbol{t}) \in \text{argmax}_{\kappa \in [L]} \text{ASW}_{-i}(\kappa; \boldsymbol{t}), \\
&\boldsymbol{x}^{\text{DIR}}(\boldsymbol{t}) = \boldsymbol{x}^{\kappa^*(\boldsymbol{t})}, && p_i^{\text{DIR}}(\boldsymbol{t}) = \text{ASW}_{-i}(\kappa^{-i,*}(\boldsymbol{t}); \boldsymbol{t}) - \text{ASW}_{-i}(\kappa^*(\boldsymbol{t}); \boldsymbol{t}).
\end{aligned} \tag{2}$$

where $\text{ASW}(\kappa; \boldsymbol{t})$ denotes the affine social welfare of outcome $\boldsymbol{x}^\kappa$. Note that the induced direct mechanisms M^{DIR} for a given AMA M^{AMA} may not be unique due to the non-uniqueness of argmax. A well-known fact about AMAs is that if M^{DIR} is induced by M^{AMA}, then M^{DIR} is truthful, as shown in the following proposition.

Proposition 2.3. (Truthfulness of AMA). *Let M^{AMA} be an AMA mechanism and M^{DIR} be a direct mechanism induced by M^{AMA}, then M^{DIR} is truthful and IR.*

The proof of Proposition 2.3 (as well as other theorems) is presented in Appendix B.

Remark 2.4. *It is important to note that a more general form of AMA has an additional parameter set $\{w_i\}_{i \in [n]}, w_i > 0$, where w_i represents the weight of buyer i. The affine social welfare is then redefined as $\text{ASW}(\kappa; \boldsymbol{t}) \coloneqq \sum_{i \in [n]} w_i \cdot \langle t_i, x_i^\kappa \rangle + \lambda^\kappa$ and $p_i^{\text{DIR}}(\boldsymbol{t}) = w_i^{-1} \left(\text{ASW}_{-i}(\kappa^{-i,*}(\boldsymbol{t}); \boldsymbol{t}) - \text{ASW}_{-i}(\kappa^*(\boldsymbol{t}); \boldsymbol{t}) \right)$.*
In this work, we fix $w_i \equiv 1$ for two reasons: (a) learning additional weights appears orthogonal to our main contribution; (b) when buyers are symmetric (as in our experiments), symmetric weights provide a natural inductive bias. Learnable w_is can be incorporated in a straightforward way.

3 Neural Affine Maximizer

This section introduces our novel concept of Neural Affine Maximizers (NAMs), which define AMAs through a continuous boosting function rather than outcome discretization as in Equation (2).

3.1 Definition

An NAM $M^{\text{NAM}} = \{\lambda^{\text{NAM}}\}$ is represented by a continuous boosting function $\lambda^{\text{NAM}} : \mathcal{X} \to \mathbb{R}$. Similar to an AMA, M^{NAM} can induce direct mechanisms $M^{\text{DIR}} = (\boldsymbol{x}^{\text{DIR}}, \boldsymbol{p}^{\text{DIR}})$ that compute the outcome $\boldsymbol{x}^{\text{DIR}}(\boldsymbol{t})$ and monetary transfer $\boldsymbol{p}^{\text{DIR}}(\boldsymbol{t})$ via the following steps for a given type profile $\boldsymbol{t}$:

$$\begin{aligned}
&\boldsymbol{t}[-i] \coloneqq (t_1, ..., t_{i-1}, \mathbf{0}_m, t_{i+1}, ..., t_n), && \text{ASW}(\boldsymbol{x}; \boldsymbol{t}) \coloneqq \sum_{i \in [n]} \langle t_i, x_i \rangle + \lambda^{\text{NAM}}(\boldsymbol{x}), \\
&\boldsymbol{x}^{\text{DIR}}(\boldsymbol{t}) \in \text{argmax}_{\boldsymbol{x} \in \mathcal{X}} \text{ASW}(\boldsymbol{x}; \boldsymbol{t}), && \boldsymbol{x}^{-i}(\boldsymbol{t}) \in \text{argmax}_{\boldsymbol{x} \in \mathcal{X}} \text{ASW}(\boldsymbol{x}; \boldsymbol{t}[-i]), \\
&p_i^{\text{DIR}}(\boldsymbol{t}) = \text{ASW}(\boldsymbol{x}^{-i}(\boldsymbol{t}); \boldsymbol{t}[-i]) - \text{ASW}(\boldsymbol{x}^{\text{DIR}}(\boldsymbol{t}); \boldsymbol{t}[-i]).
\end{aligned} \tag{3}$$

where $\mathbf{0}_m$ represents an m-dimensional zero vector. Note that each argmax is well defined because $\lambda^{\text{NAM}}(\boldsymbol{x})$ (hence $\text{ASW}(\boldsymbol{x}; \boldsymbol{t})$) is continuous and $\mathcal{X}$ is compact. Similar to AMAs, the induced direct mechanisms for an NAM may not be unique due to the non-uniqueness of argmax.

3.2 Connection between NAM and AMA

Proposition 3.1 establishes the truthfulness of NAM.

Proposition 3.1. (Truthfulness of NAM) *Let M^{NAM} be an NAM and M^{DIR} be a direct mechanism induced by M^{NAM}. Then M^{DIR} is truthful and IR.*

Proposition 3.1 follows readily if one views each NAM as an AMA with a continuum of candidate outcomes. Next, we establish the equivalence between NAMs and AMAs in Theorem 3.2.

Theorem 3.2. *Fix $M^{AMA} = \{(\boldsymbol{x}^\kappa, \lambda^\kappa)\}_{\kappa\in[L]}$ to be an arbitrary AMA mechanism (with weights $w_i \equiv 1$). Then, there exists an NAM $M^{NAM} = \{\lambda^{NAM}\}$ such that:*

(a) λ^{NAM} is concave on $\mathcal{X}$.

(b) If direct mechanism M^{DIR} is induced by M^{AMA}, then M^{DIR} is also induced by M^{NAM}.

Theorem 3.2 theoretically guarantees that, compared with AMAs, NAMs never lose expressive power, even when λ^{NAM} is restricted to be concave.

4 Learning Neural Affine Maximizers

This section introduces how we learn a Neural Affine Maximizer for a given auction model $\mathcal{A}$. At a high level, we first parameterize an NAM with a neural network that is a universal approximator of convex functions. The network parameters are then trained via first-order optimization.

4.1 Parameterization of Neural Affine Maximizers

Theorem 3.2 motivates us to utilize a universal approximator of convex functions to parameterize λ^{NAM}, in order to ensure that the neural network can represent the revenue-maximizing auction within AMA class. For this purpose, we use Input Convex Neural Network (ICNN, [2]), a universal approximator of convex functions, to parameterize λ^{NAM}. The architecture and the computation flow of a K-layer ICNN are visualized in Figure 1.

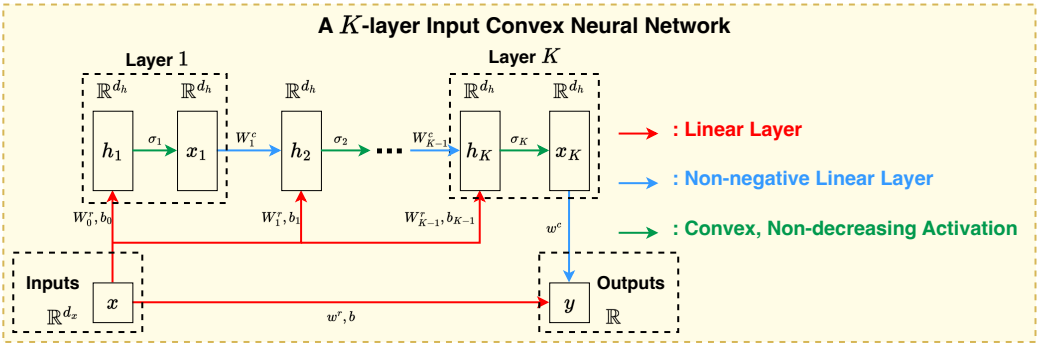


Figure 1: The architecture of an ICNN. Best viewed in color (same for the remaining figures).

Definition 4.1. A K-layer ICNN parameterizes a convex function $f : \mathbb{R}^{d_x} \to \mathbb{R}$ with parameters: $\{W_k^c\}_{1\le k\le K-1}$, $\{W_k^r\}_{0\le k\le K-1}$, $\{b_k\}_{0\le k\le K-1}$, $w^c \in \mathbb{R}_+^{d_h}$, $w^r \in \mathbb{R}^{d_x}$, $b \in \mathbb{R}$, and activation functions $\{\sigma_k\}_{k\in[K]}$, where d_h is the dimension of hidden layer, $W_k^c \in \mathbb{R}_+^{d_h\times d_h}$, $W_k^r \in \mathbb{R}^{d_h\times d_x}$, $b_k \in \mathbb{R}^{d_h}$, and $\sigma_k : \mathbb{R} \to \mathbb{R}$ is a convex, continuous and non-decreasing activation function (*e.g.*, leaky ReLU). In Figure 1, h_k and x_k denote the pre- and post-activation values of the k-th layer, respectively. For space limits, we defer the pseudocode of ICNN computation to Appendix B.6.

Remark 4.2. *One may consider using MLP (Multi-Layer Perceptron) as an alternative to ICNN. The key reason for using an ICNN is to ensure the concavity of λ^{NAM} and hence of* ASW*, which guarantees that the* argmax *values of* ASW *can be solved efficiently by convex optimization. Without such a concavity guarantee, it is unclear how to compute these* argmax *values efficiently.*

Since some ICNN parameters are required to be non-negative, we reparameterize the weights of the ICNN through a softplus function ($\mathrm{softplus}(x) = \log(1 + \exp(x))$) that maps a real number to a positive number. Let $f(\boldsymbol{x};\theta) : \mathcal{X} \to \mathbb{R}$ be the ICNN where θ denotes the network parameter, we let the boosting function $\lambda^{\mathrm{NAM}}(\boldsymbol{x};\theta) = -f(\boldsymbol{x};\theta)$ to guarantee the concavity of $\lambda^{\mathrm{NAM}}(\boldsymbol{x};\theta)$ *w.r.t.* $\boldsymbol{x}$. We next state the universal approximation of ICNNs that is originally proved by Chen et al. [7], and provide a self-contained proof in Appendix B.4.

Definition 4.3 (Universal Approximator)**.** Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be two classes of functions. We say that $\mathcal{F}_1$ is a universal approximator of $\mathcal{F}_2$ if:

1. $\mathcal{F}_1 \subseteq \mathcal{F}_2$; and
2. for every $f_2 \in \mathcal{F}_2$ and every $\varepsilon > 0$, there exists $f_1 \in \mathcal{F}_1$ such that $l_\infty(f_1, f_2) < \varepsilon$.

Theorem 4.4. *Let $\mathcal{Y} \subseteq \mathbb{R}^{d_x}$ be a convex, compact subset of d_x-dimensional Euclidean space. Let $\mathcal{G} \coloneqq \{f : \mathcal{Y} \to \mathbb{R} | f$ is an ICNN$\}$ and $\mathcal{H} \coloneqq \{f : \mathcal{Y} \to \mathbb{R} | f$ is convex and continuous$\}$. Then, $\mathcal{G}$ is a universal approximator of $\mathcal{H}$.*

Together with Theorem 3.2, Theorem 4.4 supports the expressiveness of NAM: by using ICNN as the network architecture for λ^{NAM}, NAM can approximate an optimal (unknown) AMA to arbitrary precision.

4.2 Training of Neural Affine Maximizers

We now describe how the parameters of a Neural Affine Maximizer are learned via first-order optimization. The pseudocode is left to Appendix B.7 for space limits.

To begin with, we express the auctioneer's revenue maximization as a constrained optimization problem. Throughout, θ denotes the parameters of the ICNN, $\lambda^{\text{NAM}}(\boldsymbol{x}; \theta)$ the parameterized boosting function, and $\text{ASW}(\boldsymbol{x}; \boldsymbol{t}, \theta) \coloneqq \sum_{i\in[n]} \langle x_i, t_i \rangle + \lambda^{\text{NAM}}(\boldsymbol{x}; \theta)$ the parameterized affine social welfare. According to Section 3.1, every outcome computed by the mechanism—both $\boldsymbol{x}^{\text{DIR}}(\boldsymbol{t})$ and $\boldsymbol{x}^{-i}(\boldsymbol{t})$—must maximize the affine social welfare:

$$\boldsymbol{x}^{-i}(\boldsymbol{t}) \in \text{argmax}_{\boldsymbol{a}\in\mathcal{X}} \text{ASW}(\boldsymbol{a}; \boldsymbol{t}[-i], \theta),\ \forall i \in [n],\ \ \boldsymbol{x}^{\text{DIR}}(\boldsymbol{t}) \in \text{argmax}_{\boldsymbol{a}\in\mathcal{X}} \text{ASW}(\boldsymbol{a}; \boldsymbol{t}, \theta). \tag{4}$$

To write these $n+1$ constraints compactly, we slightly abuse our notations to define $\boldsymbol{t}[-(n+1)] \coloneqq \boldsymbol{t}$ and $\boldsymbol{x}^{-(n+1)}(\boldsymbol{t}) \coloneqq \boldsymbol{x}^{\text{DIR}}(\boldsymbol{t})$. Under this notation, Equation (4) reduces to:

$$\boldsymbol{x}^{-i}(\boldsymbol{t}) \in \text{argmax}_{\boldsymbol{a}\in\mathcal{X}} \text{ASW}(\boldsymbol{a}; \boldsymbol{t}[-i], \theta), \qquad \forall i \in [n+1]. \tag{5}$$

Note that the payment rule $p_i^{\text{DIR}}(\boldsymbol{t}) \coloneqq \text{ASW}(\boldsymbol{x}^{-i}(\boldsymbol{t}); \boldsymbol{t}[-i], \theta) - \text{ASW}(\boldsymbol{x}^{\text{DIR}}(\boldsymbol{t}); \boldsymbol{t}[-i], \theta)$ (Equation (3)) is fully determined by the outcomes $\{\boldsymbol{x}^{-i}(\boldsymbol{t})\}_{i\in[n+1]}$, the type profile $\boldsymbol{t}$, and the parameter θ. Therefore, the auctioneer's utility function—namely, the total payment $\sum_{i\in[n]} p_i^{\text{DIR}}(\boldsymbol{t})$—can be abbreviated as $\sum_{i\in[n]} p_i^{\text{DIR}}(\boldsymbol{t}) \triangleq u_0(\{\boldsymbol{x}^{-i}(\boldsymbol{t})\}_{i\in[n+1]}, \theta; \boldsymbol{t})$, dropping the explicit reference to $\boldsymbol{p}$. The original learning objective (Program 1) is then captured by the following constrained optimization:

$$\max_{\{\boldsymbol{x}^{-i}(\boldsymbol{t})\}_i, \theta} \ \mathbb{E}_{\boldsymbol{t}\sim\mathcal{F}}[u_0(\{\boldsymbol{x}^{-i}(\boldsymbol{t})\}_i, \theta; \boldsymbol{t})] \text{ s.t. } \boldsymbol{x}^{-i}(\boldsymbol{t}) \in \text{argmax}_{\boldsymbol{a}\in\mathcal{X}} \text{ASW}(\boldsymbol{a}; \boldsymbol{t}[-i], \theta),\ \forall i \in [n+1], \boldsymbol{t} \in \mathcal{T}. \tag{6}$$

The remainder of Section 4.2 develops a tractable, first-order algorithm for solving Program (6).

Step 1: Sample Averaging. Because the expectation over $\mathcal{F}$ is generally intractable, we draw B i.i.d. type profiles $\mathcal{T}^B = \{\boldsymbol{t}^1, ..., \boldsymbol{t}^B\}$ from $\mathcal{F}$ and replace the expectation with a sample average, yielding:

$$\max_{\{\boldsymbol{x}^{-i,b}\}, \theta} \quad \frac{1}{B} \sum_{b\in[B]} u_0(\{\boldsymbol{x}^{-i,b}\}_i, \theta; \boldsymbol{t}^b) \quad \text{s.t.} \quad \boldsymbol{x}^{-i,b} \in \text{argmax}_{\boldsymbol{a}\in\mathcal{X}} \text{ASW}(\boldsymbol{a}; \boldsymbol{t}^b[-i], \theta), \quad \forall b \in [B], i \in [n+1]. \tag{7}$$

Step 2: Softening the Constraints. The argmax operations in Program (7) are generally non-differentiable, precluding direct gradient-based optimization. We address this by relaxing each hard argmax with a Gibbs distribution that concentrates around the maximizer. Concretely, define

$$q^\beta(\boldsymbol{a}|\boldsymbol{t}, \theta) = \frac{\exp(\beta \cdot \text{ASW}(\boldsymbol{a}; \boldsymbol{t}, \theta))}{Z^\beta(\boldsymbol{t}, \theta)}, \quad Z^\beta(\boldsymbol{t}, \theta) = \int_{\boldsymbol{a}\in\mathcal{X}} \exp(\beta \cdot \text{ASW}(\boldsymbol{a}; \boldsymbol{t}, \theta)) \mathrm{d}\boldsymbol{a}. \tag{8}$$

where $\beta > 0$ is a temperature hyperparameter controlling how sharply q^β is centered at the maximizer of ASW, and $Z^\beta(\boldsymbol{t}, \theta)$ is the normalizing constant. Substituting this relaxation into Program (7) removes the argmax constraints and yields an unconstrained objective:

$$\max_\theta \qquad \text{OBJ}^\beta(\theta) = \frac{1}{B} \sum_{b\in[B]} \mathbb{E}_{\boldsymbol{x}^{-i,b} \overset{\text{i.d.}}{\sim} q^\beta(\cdot|\boldsymbol{t}^b[-i], \theta)} [u_0(\{\boldsymbol{x}^{-i,b}\}_i, \theta; \boldsymbol{t}^b)]. \tag{9}$$

where $\overset{\text{i.d.}}{\sim}$ denotes that the samples are independently distributed.

Step 3: Covariance Trick. Following conventions in neural network optimization [1] and differentiable economics [19], our next step is to compute an unbiased estimator of $\nabla_\theta \mathrm{OBJ}^\beta(\theta)$, namely the first-order gradient of the objective. Given this estimator, θ can be automatically optimized through first-order algorithms (*e.g.*, Adam) in libraries like Pytorch.

A difficulty here is that the differentiation operator ∇_θ and the expectation operator $\mathbb{E}[\cdot]$ cannot be interchanged directly, as $\mathbb{E}_{\boldsymbol{x}^{-i,b} \overset{\text{i.d.}}{\sim} q^\beta(\cdot|\boldsymbol{t}^b[-i],\theta)}$ explicitly depends on θ. This makes the gradient computation of Equation (9) non-trivial. Ma et al. [33] proposes "covariance trick" to resolve this difficulty. Inspired by Ma et al. [33], we derive the "covariance trick" for NAM in Proposition 4.5.

Proposition 4.5. *The following identity holds:*

$$\nabla_\theta \mathbb{E}_{\boldsymbol{x}^{-i}}\Big[u_0(\{\boldsymbol{x}^{-i}\}_i,\theta;\boldsymbol{t})\Big] = \mathbb{E}_{\boldsymbol{x}^{-i}}\Big[\nabla_\theta u_0(\{\boldsymbol{x}^{-i}\}_i,\theta;\boldsymbol{t})\Big] + \beta\cdot \mathrm{Cov}_{\boldsymbol{x}^{-i}}\Big[u_0(\{\boldsymbol{x}^{-i}\}_i,\theta;\boldsymbol{t}), \nabla_\theta \sum_{i\in[n+1]} \mathrm{ASW}(\boldsymbol{x}^{-i};\boldsymbol{t}[-i],\theta)\Big]$$

where $\mathbb{E}_{\boldsymbol{x}^{-i}}$ *and* $\mathrm{Cov}_{\boldsymbol{x}^{-i}}$ *mean that* $\boldsymbol{x}^{-i} \overset{\text{i.d.}}{\sim} q^\beta(\cdot|\boldsymbol{t}[-i],\theta)$ *for* $i \in [n+1]$.

Note that the left-hand side of Proposition 4.5 is the exact gradient of Equation (9) but is not directly computable. Meanwhile, the right-hand side admits an unbiased Monte Carlo estimator given independent draws from $q^\beta(\cdot|\boldsymbol{t}[-i],\theta)$ for each $i \in [n+1]$. Based on Proposition 4.5, we construct the gradient surrogate below:

$$\begin{aligned}\mathcal{J}^\beta(\theta;\mathcal{T}^B) =& \frac{1}{2B}\sum_{b\in[B]}\Big[\big(u_0(\{\boldsymbol{y}^{-i,b}\}_i,\theta;\boldsymbol{t}^b) + u_0(\{\boldsymbol{z}^{-i,b}\}_i,\theta;\boldsymbol{t}^b)\big) + \beta\cdot sg\big(u_0(\{\boldsymbol{y}^{-i,b}\}_i,\theta;\boldsymbol{t}^b)\\ & - u_0(\{\boldsymbol{z}^{-i,b}\}_i,\theta;\boldsymbol{t}^b)\big)\cdot \sum_{i\in[n+1]}\big(\mathrm{ASW}(\boldsymbol{y}^{-i,b};\boldsymbol{t}^b[-i],\theta) - \mathrm{ASW}(\boldsymbol{z}^{-i,b};\boldsymbol{t}^b[-i],\theta)\big)\Big].\end{aligned} \tag{10}$$

Here $sg(\cdot)$ is the stop-gradient operator, defined by $\nabla_\theta(sg(f(\theta))\cdot g(\theta)) = f(\theta)\cdot\nabla_\theta g(\theta)$ that eliminates the gradient propagation through $f(\theta)$. It is implementable in libraries like PyTorch. If we can generate $\boldsymbol{y}^{-i,b}, \boldsymbol{z}^{-i,b} \overset{\text{i.d.}}{\sim} q^\beta(\cdot|\boldsymbol{t}^b[-i],\theta)$, then $\nabla_\theta\mathcal{J}^\beta(\theta;\mathcal{T}^B)$ is an unbiased estimator of $\nabla_\theta\mathrm{OBJ}^\beta(\theta)$. The next step explains how to generate $\boldsymbol{y}^{-i,b}, \boldsymbol{z}^{-i,b} \overset{\text{i.d.}}{\sim} q^\beta(\cdot|\boldsymbol{t}^b[-i],\theta)$ efficiently during the full training process.

Step 4: Sampling within $\mathcal{X}$. We focus on how to sample $\boldsymbol{y}^{-i,b} \sim q^\beta(\cdot|\boldsymbol{t}^b[-i],\theta)$ on the polyhedron $\mathcal{X}$ efficiently for each $b \in [B], i \in [n+1]$. The treatment of $\boldsymbol{z}^{-i,b}$ is similar.

We will construct a discrete-time Langevin dynamics [47] that converges to the stable distribution $q^\beta(\cdot|\boldsymbol{t}^b[-i],\theta)$. Denote $\hat\theta$ and θ as the last-iteration and current network parameters. Assume that we already have a sample $\hat{\boldsymbol{y}} \in \mathcal{X}$ drawn from $q^\beta(\cdot;\boldsymbol{t}^b[-i],\hat\theta)$. Fix the iteration step T, we sequentially construct $\boldsymbol{y}^1, ..., \boldsymbol{y}^T$ with initial point $\boldsymbol{y}^0 = \hat{\boldsymbol{y}}$ as follows:

$$\boldsymbol{y}^\tau = \boldsymbol{y}^{\tau-1} + \eta\nabla_{\boldsymbol{a}}\mathrm{ASW}(\boldsymbol{a};\boldsymbol{t}^b[-i],\theta)|_{\boldsymbol{a}=\boldsymbol{y}^{\tau-1}} + \sqrt{\frac{2\eta}{\beta}}\epsilon^\tau,\ \epsilon^\tau \overset{\text{i.i.d.}}{\sim}\mathcal{N}(\mathbf{0},\mathrm{Id}),\ \tau = 1,...,T \tag{11}$$

where $\eta > 0$ is the step size and Id is the identity matrix.

Note that $\boldsymbol{y}^\tau$ may leave $\mathcal{X}$. We therefore apply Skorokhod reflection [51] after each iteration so that $\boldsymbol{y}^\tau \in \mathcal{X}$. Specifically, let $\alpha \leftarrow \boldsymbol{y}^{\tau-1}$ be the initial point, $r \leftarrow \boldsymbol{y}^\tau - \boldsymbol{y}^{\tau-1}$ be the update direction, and assume that α stays within $\mathcal{X}$. We accept $\boldsymbol{y}^\tau \leftarrow \alpha + r$ iff $\alpha + r$ stays within $\mathcal{X}$. Let $\cap_{1\le s\le S}\{\gamma \in \mathbb{R}^{d_x} | \langle n_s,\gamma\rangle \ge c_s\}$ denote the half-space representation of the polyhedron $\mathcal{X}$, where n_s is a normal vector such that $\|n_s\| = 1$, $c_s \in \mathbb{R}$, and $d_x = nm$ is the dimension of $\mathcal{X}$. Let $\alpha(\ell) = \alpha + \ell r$ for $\ell \in [0,1]$ trace the segment from α toward $\alpha + r$, and let $t_0 = \min_{\{s|\langle n_s,r\rangle<0\}}\{\frac{c_s - \langle n_s,\alpha\rangle}{\langle n_s,r\rangle}\}$ be the first hitting time of the boundary, with minimizer s^*. We then update $\alpha \leftarrow \alpha + t_0 r$, rescale $r \leftarrow (1-t_0)r$, and reflect $r \leftarrow r - 2\langle n_{s^*}, r\rangle n_{s^*}$. These updates reflect one iteration $\alpha \to \alpha + r$ *w.r.t.* the s^*th half-space. We repeatedly update (α, r) until $\alpha + r$ stays within $\mathcal{X}$ and we accept $\boldsymbol{y}^\tau \leftarrow \alpha + r$. The reflection in a polyhedron provably terminates in finitely many steps [4].

Remark 4.6 (Bias-Compute Trade-off). *With a finite number of discrete Langevin steps T, the samples $\boldsymbol{y}^{-i,b}, \boldsymbol{z}^{-i,b}$ follow $q^\beta(\cdot|\boldsymbol{t}^b[-i],\theta)$ only approximately, which introduces a bias into the*

gradient estimator $\nabla_\theta \mathcal{J}^\beta$. On the other hand, this bias can be made arbitrarily small by reducing the step size η and increasing the number of iterations T [47], at the cost of additional compute. In practice, this represents a trade-off between the bias and the compute cost.

4.3 Inference of Neural Affine Maximizers

Given a trained NAM with boosting function $\lambda^{\text{NAM}}(\boldsymbol{x}; \theta)$, the inference is to take each type profile $\boldsymbol{t}$ as input and compute $(\boldsymbol{x}, \boldsymbol{p})$ via Equation (3). Most steps can be computed naturally. The only subtlety is computing the argmax points of $\text{ASW}(\boldsymbol{x}; \boldsymbol{t}, \theta)$. In our approach, $\text{ASW}(\boldsymbol{x}; \boldsymbol{t}, \theta)$ is guaranteed to be concave due to the architectural convexity of ICNN. Therefore, with convex optimization tools [52], these argmax points can be efficiently computed in polynomial time. The implementation details about the inference are discussed in Appendix C.1.

5 Experiments

This section reports experimental results to demonstrate the effectiveness of NAM. Throughout the experiments, $\mathcal{T}_i = [0, 1]^m$, so an auction instance is represented by $\mathcal{A} = (n, m, \mathcal{F})$. Every single experiment uses one GPU card.

5.1 Baselines and Settings

We consider the following state-of-the-art approaches that learn truthful auctions as baselines:

- LotteryAMA [10]: Parameterizing an AMA mechanism as in Equation (2). The parameters are learned through first-order algorithms.
- AMenuNet [17]: Similar to LotteryAMA, but using a transformer architecture to overparameterize the AMA.
- OD-VVCA [18]: Enumerating boosts for all deterministic VVCA allocations as trainable parameters, optimized via zeroth-order methods.
- GemNet [55]: Learning buyer-level menus, post-processed by a mixed-integer linear program to enforce feasible allocations.

We implement LotteryAMA following its description [10] and incorporate AMenuNet's and OD-VVCA's open-source implementations [17, 18] into our experiments. For GemNet, we report the values from its original paper [55] when possible as it's not open-sourced. We set the menu size of LotteryAMA and AMenuNet to $8 \cdot (n+m)^2$ for a comparable computational complexity with NAM. [3] We note that our choices of menu size are competitive with their original choices and, partially for this reason, our reported values for these baselines exceed their original reports.

We also compare three classical, widely used baselines [10, 19, 55], respectively:

- VCG [8, 25, 53]: The arguably most classical mechanism with broad applicability.
- Itemwise: Run the optimal auction on each good (Myerson's auction) separately.
- Bundle: Bundle all goods as one good and run optimal auction on this bundling.

We denote $\mathcal{F}_{n,m}$ as the setting with n buyers, m goods and type profile distribution $\mathcal{F}$. We incorporate the following choices of $\mathcal{F}$ in the main text:

- $\mathcal{F} = U$: $t_{ij} \overset{\text{i.i.d.}}{\sim} U([0, 1])$, where $U(X)$ represents the uniform distribution on X.
- $\mathcal{F} = \text{Beta}$: $t_{ij} \overset{\text{i.i.d.}}{\sim} \text{Beta}(\alpha = 0.5, \beta = 0.5)$. [4]
- $\mathcal{F} = I$: $t_{ij} \overset{\text{i.i.d.}}{\sim} \frac{3}{4}U([0, \frac{3}{8}]) + \frac{1}{4}U([\frac{3}{8}, 1])$. This is an irregular distribution, which is widely used as a benchmark [19, 55].

[3] See Appendix C.2 for more details about the complexity analysis.

[4] The density function of beta distribution is: $f(x) = \frac{1}{B(\alpha,\beta)} x^{\alpha-1} \cdot (1-x)^{\beta-1}$, where $B(\alpha, \beta)$ is a normalizing constant.

Table 1: Auctioneer's revenue with n buyers, m goods and i.i.d., $U([0,1])$ valuations. Values of GemNet are cited from its original paper [55].

Method & Settings	$U_{2,2}$	$U_{2,5}$	$U_{3,3}$	$U_{3,10}$	$U_{5,5}$	$U_{5,10}$	$U_{3,30}$	$U_{10,20}$	$U_{15,10}$
NAM	$\underline{0.873}_{\pm 0.002}$	$\underline{2.312}_{\pm 0.002}$	$\underline{1.660}_{\pm 0.001}$	$\mathbf{5.822}_{\pm 0.003}$	$\mathbf{3.449}_{\pm 0.002}$	$\mathbf{7.000}_{\pm 0.002}$	$\mathbf{18.142}_{\pm 0.004}$	$\mathbf{16.606}_{\pm 0.003}$	$\mathbf{8.780}_{\pm 0.001}$
LotteryAMA	$\underline{0.870}_{\pm 0.004}$	$\mathbf{2.324}_{\pm 0.003}$	$1.636_{\pm 0.007}$	$5.393_{\pm 0.017}$	$3.158_{\pm 0.010}$	$5.749_{\pm 0.037}$	$15.529_{\pm 0.087}$	$11.316_{\pm 0.034}$	$5.938_{\pm 0.030}$
AMenuNet	$0.848_{\pm 0.031}$	$2.266_{\pm 0.006}$	$1.634_{\pm 0.007}$	$5.683_{\pm 0.006}$	$3.367_{\pm 0.013}$	$6.436_{\pm 0.043}$	$16.402_{\pm 0.024}$	$12.744_{\pm 0.028}$	$7.012_{\pm 0.007}$
OD-VVCA	$0.832_{\pm 0.030}$	$2.255_{\pm 0.007}$	$1.634_{\pm 0.004}$	$\mathbf{5.821}_{\pm 0.006}$	$\underline{3.433}_{\pm 0.004}$	$\underline{6.986}_{\pm 0.003}$	-	-	-
GemNet	**0.878**	$\underline{2.310}$	**1.675**	-	-	-	-	-	-
Itemwise	0.833	2.083	1.593	5.311	3.358	6.713	15.941	16.360	$\underline{8.750}$
Bundle	0.839	2.186	1.508	5.009	2.826	5.454	15.000	11.300	6.147
VCG	0.667	1.667	1.500	5.000	3.335	6.669	15.000	16.356	$\underline{8.750}$

Table 2: Auctioneer's revenue with n buyers, m goods and non-uniform distribution $\mathcal{F} \in \{\mathrm{Beta}, I\}$.

Method & Settings	$\mathrm{Beta}_{2,2}$	$\mathrm{Beta}_{3,10}$	$\mathrm{Beta}_{3,30}$	$\mathrm{Beta}_{15,10}$	$I_{2,2}$	$I_{3,10}$	$I_{3,30}$	$I_{15,10}$
NAM	$\mathbf{0.915}_{\pm 0.002}$	$\mathbf{6.146}_{\pm 0.002}$	$\mathbf{19.098}_{\pm 0.007}$	$\underline{9.479}_{\pm 0.002}$	$\mathbf{0.472}_{\pm 0.003}$	$\mathbf{3.648}_{\pm 0.003}$	$\mathbf{11.688}_{\pm 0.009}$	$\mathbf{7.070}_{\pm 0.001}$
LotteryAMA	$0.903_{\pm 0.021}$	$5.543_{\pm 0.049}$	$15.902_{\pm 0.027}$	$6.274_{\pm 0.028}$	$0.447_{\pm 0.023}$	$3.430_{\pm 0.024}$	$9.942_{\pm 0.022}$	$4.258_{\pm 0.005}$
AMenuNet	$0.888_{\pm 0.013}$	$5.852_{\pm 0.010}$	$16.756_{\pm 0.039}$	$7.474_{\pm 0.008}$	$\mathbf{0.467}_{\pm 0.016}$	$3.580_{\pm 0.004}$	$10.483_{\pm 0.009}$	$5.005_{\pm 0.004}$
OD-VVCA	$0.777_{\pm 0.022}$	$5.987_{\pm 0.007}$	-	-	$0.437_{\pm 0.002}$	$\underline{3.613}_{\pm 0.004}$	-	-
Itemwise	0.890	5.759	17.285	**9.482**	0.438	2.931	8.784	6.968
Bundle	0.842	5.002	15.004	6.406	0.465	3.103	9.347	4.158
VCG	0.593	5.000	15.002	$\underline{9.481}$	0.354	2.774	8.318	6.968

For space limits, we leave the other choices of $\mathcal{F}$, including a bernoulli distribution and a distribution that is correlated among buyers, as well as the corresponding results to Appendix C.3.

We use **bold** for the best value in each setting (with a tolerance of one standard deviation), underline for entries with $> 99\%$ of the best value, and '-' when a method is prohibitively expensive (> 12 hours) or the settings are not reported in the original paper. We report mean values and standard deviations in 5 independent runs.

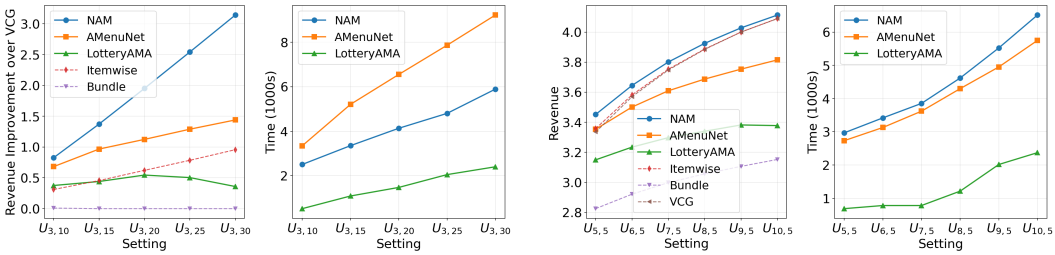
5.2 Experimental Results

Performance on Uniform Distribution. We first consider general $U_{n,m}$ settings up to 15 buyers or 30 goods, which have been widely studied [10, 17, 18, 55]. We note that the settings we consider are already large-scale, as most prior works consider settings with at most 5 buyers or 10 goods. Table 1 shows the auctioneer's expected revenue learned by different approaches on these settings. Notably, NAM achieves $> 99\%$ best revenues on all settings, demonstrating NAM's robustness and competitiveness on diverse scenarios. Additionally, in $U_{3,30}$, NAM learns a truthful auction with 10.6% revenue improvement and lower training time compared with the second-best approach (Figure 2(b)), demonstrating the advantage of NAM for large-scale scenarios. We notice that NAM also performs consistently across different random seeds.

Performance on Non-uniform Distributions. We also consider experiments on diverse distributions to verify the robustness of NAM. Table 2 presents the results on the Beta and irregular distributions. Notably, in both $\mathrm{Beta}_{n,m}$ and $I_{n,m}$ settings, NAM learns auctions that either outperform or are competitive with all baselines up to 15 buyers or 30 goods; for instance, NAM surpasses the second-best method by 10.5% in $\mathrm{Beta}_{3,30}$ and 11.5% in $I_{3,30}$.

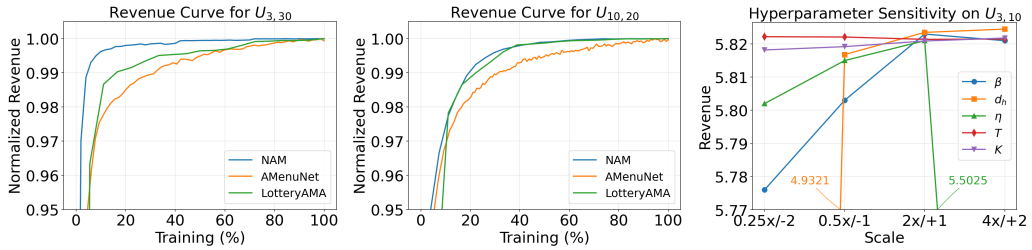
Scaling Performance. We vary the numbers of buyers and goods to evaluate the scalability of NAM. We visualize revenue and training time for NAM, LotteryAMA, and AMenuNet in Figure 2 and required memory for these approaches in Figure 4 in Appendix C.3.[5] Notably, as the number of buyers n grows, NAM continues to learn auctions that outperform VCG, while LotteryAMA and AMenuNet fail. As the number of goods m grows, the revenue gap between NAM and the second-best approach (AMenuNet) widens. Additionally, the training time and required memory of NAM grow mildly with n and m. We also visualize the revenue curve of these approaches in $U_{3,30}$ and $U_{10,20}$ as a function of training times (resp. TFLOPS) in Figure 3(a)(b) (resp. Figure 5 in Appendix C). The result shows that NAM converges smoothly and quickly.

[5]We skip OD-VVCA for this comparison as OD-VVCA has inevitable exponential complexity *w.r.t.* problem size [18].



(a) (b) (c) (d)

Figure 2: (a)(b): Revenue and Training Time comparison on $U_{3,m}$ settings with $m \in \{10, 15, 20, 25, 30\}$. The revenues are subtracted by VCG revenues for better visualization. (c)(d): Revenue and Training Time comparison on $U_{n,5}$ settings with $n \in \{5, 6, 7, 8, 9, 10\}$.



(a) (b) (c)

Figure 3: (a)(b): Revenue curves in the training process of NAM , Lottery-AMA and AMenuNet. Each revenue curve is normalized to reach a maximum value of 1 by dividing a factor; (c): Revenue in $U_{3,10}$ by ablating NAM hyperparameters from $0.25 \times (-2)$ to $4 \times (+2)$.

Hyperparameter Sensitivity. We perform one-dimensional ablations around default hyperparameters in $U_{3,10}$: temperature $\beta = 512$, Langevin iterations $T = 32$ and step size $\eta = 0.04$, as well as ICNN hidden width $d_h = 10 \cdot nm$ and depth $K = 3$. For each axis we try $\{\frac{1}{4}, \frac{1}{2}, 2, 4\}\times$ scaling for (β, T, η, d_h) and offsets $\{-2, -1, +1, +2\}$ on K. Results in Figure 3(c) show that revenues remain stable for almost all ablations, demonstrating the robustness of NAM over hyperparameter perturbations. The only exceptions are excessively narrow network widths and overly large Langevin step size. Similar failure modes have been discussed across deep learning areas [15, 30, 40].

6 Conclusions and Discussions

This paper proposes the Neural Affine Maximizer (NAM)—a discretization-free mechanism as an alternative to AMAs, as well as an algorithm to learn NAMs. Compared with existing approaches, NAM finds truthful auctions with competitive or larger revenues in all scenarios, and achieves a 10.6% revenue improvement in large-scale auctions with lower running times. Our work motivates many future works, *e.g.*, designing permutation-equivariant convex neural networks that could replace ICNNs and potentially accelerate training [27, 33, 43], and designing alternative algorithms for training NAMs with lower complexity *w.r.t.* n, m that might apply to larger-scale auctions.

NAM possesses some features that differ from existing approaches. First, training time scales linearly with the number of samples B, which induces a trade-off: a small B is prone to overfitting, while a large B makes training costly. We find empirically that $B = 2^{14}$ strikes a good balance. Second, NAM incurs inevitably larger inference time than baselines, as it requires solving convex programs to compute outcomes and monetary transfers. In our most inference-intensive setting, NAM takes 2 minutes to infer a batch of 2^{14} type profiles (as a comparison, AMenuNet uses 7 seconds). This would become a bottleneck in scenarios that require low latency (*e.g.*, ad auctions). On the other hand, we believe that our "boosting function" perspectives for AMAs provide useful insights on auction learning and differentiable economics. Solving convex programs during inference is an unavoidable side effect.

AI USE STATEMENT

In this work, we used generative AI tools to assist with polishing the writing and checking the grammar and proofs of the manuscript, to assist in writing and debugging experimental codes and scripts, to help identify relevant literature, and to provide revision suggestions on drafts of the manuscript. We did not use generative AI tools for conceiving research ideas or the methodology of this work, nor for producing the original content of the manuscript, the original proofs of the theoretical results, or the original experimental code—all of these were conceived, derived, and written solely by the authors.

We have reviewed all AI-assisted work: all AI-assisted text was checked and revised by the authors, all AI-assisted code was verified and tested by the authors, and all AI-suggested references were checked against their original publications. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The full source code for reproducing all experiments is provided in the supplemental material. Complete proofs of all theoretical results are provided in Appendix B, and implementation details are provided in Appendix C.

REFERENCES

[1] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

[2] Brandon Amos, Lei Xu, and J Zico Kolter. Input convex neural networks. In *International conference on machine learning*, pages 146–155. PMLR, 2017.

[3] Jerome Bracken and James T McGill. Mathematical programs with optimization problems in the constraints. *Operations research*, 21(1):37–44, 1973.

[4] Dmitry Burago, Sergei Ferleger, and Alexey Kononenko. Uniform estimates on the number of collisions in semi-dispersing billiards. *Annals of Mathematics*, 147(3):695–708, 1998.

[5] Giuseppe C Calafiore, Stephane Gaubert, and Corrado Possieri. Log-sum-exp neural networks and posynomial models for convex and log-log-convex data. *IEEE transactions on neural networks and learning systems*, 31(3):827–838, 2019.

[6] Lesi Chen, Yaohua Ma, and Jingzhao Zhang. Near-optimal nonconvex-strongly-convex bilevel optimization with fully first-order oracles. *Journal of Machine Learning Research*, 26(109): 1–56, 2025.

[7] Yize Chen, Yuanyuan Shi, and Baosen Zhang. Optimal control via neural networks: A convex approach. In *International Conference on Learning Representations*, 2019.

[8] Edward H Clarke. Multipart pricing of public goods. *Public choice*, 11:17–33, 1971.

[9] Michael Curry, Ping-Yeh Chiang, Tom Goldstein, and John Dickerson. Certifying strategyproof auction networks. *Advances in Neural Information Processing Systems*, 33:4987–4998, 2020.

[10] Michael Curry, Tuomas Sandholm, and John Dickerson. Differentiable economics for randomized affine maximizer auctions. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 2023.

[11] Michael J Curry, Uro Lyi, Tom Goldstein, and John P Dickerson. Learning revenue-maximizing auctions with differentiable matching. In *International Conference on Artificial Intelligence and Statistics*, pages 6062–6073. PMLR, 2022.

[12] Constantinos Daskalakis, Alan Deckelbaum, and Christos Tzamos. Strong duality for a multiple-good monopolist. volume 85, pages 735–767. Wiley Online Library, 2017.

[13] Stephan Dempe. *Foundations of bilevel programming*. Springer, 2002.

[14] Justin Domke. Generic methods for optimization-based modeling. In *Artificial intelligence and statistics*, pages 318–326. PMLR, 2012.

[15] Simon Du and Wei Hu. Width provably matters in optimization for deep linear neural networks. In *International Conference on Machine Learning*, pages 1655–1664. PMLR, 2019.

[16] Zhijian Duan, Jingwu Tang, Yutong Yin, Zhe Feng, Xiang Yan, Manzil Zaheer, and Xiaotie Deng. A context-integrated transformer-based neural network for auction design. In *International Conference on Machine Learning*, pages 5609–5626. PMLR, 2022.

[17] Zhijian Duan, Haoran Sun, Yurong Chen, and Xiaotie Deng. A scalable neural network for dsic affine maximizer auction design. *Advances in Neural Information Processing Systems*, 36:56169–56185, 2023.

[18] Zhijian Duan, Haoran Sun, Yichong Xia, Siqiang Wang, Zhilin Zhang, Chuan Yu, Jian Xu, and Xiaotie Deng. Automated deterministic auction design with objective decomposition. In *Proceedings of the ACM Web Conference 2026*, pages 225–236, 2026.

[19] Paul Dütting, Zhe Feng, Harikrishna Narasimhan, David C Parkes, and Sai Srivatsa Ravindranath. Optimal auctions through deep learning: Advances in differentiable economics. *Journal of the ACM*, 71(1):1–53, 2024.

[20] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[21] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International conference on machine learning*, pages 1568–1577. PMLR, 2018.

[22] Saeed Ghadimi and Mengdi Wang. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*, 2018.

[23] Yiannis Giannakopoulos and Elias Koutsoupias. Duality and optimality of auctions for uniform distributions. volume 47, pages 121–165. SIAM, 2018.

[24] Noah Golowich, Harikrishna Narasimhan, and David C Parkes. Deep learning for multi-facility location mechanism design. In *IJCAI*, pages 261–267, 2018.

[25] Theodore Groves. Incentives in teams. *Econometrica: Journal of the Econometric Society*, pages 617–631, 1973.

[26] Guillaume Haeringer. *Market design: auctions and matching*. MIT Press, 2018.

[27] Jiequn Han, Yingzhou Li, Lin Lin, Jianfeng Lu, Jiefu Zhang, and Linfeng Zhang. Universal approximation of symmetric and anti-symmetric functions. *Communications in Mathematical Sciences*, 20(5):1397–1408, 2022.

[28] Minhui Huang, Xuxing Chen, Kaiyi Ji, Shiqian Ma, and Lifeng Lai. Efficiently escaping saddle points in bilevel optimization. *Journal of machine learning research*, 26(1):1–61, 2025.

[29] Dmitry Ivanov, Iskander Safiulin, Igor Filippov, and Ksenia Balabaeva. Optimal-er auctions through attention. *Advances in Neural Information Processing Systems*, 35:34734–34747, 2022.

[30] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35: 26565–26577, 2022.

[31] Albert S Kyle. Informed speculation with imperfect competition. *The Review of Economic Studies*, 56(3):317–355, 1989.

[32] Yunxuan Ma, Yide Bian, Hao Xu, Weitao Yang, Jingshu Zhao, Zhijian Duan, Feng Wang, and Xiaotie Deng. Large-scale contextual market equilibrium computation through deep learning. In *International Workshop on Frontiers in Algorithmics*, pages 356–371. Springer, 2025.

[33] Yunxuan Ma, Steven (Siqiang) Wang, Zhijian Duan, Yukun Cheng, and Xiaotie Deng. Learning truthful mechanisms without discretization. 2026.

[34] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pages 2113–2122. PMLR, 2015.

[35] Alejandro M Manelli and Daniel R Vincent. Bundling as an optimal selling mechanism for a multiple-good monopolist. *Journal of Economic Theory*, 127(1):1–35, 2006.

[36] Peter McMullen. The maximum numbers of faces of a convex polytope. *Mathematika*, 17(2): 179–184, 1970.

[37] Paul Milgrom. Auction research evolving: Theorems and market designs. *American Economic Review*, 111(5):1383–1405, 2021.

[38] Roger B Myerson. Optimal auction design. *Mathematics of operations research*, 6(1):58–73, 1981.

[39] Harikrishna Narasimhan, Shivani Agarwal, and David C. Parkes. Automated mechanism design without money via machine learning. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 2016.

[40] Quynh Nguyen and Matthias Hein. The loss surface of deep and wide neural networks. In *International conference on machine learning*, pages 2603–2612. PMLR, 2017.

[41] Gregory Pavlov. Optimal mechanism for selling two goods. *The BE Journal of Theoretical Economics*, 11(1):000010220219351704 1664, 2011.

[42] Neehar Peri, Michael Curry, Samuel Dooley, and John Dickerson. Preferencenet: Encoding human preferences in auction design with deep learning. *Advances in Neural Information Processing Systems*, 34:17532–17542, 2021.

[43] Tian Qin, Fengxiang He, Dingfeng Shi, Wenbing Huang, and Dacheng Tao. Benefits of permutation-equivariance in auction mechanisms. *Advances in Neural Information Processing Systems*, 35:18131–18142, 2022.

[44] Jad Rahme, Samy Jelassi, Joan Bruna, and S Matthew Weinberg. A permutation-equivariant neural network architecture for auction design. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 5664–5672, 2021.

[45] Jad Rahme, Samy Jelassi, and S. Matthew Weinberg. Auction learning as a two-player game. In *International Conference on Learning Representations*, 2021.

[46] Sai Srivatsa Ravindranath, Zhe Feng, Shira Li, Jonathan Ma, Scott D Kominers, and David C Parkes. Deep learning for two-sided matching. *arXiv preprint arXiv:2107.03427*, 2021.

[47] Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, 1996.

[48] Tuomas Sandholm. Automated mechanism design: A new application area for search algorithms. In *International Conference on Principles and Practice of Constraint Programming*, pages 19–36. Springer, 2003.

[49] Tuomas Sandholm and Anton Likhodedov. Automated design of revenue-maximizing combinatorial auctions. *Operations Research*, 63(5):1000–1025, 2015.

[50] Weiran Shen, Pingzhong Tang, and Song Zuo. Automated mechanism design via neural networks. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19*, pages 215–223. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[51] Anatoliy V Skorokhod. Stochastic equations for diffusion processes in a bounded region. *Theory of Probability & Its Applications*, 6(3):264–274, 1961.

[52] Lieven Vandenberghe and Stephen Boyd. *Convex optimization*, volume 1. Cambridge university press Cambridge, 2004.

[53] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37, 1961.

[54] Tonghan Wang, Paul Duetting, Dmitry Ivanov, Inbal Talgam-Cohen, and David C Parkes. Deep contract design via discontinuous networks. *Advances in Neural Information Processing Systems*, 36:65818–65836, 2023.

[55] Tonghan Wang, Yanchen Jiang, and David C. Parkes. Gemnet: Menu-based, strategy-proof multi-bidder auctions through deep learning. 2024.

[56] Xavier Warin. The groupmax neural network approximation of convex functions. *IEEE Transactions on Neural Networks and Learning Systems*, 35(8):11608–11612, 2023.

[57] Robert Wilson. Auctions of shares. *The Quarterly Journal of Economics*, 93(4):675–689, 1979.

[58] Andrew Chi-Chih Yao. Dominant-strategy versus bayesian multi-item auctions: Maximum revenue determination and comparison. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 3–20, 2017.

[59] Peiran Yu, Junyi Li, and Heng Huang. Hessian free efficient single loop iterative differentiation methods for bi-level optimization problems. *Transactions on Machine Learning Research*, 2024.

[60] Stephan Zheng, Alexander Trott, Sunil Srinivasa, David C Parkes, and Richard Socher. The ai economist: Taxation policy design via two-level deep multiagent reinforcement learning. *Science advances*, 8(18):eabk2607, 2022.

APPENDIX

A FURTHER RELATED WORKS

Differentiable Economics. Differentiable Economics is a technical paradigm that aims to solve economic problems with differentiable functions (*e.g.*, neural networks) and gradient-based optimization. Within this field, much progress has been made in the context of social choice [39], facility location [24], matching market [46], contract design [54], market equilibrium [32], and economic tax design [60].

Automated Mechanism Design (AMD) is arguably one of the most widely studied topics in this area. The integration of differentiable economics into AMD was first studied by Dütting et al. [19], followed by Curry et al. [9; 11], Duan et al. [16], Ivanov et al. [29], Peri et al. [42], Rahme et al. [44; 45]. However, these approaches are not guaranteed to output truthful auctions. On the truthful-learning side, Shen et al. [50] initiates the learning of truthful single-buyer auctions, while Curry et al. [10], Duan et al. [17; 18], Ma et al. [33], Wang et al. [55] extend it to truthful multi-buyer auctions. This work also lies in the field of differentiable economics.

Bi-level Optimization. AMD can be viewed as a bi-level optimization problem [13] from a deeper perspective, where the outer level (auctioneer) optimizes revenue subject to the rationality constraints of the inner level (*e.g.*, regret minimization as in Dütting et al. [19] or affine welfare maximization as in Curry et al. [10]). Bi-level optimization dates back to Bracken and McGill [3] and has recently attracted attention in machine learning, with applications to hyperparameter optimization [21, 34] and meta-learning [20].

While bi-level algorithms generally require the strict convexity of the inner program and second-order derivatives to tackle the bi-level structure (see, for example, Chen et al. [6], Domke [14], Ghadimi and Wang [22], Huang et al. [28], Maclaurin et al. [34], Yu et al. [59]), this work only requires the (weak) convexity of the inner program and uses the *covariance trick* (Proposition 4.5) and Skorokhod reflection to tackle the bi-level structure, eliminating the need to compute second-order information. We believe this work provides a new insight for bi-level optimization.

Convex Neural Networks. Max-of-Affine (MoA) functions and Log-sum-exp (LSE) functions are two classical convex function classes, with universal approximation established by Calafiore et al. [5]. However, these function designs have incomplete network architectures and suffer from the curse of dimensionality, with a counterexample provided in McMullen [36].

Amos et al. [2] proposes Input Convex Neural Network (ICNN), a network-based convex function approximator, and the universal approximation property of ICNN is proved by Chen et al. [7]. Later, Warin [56] propose GroupMax Network and show its universal approximation property. Surprisingly, we find that ICNN generally outperforms GroupMax Network in our task. This work connects to this line of literature by utilizing ICNN as architectural design of Neural Affine Maximizers.

Auction Theory and Market Design. Beyond the differentiable economics literature, our work is grounded in the classical auction theory of Myerson [38] and the broader market design tradition [26, 37]. From the contract-theoretic viewpoint, auction design with privately informed buyers is a classic instance of screening under adverse selection: the auctioneer (a principal) commits to a mechanism, and the buyers (agents with private types) self-select by reporting their types, with (IC) and (IR) corresponding to the self-selection and participation constraints [38]. We note that the demand-schedule, price-quantity bidding formats studied in financial market microstructure, such as uniform-price auctions and double auctions [31, 57], operate under a different market model, where the auctioneer may not freely set the mechanism rules and bidder strategies are expressed as schedules rather than type reports. Extending AMD to such settings is non-trivial future direction.

B Omitted Proofs

B.1 Proof of Proposition 2.3

Proposition 2.3. (Truthfulness of AMA). *Let M^{AMA} be an AMA mechanism and M^{DIR} be a direct mechanism induced by M^{AMA}, then M^{DIR} is truthful and IR.*

Proof. Recall that $M^{\mathrm{AMA}} = \{\boldsymbol{x}^\kappa, \lambda^\kappa\}_{\kappa\in[L]}$ and $M^{\mathrm{DIR}} = (\boldsymbol{x}^{\mathrm{DIR}}, \boldsymbol{p}^{\mathrm{DIR}})$.

Consider the utility of player i by reporting t_i', when her true valuation is t_i and other players report $\boldsymbol{t}_{-i}$:

$$\begin{aligned} &u_i(x_i^{\mathrm{DIR}}(t_i';\boldsymbol{t}_{-i}), p_i^{\mathrm{DIR}}(t_i';\boldsymbol{t}_{-i}); t_i) = \langle x_i^{\mathrm{DIR}}(t_i';\boldsymbol{t}_{-i}), t_i\rangle - p_i^{\mathrm{DIR}}(t_i';\boldsymbol{t}_{-i}) \\ =&\langle x_i^{\kappa^*(t_i',\boldsymbol{t}_{-i})}, t_i\rangle + \mathrm{ASW}_{-i}(\kappa^*(t_i',\boldsymbol{t}_{-i});(t_i',\boldsymbol{t}_{-i})) - \mathrm{ASW}_{-i}(\kappa^{-i,*}(t_i',\boldsymbol{t}_{-i});(t_i',\boldsymbol{t}_{-i})) \end{aligned} \tag{12}$$

If player i reports truthfully, then her utility is

$$\begin{aligned} &u_i(x_i^{\mathrm{DIR}}(\boldsymbol{t}), p_i^{\mathrm{DIR}}(\boldsymbol{t}); t_i) \\ =&\mathrm{ASW}(\kappa^*(\boldsymbol{t});\boldsymbol{t}) - \mathrm{ASW}_{-i}(\kappa^{-i,*}(\boldsymbol{t});\boldsymbol{t}) \\ =&\max_\kappa \mathrm{ASW}(\kappa;\boldsymbol{t}) - \max_\kappa \mathrm{ASW}_{-i}(\kappa;\boldsymbol{t}) \end{aligned} \tag{13}$$

Since $\langle x_i^\kappa, t_i\rangle \geq 0$ always hold, we have $\mathrm{ASW}(\kappa;\boldsymbol{t}) \geq \mathrm{ASW}_{-i}(\kappa;\boldsymbol{t})$ for all k, therefore

$$u_i(x_i^{\mathrm{DIR}}(\boldsymbol{t}), p_i^{\mathrm{DIR}}(\boldsymbol{t}); t_i) = \max_\kappa \mathrm{ASW}(\kappa;\boldsymbol{t}) - \max_\kappa \mathrm{ASW}_{-i}(\kappa;\boldsymbol{t}) \geq 0 \tag{14}$$

This means that M^{DIR} is IR.

Notice that $\mathrm{ASW}_{-i}(\kappa; t_i', \boldsymbol{t}_{-i})$ does not rely on t_i' and t_i by definition, therefore, we transform Equation (12) to:

$$\begin{aligned} &u_i(x_i^{\mathrm{DIR}}(t_i';\boldsymbol{t}_{-i}), p_i^{\mathrm{DIR}}(t_i';\boldsymbol{t}_{-i}); t_i) = \langle x_i^{\kappa^*(t_i',\boldsymbol{t}_{-i})}, t_i\rangle + \mathrm{ASW}_{-i}(\kappa^*(t_i',\boldsymbol{t}_{-i});\boldsymbol{t}) - f(\boldsymbol{t}_{-i}) \\ =&\mathrm{ASW}(\kappa^*(t_i',\boldsymbol{t}_{-i});\boldsymbol{t}) - f(\boldsymbol{t}_{-i}) \end{aligned} \tag{15}$$

Here the last term is defined as $f(\boldsymbol{t}_{-i}) := \max_{\kappa\in[L]} \mathrm{ASW}_{-i}(\kappa;\boldsymbol{t})$, which equals $\mathrm{ASW}_{-i}(\kappa^{-i,*}(t_i',\boldsymbol{t}_{-i});(t_i',\boldsymbol{t}_{-i}))$ by the definition of $\kappa^{-i,*}$ and does not depend on t_i or t_i', since ASW_{-i} drops buyer i's valuation term.

By definition of maximization,

$$\mathrm{ASW}(\kappa^*(t_i',\boldsymbol{t}_{-i});\boldsymbol{t}) \leq \max_\kappa \mathrm{ASW}(\kappa;\boldsymbol{t}) = \mathrm{ASW}(\kappa^*(\boldsymbol{t});\boldsymbol{t}) \tag{16}$$

Note that after removing the term $f(\boldsymbol{t}_{-i})$ that is constant for player i, RHS of Equation (16) is the utility of truthful reporting for player i, while LHS of Equation (16) is the utility of reporting t_i' for player i. Therefore, by reporting $t_i' = t_i$, player i achieves her maximum utility. Therefore, M^{DIR} is truthful.

□

B.2 Proof of Proposition 3.1

Proposition 3.1. (Truthfulness of NAM) *Let M^{NAM} be an NAM and M^{DIR} be a direct mechanism induced by M^{NAM}. Then M^{DIR} is truthful and IR.*

Proof. Recall that the boosting function of an NAM is a continuous function $\lambda^{\mathrm{NAM}} : \mathcal{X} \to \mathbb{R}$ and $M^{\mathrm{DIR}} = (\boldsymbol{x}^{\mathrm{DIR}}, \boldsymbol{p}^{\mathrm{DIR}})$, where M^{DIR} is induced by M^{NAM}.

Consider the utility of player i by reporting t_i', when her true valuation is t_i and other players report $\boldsymbol{t}_{-i}$:

$$
\begin{aligned}
&u_i(x_i^{\mathrm{DIR}}(t_i';\boldsymbol{t}_{-i}),p_i^{\mathrm{DIR}}(t_i';\boldsymbol{t}_{-i});t_i)=\langle x_i^{\mathrm{DIR}}(t_i';\boldsymbol{t}_{-i}),t_i\rangle-p_i^{\mathrm{DIR}}(t_i';\boldsymbol{t}_{-i})\\
=&\langle x_i^{\mathrm{DIR}}(t_i';\boldsymbol{t}_{-i}),t_i\rangle-\mathrm{ASW}(\boldsymbol{x}^{-i}(t_i',\boldsymbol{t}_{-i});\boldsymbol{t}[-i])+\mathrm{ASW}(\boldsymbol{x}^{\mathrm{DIR}}(t_i',\boldsymbol{t}_{-i});\boldsymbol{t}[-i])
\end{aligned}
\tag{17}
$$

By reporting truthfully and note that $\mathrm{ASW}(\boldsymbol{x};\boldsymbol{t}[-i])+\langle x_i,t_i\rangle=\mathrm{ASW}(\boldsymbol{x};\boldsymbol{t})$, player i's utility becomes,

$$
\mathrm{ASW}(\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t});\boldsymbol{t})-\mathrm{ASW}(\boldsymbol{x}^{-i}(\boldsymbol{t});\boldsymbol{t}[-i])=\max_{\boldsymbol{x}}\mathrm{ASW}(\boldsymbol{x};\boldsymbol{t})-\max_{\boldsymbol{x}}\mathrm{ASW}(\boldsymbol{x};\boldsymbol{t}[-i])
\tag{18}
$$

As $\mathrm{ASW}(\boldsymbol{x};\boldsymbol{t})\geq\mathrm{ASW}(\boldsymbol{x};\boldsymbol{t}[-i])$ holds universally, we have $\max_{\boldsymbol{x}}\mathrm{ASW}(\boldsymbol{x};\boldsymbol{t})\geq\max_{\boldsymbol{x}}\mathrm{ASW}(\boldsymbol{x};\boldsymbol{t}[-i])$ holds universally as well. Therefore, M^{DIR} is IR.

Next, by observing that $\mathrm{ASW}(\boldsymbol{x}^{-i}(t_i',\boldsymbol{t}_{-i});\boldsymbol{t}[-i])=\mathrm{ASW}(\boldsymbol{x}^{-i}(t_i',\boldsymbol{t}_{-i});(t_i',\boldsymbol{t}_{-i})[-i])=\max_{\boldsymbol{x}}\mathrm{ASW}(\boldsymbol{x};(t_i';\boldsymbol{t}_{-i})[-i])=\max_{\boldsymbol{x}}\mathrm{ASW}(\boldsymbol{x};\boldsymbol{t}[-i])$, we know this term does not rely on t_i' and t_i. We then denote this term as $f(\boldsymbol{t}_{-i})$.

Then, player i's utility of reporting t_i' is:

$$
\begin{aligned}
&\langle x_i^{\mathrm{DIR}}(t_i';\boldsymbol{t}_{-i}),t_i\rangle+\mathrm{ASW}(\boldsymbol{x}^{\mathrm{DIR}}(t_i',\boldsymbol{t}_{-i});\boldsymbol{t}[-i])-f(\boldsymbol{t}_{-i})\\
=&\mathrm{ASW}(\boldsymbol{x}^{\mathrm{DIR}}(t_i';\boldsymbol{t}_{-i});\boldsymbol{t})-f(\boldsymbol{t}_{-i})\leq\max_{\boldsymbol{x}}\mathrm{ASW}(\boldsymbol{x};\boldsymbol{t})-f(\boldsymbol{t}_{-i})=\mathrm{ASW}(\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t});\boldsymbol{t})-f(\boldsymbol{t}_{-i})
\end{aligned}
\tag{19}
$$

where RHS of Equation (19) is the utility of truthful reporting for player i. Therefore, M^{DIR} is truthful. □

B.3 Proof of Theorem 3.2

Theorem 3.2. *Fix $M^{AMA}=\{(\boldsymbol{x}^\kappa,\lambda^\kappa)\}_{\kappa\in[L]}$ to be an arbitrary AMA mechanism (with weights $w_i\equiv 1$). Then, there exists an NAM $M^{NAM}=\{\lambda^{NAM}\}$ such that:*

(a) λ^{NAM} is concave on $\mathcal{X}$.

(b) If direct mechanism M^{DIR} is induced by M^{AMA}, then M^{DIR} is also induced by M^{NAM}.

Proof. Our proof consists of three steps: (a) We construct an extended AMA $\widetilde{M}^{\mathrm{AMA}}$ that is equivalent with M^{AMA} and its outcome candidates are sufficiently "broad"; (b) Based on $\widetilde{M}^{\mathrm{AMA}}$, we construct the λ^{NAM} and verify the concavity and continuity; (c) We show that if M^{DIR} is induced by $\widetilde{M}^{\mathrm{AMA}}$, then M^{DIR} is induced by $M^{\mathrm{NAM}}=\{\lambda^{\mathrm{NAM}}\}$.

Step (a). We first construct an extended AMA whose outcome candidates is sufficiently "broad".

Specifically, let $(y^1,...,y^J)$ be all vertices of $\mathcal{X}$ and $C\geq 0$ as a sufficiently large number. We consider the extended AMA $\widetilde{M}^{\mathrm{AMA}}=\{(\boldsymbol{x}^\kappa,\lambda^\kappa)\}_{k\in[L+J]}$ where $\boldsymbol{x}^{L+j}=y^j,\lambda^{L+j}=-C$ for $j\in[J]$. The C is chosen such that for all direct mechanism M^{DIR} induced by M^{AMA}, M^{DIR} is also induced by $\widetilde{M}^{\mathrm{AMA}}$.

We show that such C always exists. Let $\kappa\in[L]$ be arbitrary index. Consider the following inequality:

$$
\begin{aligned}
&\forall j\in[J],\boldsymbol{t}\in\mathcal{T},\quad\mathrm{ASW}(\kappa;\boldsymbol{t})>\mathrm{ASW}(L+j;\boldsymbol{t})\Leftrightarrow\forall j\in[J],\boldsymbol{t}\in\mathcal{T},\quad\sum_i\langle t_i,x_i^\kappa\rangle+\lambda^\kappa>\sum_i\langle t_i,y_i^j\rangle-C\\
\Leftrightarrow&\forall j\in[J],\boldsymbol{t}\in\mathcal{T},\quad C>\sum_i\langle t_i,y_i^j\rangle-\sum_i\langle t_i,x_i^\kappa\rangle-\lambda^\kappa
\end{aligned}
\tag{20}
$$

Since RHS is linear on $\boldsymbol{t}$, $\mathcal{T}$ is compact and $[J]$ is finite, we know that there is $C^*>0$ such that inequality (20) holds.

Define $\tilde{\kappa}^*(\boldsymbol{t}) = \text{argmax}_{k\in[L+J]}\text{ASW}(\kappa;\boldsymbol{t})$. Inequality (20) indicates $\kappa^*(\boldsymbol{t}) = \tilde{\kappa}^*(\boldsymbol{t})$ for all $\boldsymbol{t}$. Similarly, there is also $C_i^* > 0$ such that $\kappa^{-i,*}(\boldsymbol{t}) = \tilde{\kappa}^{-i,*}(\boldsymbol{t})$ for all $\boldsymbol{t}$. Let C be the maximum value of $C^*, \{C_i^*\}_{i\in[n]}$, we know that by choosing such a value of C, the computation flow of M^{AMA} in Equation (2) is identical of the computation flow of $\widetilde{M}^{\text{AMA}}$. Therefore, M^{DIR} being induced by M^{AMA} indicates that M^{DIR} is induced by $\widetilde{M}^{\text{AMA}}$.

Step (b). We *w.l.o.g.* assume that the outcome candidates of M^{AMA} cover all vertices of $\mathcal{X}$.

We define

$$\begin{aligned} A(\boldsymbol{x}) &= \{\{\alpha^\kappa\}_{\kappa\in[L]} : \sum_{\kappa\in[L]}\alpha^\kappa\boldsymbol{x}^\kappa = \boldsymbol{x};\ \sum_{\kappa\in[L]}\alpha^\kappa = 1, \forall\kappa\in[L], \alpha^\kappa \geq 0\} \\ \lambda^{\text{NAM}}(\boldsymbol{x}) &= \sup_{\{\alpha^\kappa\}_{\kappa\in[L]}\in A(\boldsymbol{x})}\sum_{\kappa\in[L]}\alpha^\kappa\lambda^\kappa \end{aligned} \tag{21}$$

.

Here, $\boldsymbol{x}$ is a convex combination of $\{\boldsymbol{x}^\kappa\}_{\kappa\in[L]}$ with coefficient $\{\alpha^\kappa\}_{\kappa\in[L]}$. As $\{\boldsymbol{x}^\kappa\}_{\kappa\in[L]}$ covers all vertices of $\mathcal{X}$, we know that such convex combination always exists, therefore $A(\boldsymbol{x})$ is non-empty and the supremum is well-defined. Moreover, $A(\boldsymbol{x})$ is a closed set. Therefore the supremum value can always be achieved at some $\{\alpha^\kappa\}_{\kappa\in[L]} \in A$.

We next show the concavity and continuity of λ^{NAM}.

We characterize λ^{NAM} via linear-programming duality. Viewed as a function of the right-hand side, λ^{NAM} is the optimal value of the linear program $V(\boldsymbol{x}, 1)$, where

$$V(\boldsymbol{z}) \coloneqq \max_{\alpha\geq 0}\left\{\sum_{\kappa\in[L]}\alpha^\kappa\lambda^\kappa \ :\ \sum_{\kappa\in[L]}\alpha^\kappa\begin{pmatrix}\boldsymbol{x}^\kappa\\1\end{pmatrix} = \boldsymbol{z}\right\}, \qquad \boldsymbol{z}\in\mathbb{R}^{d_x+1},\ d_x = nm,$$

whose feasible domain is non-empty and compact for every $\boldsymbol{z}\in\mathcal{X}\times\{1\}$ (as $\{\boldsymbol{x}^\kappa\}_{\kappa\in[L]}$ covers all vertices of $\mathcal{X}$), so V is finite on $\mathcal{X}\times\{1\}$. By strong duality, V admits the dual representation

$$V(\boldsymbol{z}) \ =\ \min_{\boldsymbol{y}}\left\{\langle\boldsymbol{z},\boldsymbol{y}\rangle \ :\ \langle\begin{pmatrix}\boldsymbol{x}^\kappa\\1\end{pmatrix},\boldsymbol{y}\rangle \geq \lambda^\kappa,\ \forall\kappa\in[L]\right\},$$

where the dual feasible set $P \coloneqq \{\boldsymbol{y} : \langle(\boldsymbol{x}^\kappa, 1), \boldsymbol{y}\rangle \geq \lambda^\kappa, \forall\kappa\in[L]\}$ is a non-empty polyhedron that does not depend on $\boldsymbol{z}$. Note that P may be unbounded, but it contains no lines. Indeed, in our setting $\mathcal{X} = (\Delta_n)^m$ is full-dimensional in $\mathbb{R}^{d_x}$ (its affine hull is $\mathbb{R}^{d_x}$ itself), and its vertices consist of the all-zero outcome and, for every good j and buyer i, the outcome that allocates good j to buyer i; since $\{\boldsymbol{x}^\kappa\}_{\kappa\in[L]}$ covers all these vertices, the vectors $\{(\boldsymbol{x}^\kappa, 1)\}_{\kappa\in[L]}$ span $\mathbb{R}^{d_x+1}$. Consequently, $\langle(\boldsymbol{x}^\kappa, 1), \boldsymbol{y}\rangle = 0$ for all $\kappa\in[L]$ forces $\boldsymbol{y} = \boldsymbol{0}$. Hence P is pointed and has finitely many vertices (extreme points) $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_J$; by the fundamental theorem of linear programming, whenever the optimum of the dual program is finite, it is attained at some vertex of P. Consequently, for every $\boldsymbol{z}\in\mathcal{X}\times\{1\}$ where V is finite, the value $V(\boldsymbol{z})$ equals a minimum over the finitely many affine functions $\boldsymbol{z}\mapsto\langle\boldsymbol{z},\boldsymbol{y}_j\rangle$, and we conclude

$$\lambda^{\text{NAM}}(\boldsymbol{x}) \ =\ V(\boldsymbol{x}, 1) \ =\ \min_{j\in[J]}\langle\boldsymbol{x},\boldsymbol{y}_j^x\rangle + \boldsymbol{y}_j^0,$$

where $\boldsymbol{y}_j = (\boldsymbol{y}_j^x, \boldsymbol{y}_j^0)$. Thus λ^{NAM} is a pointwise minimum of finitely many affine functions of $\boldsymbol{x}$, and is therefore concave and continuous on $\mathcal{X}$.

Step (c). The last step is to show that M^{DIR} is induced by M^{NAM} given M^{DIR} is induced by M^{AMA}. To see so, we need to verify the two inclusions and one equality in Equation (3), which is listed below:

$$
\begin{aligned}
\boldsymbol{t}[-i] \coloneqq & (t_1, ..., t_{i-1}, \mathbf{0}_m, t_{i+1}, ..., t_n) \\
\mathrm{ASW}(\boldsymbol{x}; \boldsymbol{t}) \coloneqq & \sum_{i \in [n]} \langle t_i, x_i \rangle + \lambda^{\mathrm{NAM}}(\boldsymbol{x}) \\
\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t}) \in & \mathrm{argmax}_{\boldsymbol{x} \in \mathcal{X}} \mathrm{ASW}(\boldsymbol{x}; \boldsymbol{t}) \\
\boldsymbol{x}^{-i}(\boldsymbol{t}) \in & \mathrm{argmax}_{\boldsymbol{x} \in \mathcal{X}} \mathrm{ASW}(\boldsymbol{x}; \boldsymbol{t}[-i]) \\
p_i^{\mathrm{DIR}}(\boldsymbol{t}) = & \mathrm{ASW}(\boldsymbol{x}^{-i}(\boldsymbol{t}); \boldsymbol{t}[-i]) - \mathrm{ASW}(\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t}); \boldsymbol{t}[-i])
\end{aligned}
\tag{22}
$$

We first show that the first inclusion holds ($\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t}) \in \mathrm{argmax}_{\boldsymbol{x} \in \mathcal{X}} \mathrm{ASW}(\boldsymbol{x}; \boldsymbol{t})$). Since a direct mechanism does not directly have $\boldsymbol{x}^{-i}(\boldsymbol{t})$, we would like to define that $\boldsymbol{x}^{-i}(\boldsymbol{t}) = \boldsymbol{x}^{\kappa^{-i,*}(\boldsymbol{t})}$. Then the second inclusion is similar to prove and is thus omitted.

Fix $\boldsymbol{t}$. Assume the contrary that $\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t})$ is not the maximum point of $\mathrm{ASW}(\cdot; \boldsymbol{t})$. Then, there is $\boldsymbol{x}^*$ such that $\mathrm{ASW}(\boldsymbol{x}^*; \boldsymbol{t}) > \mathrm{ASW}(\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t}); \boldsymbol{t})$. By definition of λ^{NAM}, there exists $\alpha \in A(\boldsymbol{x}^*)$ such that $\sum_k \alpha^\kappa \boldsymbol{x}^\kappa = \boldsymbol{x}^*$ and $\sum_k \alpha^\kappa \lambda^\kappa = \lambda^{\mathrm{NAM}}(\boldsymbol{x}^*)$. We have,

$$
\begin{aligned}
\mathrm{ASW}(\boldsymbol{x}^*; \boldsymbol{t}) &= \sum_{i \in [n]} \langle t_i, x_i^* \rangle + \lambda^{\mathrm{NAM}}(\boldsymbol{x}^*) \\
&= \sum_{i \in [n]} \sum_{\kappa \in [L]} \alpha^\kappa \langle t_i, x_i^\kappa \rangle + \sum_{\kappa \in [L]} \alpha^\kappa \lambda^\kappa \\
&= \sum_{\kappa \in [L]} \alpha^\kappa \mathrm{ASW}(\kappa; \boldsymbol{t})
\end{aligned}
\tag{23}
$$

Here the second equality uses $\sum_{\kappa \in [L]} \alpha^\kappa \lambda^\kappa = \lambda^{\mathrm{NAM}}(\boldsymbol{x}^*)$ by the choice of α, and the last equality uses $\mathrm{ASW}(\kappa; \boldsymbol{t}) = \sum_{i \in [n]} \langle t_i, x_i^\kappa \rangle + \lambda^\kappa$, the affine social welfare of the candidate $\boldsymbol{x}^\kappa$ in M^{AMA} (Equation (2)). Therefore, there is $\kappa \in [L]$ such that $\mathrm{ASW}(\boldsymbol{x}^\kappa; \boldsymbol{t}) \geq \mathrm{ASW}(\boldsymbol{x}^*; \boldsymbol{t}) > \mathrm{ASW}(\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t}); \boldsymbol{t}) = \max_{\kappa' \in [L]} \mathrm{ASW}(\boldsymbol{x}^{k'}; \boldsymbol{t})$, which leads to a contradiction. The last equation is due to the definition of equivalence between M^{DIR} and M^{AMA} in Equation (2).

Lastly, we verify the equation $p_i^{\mathrm{DIR}}(\boldsymbol{t}) = \mathrm{ASW}(\boldsymbol{x}^{-i}(\boldsymbol{t}); \boldsymbol{t}[-i]) - \mathrm{ASW}(\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t}); \boldsymbol{t}[-i])$. Note that

$$
p_i^{\mathrm{DIR}}(\boldsymbol{t}) \coloneqq \mathrm{ASW}_{-i}(\kappa^{-i,*}(\boldsymbol{t}); \boldsymbol{t}) - \mathrm{ASW}_{-i}(\kappa^*(\boldsymbol{t}); \boldsymbol{t}) \tag{24}
$$

We will prove independently that $\mathrm{ASW}_{-i}(\kappa^{-i,*}(\boldsymbol{t}); \boldsymbol{t}) = \mathrm{ASW}(\boldsymbol{x}^{-i}(\boldsymbol{t}); \boldsymbol{t}[-i])$ and $\mathrm{ASW}_{-i}(\kappa^*(\boldsymbol{t}); \boldsymbol{t}) = \mathrm{ASW}(\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t}); \boldsymbol{t}[-i])$.

To show this, we first show that $\lambda^{\kappa^*(\boldsymbol{t})} = \lambda^{\mathrm{NAM}}(\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t}))$

Since $\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t}) = \boldsymbol{x}^{\kappa^*(\boldsymbol{t})}$, it's straight-forward that $\lambda^{\mathrm{NAM}}(\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t})) \geq \lambda^{\kappa^*(\boldsymbol{t})}$ by definition of λ^{NAM} and supreme. If fix some $\boldsymbol{t}$ and $\lambda^{\mathrm{NAM}}(\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t})) > \lambda^{\kappa^*(\boldsymbol{t})}$, we could find some $\alpha \in A(\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t}))$ such that,

$$
\sum_k \alpha^\kappa \lambda^\kappa = \lambda^{\mathrm{NAM}}(\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t})), \qquad \sum_k \alpha^\kappa \boldsymbol{x}^\kappa = \boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t}) \tag{25}
$$

Then,

$$\begin{aligned}\mathrm{ASW}(\kappa^*(\boldsymbol{t});\boldsymbol{t}) =& \lambda^{\kappa^*(\boldsymbol{t})} + \sum_{i\in[n]} \langle t_i, x_i^{\kappa^*(\boldsymbol{t})}\rangle \\ =& \lambda^{\kappa^*(\boldsymbol{t})} + \sum_{i\in[n]} \langle t_i, x_i^{\mathrm{DIR}}(\boldsymbol{t})\rangle \cdots \text{by definition of } \boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t}) \\ <& \lambda^{\mathrm{NAM}}(\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t})) + \sum_{i\in[n]} \langle t_i, x_i^{\mathrm{DIR}}(\boldsymbol{t})\rangle \cdots \text{by assumption} \\ =& \sum_k \alpha^\kappa \left(\lambda^\kappa + \sum_{i\in[n]} \langle t_i, x_i^\kappa\rangle\right) \cdots \text{by equation in Equation (25)} \\ \leq& \max_\kappa \lambda^\kappa + \sum_{i\in[n]} \langle t_i, x_i^\kappa\rangle \\ =& \mathrm{ASW}(\kappa^*(\boldsymbol{t});\boldsymbol{t})\end{aligned} \tag{26}$$

which leads to a contradiction.

Similarly, we also have that $\lambda^{\kappa^{-i,*}(\boldsymbol{t})} = \lambda^{\mathrm{NAM}}(\boldsymbol{x}^{-i}(\boldsymbol{t}))$.

Based on above, we have

$$\begin{aligned}\mathrm{ASW}_{-i}(\kappa^{-i,*}(\boldsymbol{t});\boldsymbol{t}) &= \sum_{j\neq i} \langle t_j, x_j^{\kappa^{-i,*}(\boldsymbol{t})}\rangle + \lambda^{\kappa^{-i,*}(\boldsymbol{t})} \\ = \sum_{j\neq i} \langle t_j, x_j^{-i}(\boldsymbol{t})\rangle + \lambda^{\mathrm{NAM}}(\boldsymbol{x}^{-i}(\boldsymbol{t})) &= \mathrm{ASW}(\boldsymbol{x}^{-i}(\boldsymbol{t});\boldsymbol{t}[-i])\end{aligned} \tag{27}$$

and

$$\begin{aligned}\mathrm{ASW}_{-i}(\kappa^*(\boldsymbol{t});\boldsymbol{t}) &= \sum_{j\neq i} \langle t_j, x_j^{\kappa^*(\boldsymbol{t})}\rangle + \lambda^{\kappa^*(\boldsymbol{t})} \\ = \sum_{j\neq i} \langle t_j, x_j^{\mathrm{DIR}}(\boldsymbol{t})\rangle + \lambda^{\mathrm{NAM}}(\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t})) &= \mathrm{ASW}(\boldsymbol{x}^{\mathrm{DIR}}(\boldsymbol{t});\boldsymbol{t}[-i])\end{aligned} \tag{28}$$

which together completes the proof. □

B.4 Proof of Theorem 4.4

Theorem 4.4. *Let $\mathcal{Y} \subseteq \mathbb{R}^{d_x}$ be a convex, compact subset of d_x-dimensional Euclidean space. Let $\mathcal{G} \coloneqq \{f : \mathcal{Y} \to \mathbb{R} | f$ is an ICNN$\}$ and $\mathcal{H} \coloneqq \{f : \mathcal{Y} \to \mathbb{R} | f$ is convex and continuous$\}$. Then, $\mathcal{G}$ is a universal approximator of $\mathcal{H}$.*

Proof of Theorem 4.4. **Prove (1).**

Let $f : \mathcal{Y} \to \mathbb{R}$ be a function represented by an ICNN. We will actually prove that f is continuous and convex.

First, notice that the composition of continuous functions is always continuous. Also note that in ICNN, all computation flows, including linear transformations and activation functions, are continuous transformations. Therefore, f is also continuous.

Second, recall that a non-negatively weighted sum of convex functions is convex, and that $g \circ h$ is convex if g is convex and non-decreasing and h is convex. In Figure 1, every element in h_1 is a linear function of x. Then, every element of x_1, which is a convex, non-decreasing function of some element in h_1, is a convex function of x. Now assume that every element of x_k is a convex function of x. Then, every element of h_{k+1}, which is a non-negatively weighted sum of x_k plus a linear function of x, is a convex function of x; every element of x_{k+1}, which is a convex, non-decreasing function of some element in h_{k+1}, is also convex in x. In all, we conclude that x_K is convex in x. Therefore, y, the output of ICNN, which is a non-negative-weighted sum of x_K, is convex on x. This completes the proof of (1).

Prove (2).

It is not hard to verify that if $\mathcal{F}_1$ is a universal approximator of $\mathcal{F}_2$ and $\mathcal{F}_2$ is a universal approximator of $\mathcal{F}_3$, then $\mathcal{F}_1$ is a universal approximator of $\mathcal{F}_3$.

A well-known result is that the set of max-of-affine functions is a universal approximator of $\mathcal{H}$ [5]. Therefore, we only need to prove that every max-of-affine function can be exactly represented by some ICNN.

A max-of-affine function is represented by

$$f(x) = \max_{i\in[n]} \quad \langle w_i, x\rangle + b_i \tag{29}$$

with $b_i \in \mathbb{R}, w_i \in \mathbb{R}^{d_x}$.

We will prove this by mathematical induction on n, and prove that every max-of-n-affine function can be exactly represented by ICNN with $d_h = 1$.

When $n = 1$, $f(x) = \langle w_1, x\rangle + b_1$. Take a 1-layer ICNN ($K = 1$) with $W_0^r = 0$, $b_0 = 0$, $\sigma_1 = \text{ReLU}$, $w^c = 0$, $w^r = w_1$, and $b = b_1$. Then $x_1 = \sigma_1(\langle W_0^r, x\rangle + b_0) = 0$ and $y(x;\theta) = w^c \cdot x_1 + \langle w^r, x\rangle + b = \langle w_1, x\rangle + b_1 = f(x)$, finishing the proof.

Now let $n \geq 2$. By the inductive hypothesis, every max-of-$(n-1)$-affine function can be exactly represented by ICNN. It is clear that Equation (29) can be rewritten as:

$$f(x) = \langle w_n, x\rangle + b_n + \max(0, g(x)) \tag{30}$$

where $g(x) = \max_{i\in[n-1]}\langle w_i - w_n, x\rangle + (b_i - b_n)$. We know that $g(x)$ can be represented by a K-layer ICNN.

Let θ, including $\{W_k^c\}_{1\leq k\leq K-1}$, $\{W_k^r\}_{0\leq k\leq K-1}$, $\{b_k\}_{0\leq k\leq K-1}$, $w^c \in \mathbb{R}_+$, $w^r \in \mathbb{R}^{d_x}$, $b \in \mathbb{R}$, $\{\sigma_k\}_{k\in[K]}$, where $W_k^c \in \mathbb{R}_+$, $W_k^r \in \mathbb{R}^{d_x}$, $b_k \in \mathbb{R}$, be the K-layer ICNN that represents $g(x)$.

Next, we construct a $K+1$-layer ICNN with parameters θ' including $\{W_k'^c\}_{1\leq k\leq K}$, $\{W_k'^r\}_{0\leq k\leq K}$, $\{b_k'\}_{0\leq k\leq K}$, $w'^c \in \mathbb{R}_+$, $w'^r \in \mathbb{R}^{d_x}$, $b' \in \mathbb{R}$, $\{\sigma_k'\}_{k\in[K+1]}$, where $W_k'^c \in \mathbb{R}_+$, $W_k'^r \in \mathbb{R}^{d_x}$, $b_k \in \mathbb{R}$.

We set $W_k'^c = W_k^c, W_k'^r = W_k^r, b_k' = b_k$ for $k \leq K-1$, $\sigma_k' = \sigma_k$ for $k \in [K]$ and $W_K'^c = w^c, W_K'^r = w^r, b_K' = b$. It's clear to see that $h_{K+1}(x;\theta') = y(x;\theta) = g(x)$, where $h_{K+1}(x;\theta')$ represents the hidden element h_{K+1} in the ICNN parameterized by θ'.

Let $\sigma_{K+1}(\cdot) = \text{RELU}(\cdot)$, then $x_{K+1}(x;\theta') = \max(0, h_{K+1}(x;\theta')) = \max(0, g(x))$. Let $w_c' = 1, w_r' = w_n, b' = b_n$, we know that $y(x;\theta') = w_c' \cdot x_{K+1}(x;\theta') + \langle w_r', x\rangle + b' = \langle w_n, x\rangle + b_n + \max(0, g(x)) = f(x)$. Therefore, we complete the proof.

□

B.5 Proof of Proposition 4.5

Proposition 4.5. *The following identity holds:*

$$\nabla_\theta \mathbb{E}_{\boldsymbol{x}^{-i}}\Big[u_0(\{\boldsymbol{x}^{-i}\}_i, \theta; \boldsymbol{t})\Big] = \mathbb{E}_{\boldsymbol{x}^{-i}}\Big[\nabla_\theta u_0(\{\boldsymbol{x}^{-i}\}_i, \theta; \boldsymbol{t})\Big] + \beta \cdot \text{Cov}_{\boldsymbol{x}^{-i}}\Big[u_0(\{\boldsymbol{x}^{-i}\}_i, \theta; \boldsymbol{t}), \nabla_\theta \sum_{i\in[n+1]} \text{ASW}(\boldsymbol{x}^{-i}; \boldsymbol{t}[-i], \theta)\Big]$$

where $\mathbb{E}_{\boldsymbol{x}^{-i}}$ *and* $\text{Cov}_{\boldsymbol{x}^{-i}}$ *mean that* $\boldsymbol{x}^{-i} \overset{\text{i.d.}}{\sim} q^\beta(\cdot|\boldsymbol{t}[-i], \theta)$ *for* $i \in [n+1]$.

Proof. Fix $\boldsymbol{t}$ and $\beta > 0$. For each $i \in [n+1]$, write

$$q_i(\boldsymbol{x}^{-i};\theta) := q_\beta(\boldsymbol{x}^{-i} \mid \boldsymbol{t}[-i], \theta) = \frac{\exp\big(\beta \cdot \text{ASW}(\boldsymbol{x}^{-i}; \boldsymbol{t}[-i], \theta)\big)}{Z_\beta(\boldsymbol{t}[-i], \theta)},$$

$$Z_\beta(\boldsymbol{t}[-i], \theta) := \int_{a\in\mathcal{X}} \exp\big(\beta \cdot \text{ASW}(a; \boldsymbol{t}[-i], \theta)\big)\, \mathrm{d}a.$$

Let $\boldsymbol{x}^{-[n+1]} := (\boldsymbol{x}^{-i})_{i\in[n+1]} \in \mathcal{X}^{n+1}$ and define the product density $q(\boldsymbol{x}^{-[n+1]};\theta) := \prod_{i\in[n+1]} q_i(\boldsymbol{x}^{-i};\theta)$. Then

$$\mathbb{E}_{\boldsymbol{x}^{-i}\sim q_\beta(\cdot|\boldsymbol{t}[-i],\theta)}\big[u_0(\boldsymbol{x}^{-[n+1]}, \theta; \boldsymbol{t})\big] = \int_{\boldsymbol{x}^{-[n+1]}\in\mathcal{X}^{n+1}} u_0(\boldsymbol{x}^{-[n+1]}, \theta; \boldsymbol{t})\, q(\boldsymbol{x}^{-[n+1]};\theta) \prod_{i\in[n+1]} \mathrm{d}x_i.$$

By the chain rule of differentiation,

$$\begin{aligned}\nabla_\theta \mathbb{E}_{\boldsymbol{x}^{-i}}\big[u_0(\boldsymbol{x}^{-[n+1]},\theta;\boldsymbol{t})\big] &= \int \nabla_\theta u_0(\boldsymbol{x}^{-[n+1]},\theta;\boldsymbol{t})\, q(\boldsymbol{x}^{-[n+1]};\theta)\prod_i \mathrm{d}x_i \\ &+ \int u_0(\boldsymbol{x}^{-[n+1]},\theta;\boldsymbol{t})\,\nabla_\theta q(\boldsymbol{x}^{-[n+1]};\theta)\prod_i \mathrm{d}x_i \\ &= \mathbb{E}_{\boldsymbol{x}^{-i}}\big[\nabla_\theta u_0(\boldsymbol{x}^{-[n+1]},\theta;\boldsymbol{t})\big] \;+\; \mathbb{E}_{\boldsymbol{x}^{-i}}\Big[u_0(\boldsymbol{x}^{-[n+1]},\theta;\boldsymbol{t})\,\nabla_\theta \log q(\boldsymbol{x}^{-[n+1]};\theta)\Big],\end{aligned} \tag{31}$$

where we used the log-gradient identity $\nabla_\theta q = q\,\nabla_\theta \log q$.

Since $q(\boldsymbol{x}^{-[n+1]};\theta) = \prod_i q_i(\boldsymbol{x}^{-i};\theta)$, we have $\nabla_\theta \log q(\boldsymbol{x}^{-[n+1]};\theta) = \sum_{i\in[n+1]} \nabla_\theta \log q_i(\boldsymbol{x}^{-i};\theta)$. Moreover,

$$\log q_i(a;\theta) = \beta\,\mathrm{ASW}(a;\boldsymbol{t}[-i],\theta) - \log Z_\beta(\boldsymbol{t}[-i],\theta),$$

so

$$\nabla_\theta \log q_i(a;\theta) = \beta\,\nabla_\theta \mathrm{ASW}(a;\boldsymbol{t}[-i],\theta) - \nabla_\theta \log Z_\beta(\boldsymbol{t}[-i],\theta).$$

A direct calculation gives

$$\begin{aligned}\nabla_\theta Z_\beta(\boldsymbol{t}[-i],\theta) &= \int_{a\in\mathcal{X}} \exp\big(\beta\cdot\mathrm{ASW}(a;\boldsymbol{t}[-i],\theta)\big)\cdot\beta\,\nabla_\theta \mathrm{ASW}(a;\boldsymbol{t}[-i],\theta)\,\mathrm{d}a, \\ \nabla_\theta \log Z_\beta(\boldsymbol{t}[-i],\theta) &= \frac{\nabla_\theta Z_\beta(\boldsymbol{t}[-i],\theta)}{Z_\beta(\boldsymbol{t}[-i],\theta)} = \beta\int_{a\in\mathcal{X}} q_i(a;\theta)\,\nabla_\theta \mathrm{ASW}(a;\boldsymbol{t}[-i],\theta)\,\mathrm{d}a \\ &= \beta\,\mathbb{E}_{\boldsymbol{x}^{-i}\sim q_i(\cdot;\theta)}\big[\nabla_\theta \mathrm{ASW}(\boldsymbol{x}^{-i};\boldsymbol{t}[-i],\theta)\big].\end{aligned}$$

Therefore, for each $i \in [n+1]$,

$$\nabla_\theta \log q_i(\boldsymbol{x}^{-i};\theta) = \beta\Big(\nabla_\theta \mathrm{ASW}(\boldsymbol{x}^{-i};\boldsymbol{t}[-i],\theta) - \mathbb{E}_{\boldsymbol{x}^{-i}}\big[\nabla_\theta \mathrm{ASW}(\boldsymbol{x}^{-i};\boldsymbol{t}[-i],\theta)\big]\Big).$$

Plugging this back yields

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{x}^{-i}}\Big[u_0(\boldsymbol{x}^{-[n+1]},\theta;\boldsymbol{t})\,\nabla_\theta \log q(\boldsymbol{x}^{-[n+1]};\theta)\Big] \\ =&\sum_{i\in[n+1]} \mathbb{E}_{\boldsymbol{x}^{-i}}\Big[u_0(\boldsymbol{x}^{-[n+1]},\theta;\boldsymbol{t})\,\nabla_\theta \log q_i(\boldsymbol{x}^{-i};\theta)\Big] \\ =&\beta\sum_{i\in[n+1]}\Big(\mathbb{E}\big[u_0(\boldsymbol{x}^{-[n+1]},\theta;\boldsymbol{t})\,\nabla_\theta \mathrm{ASW}(\boldsymbol{x}^{-i};\boldsymbol{t}[-i],\theta)\big] - \mathbb{E}[u_0(\boldsymbol{x}^{-[n+1]},\theta;\boldsymbol{t})]\,\mathbb{E}[\nabla_\theta \mathrm{ASW}(\boldsymbol{x}^{-i};\boldsymbol{t}[-i],\theta)]\Big) \\ =&\beta\,\mathrm{Cov}_{\boldsymbol{x}^{-i}}\Big(u_0(\boldsymbol{x}^{-[n+1]},\theta;\boldsymbol{t}),\,\nabla_\theta \sum_{i\in[n+1]} \mathrm{ASW}(\boldsymbol{x}^{-i};\boldsymbol{t}[-i],\theta)\Big),\end{aligned} \tag{32}$$

Combining Equation (32) with Equation (31) proves Proposition 4.5.

□

B.6 Computation Flow of ICNN

Given input $x \in \mathbb{R}^{d_x}$, the computation flow of ICNN to output $y = f(x) \in \mathbb{R}$ is illustrated in Algorithm 1.

B.7 Pseudocode of training NAMs

Following the steps described in Section 4.2, we provide a pseudocode of training NAMs in Algorithm 2.

Algorithm 1: Computation of ICNN

Input: $x \in \mathbb{R}^{d_x}$

Output: $y = f(x) \in \mathbb{R}$

parameters: $\{W_k^c\}_{1\le k\le K-1}$, $\{W_k^r\}_{0\le k\le K-1}$, $\{b_k\}_{0\le k\le K-1}$, $w^c \in \mathbb{R}_+^{d_h}$, $w^r \in \mathbb{R}^{d_x}$, $b \in \mathbb{R}$, where $\forall k$, $W_k^c \in \mathbb{R}_+^{d_h \times d_h}$, $W_k^r \in \mathbb{R}^{d_h \times d_x}$, $b_k \in \mathbb{R}^{d_h}$

$x^0 \leftarrow x$;

$h^1 \leftarrow W_0^r \cdot x + b_0$;

$x^1 \leftarrow \sigma_1(h^1)$, with activation function applied element-wise;

for $k = 1, 2, ..., K-1$ **do**

  $h^{k+1} \leftarrow W_k^c \cdot x^k + W_k^r \cdot x + b_k$;

  $x^{k+1} \leftarrow \sigma_{k+1}(h^{k+1})$, with activation function applied element-wise;

end

$y \leftarrow \langle w^c, x^K \rangle + \langle w^r, x \rangle + b$;

return y

Algorithm 2: Training of NAM

Input: a mechanism design problem $(n, m, \mathcal{F})$

Output: the ICNN parameter θ of the boosting function $\lambda^{\text{NAM}}(\cdot\,; \theta)$

parameters: B: batch size, T: iterations of Langevin dynamics, I: total training iterations, β: temperature, η: step size in Langevin dynamics, η_θ: learning rates for θ

Initialize θ randomly;

Initialize the samples $\{\boldsymbol{y}^{-i,b}, \boldsymbol{z}^{-i,b}\}_{i,b}$ by i.i.d. samples from a uniform distribution followed by a softmax so that they lie in $\mathcal{X}$;

Sample a batch of type profiles $\{\boldsymbol{t}^b\}_{b\in[B]} \overset{\text{i.i.d.}}{\sim} \mathcal{F}$;

for $t = 1, 2, ..., I$ **do**

  `/* a) Updating the samples by Langevin dynamics */`

  for $b = 1, 2, ..., B;\quad i = 1, 2, ..., n+1$ **do**

    for $s = 1, 2, ..., T$ **do**

      $\boldsymbol{y}^{-i,b} \leftarrow \boldsymbol{y}^{-i,b} + \eta \cdot \nabla_{\boldsymbol{x}} \mathrm{ASW}(\boldsymbol{y}^{-i,b}; \boldsymbol{t}^b[-i], \theta) + \sqrt{2\eta\beta^{-1}} \cdot \boldsymbol{\epsilon}^{-i,b,y}$ where $\boldsymbol{\epsilon}^{-i,b,y} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \mathrm{Id})$ and Id is the nm-th order unit matrix (notice that $\boldsymbol{y}^{-i,b}$ is nm-dimensional);

      $\boldsymbol{z}^{-i,b} \leftarrow \boldsymbol{z}^{-i,b} + \eta \cdot \nabla_{\boldsymbol{x}} \mathrm{ASW}(\boldsymbol{z}^{-i,b}; \boldsymbol{t}^b[-i], \theta) + \sqrt{2\eta\beta^{-1}} \cdot \boldsymbol{\epsilon}^{-i,b,z}$ where $\boldsymbol{\epsilon}^{-i,b,z} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \mathrm{Id})$;

      Project $\boldsymbol{y}^{-i,b}, \boldsymbol{z}^{-i,b}$ onto $\mathcal{X}$ using Skorokhod reflection;

    end

  end

  `/* b) Updating the parameter θ by the covariance trick */`

  Compute the gradient surrogate $\mathcal{J}^\beta(\theta; \mathcal{T}^B)$ in Equation (10) using $\boldsymbol{y}^{-i,b}$ and $\boldsymbol{z}^{-i,b}$;

  Compute $\nabla_\theta \mathcal{J}^\beta(\theta; \mathcal{T}^B)$, and update θ using stochastic gradient ascent;

  `/* c) Learning rate schedule */`

  Decrease η_θ and η when appropriate;

end

return: the learned θ

C More Experimental Details

C.1 Implementation Details

Hyperparameters. We list the hyperparameters used for NAM in Table 3. Recall that n is the number of buyers and m is the number of goods.

Table 3: Hyperparameters.

Hyperparameter	Value
samples for training: B	2^{14}
samples for evaluation:	2^{16}
ICNN network layer: K	3
ICNN network hidden dimension: d_h	$10nm$
ICNN initial learning rate	$3 \cdot 10^{-4}$
ICNN final learning rate	$3 \cdot 10^{-5}$
ICNN optimizer	Adam($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$)
step size for Langevin dynamics η	0.04
Langevin dynamics iteration T	32
training iterations	$10,000$
temperature: β	512
menu size (for LotteryAMA and AMenuNet)	$8(n+m)^2$

For self-implemented LotteryAMA, we use the same hyperparameter values reported in the original papers [10]. We decrease the default batch size of AMenuNet solely in large-scale settings (i.e., ≥ 10 buyers and 10 goods) for memory constraints.

Inference. At test time, the outcomes $\boldsymbol{x}^{-i}(\boldsymbol{t})$ are computed by batched projected gradient ascent (with Nesterov momentum, 500 iterations and fine-tuned learning rate schedule) on the concave objective function $\mathrm{ASW}(\cdot; \boldsymbol{t}[-i], \theta)$ over $\mathcal{X}$, with a projection onto $\mathcal{X}$ after each iteration. We empirically find that this procedure leads to $\sim 10^{-6}$ precision on the objective function, leading to sufficiently precise computation of global maximizers. We note that the precision of the estimated revenue and the DSIC approximation degree (regret) are also controlled by the solver precision. One can tune the inference procedure to get an arbitrary high precision, at the cost of more compute.

C.2 Complexity Analysis

As ICNN has a fully connected architecture with linearly-scaling hidden dimension, computing each $\mathrm{ASW}(\boldsymbol{x})$ takes $\Theta(d_h^2) = \Theta(n^2m^2)$ time. One training step of NAM—which runs T Langevin iterations (Equation (11)) for each of the B samples and each of the $n+1$ constraints—has complexity $\Theta(B \cdot T \cdot n^3m^2)$. Through our experiments, B, T are made constants.

For both LotteryAMA and AMenuNet, one inference of $\kappa^*(\boldsymbol{t})$ requires computing the argmax of $\mathrm{ASW}(k; \boldsymbol{t})$ over all candidate outcomes $\boldsymbol{x}^k$. In addition, each candidate outcome is of size nm. As a consequence, the total complexity of one training step for LotteryAMA and AMenuNet is $\Theta((n+1) \cdot 8(n+m)^2 \cdot nm) = \Omega(n^3m^2)$.

C.3 More Experimental Results

Results on more distributions $\mathcal{F}$. We consider two additional distributions as experimental settings:

- $\mathcal{F} = B$: $t_{ij} \overset{\text{i.i.d.}}{\sim} \mathrm{Bernoulli}(p)$,[6] where $B_{15,10}$ uses $p = 0.05$ and the other settings use $p = 0.2$.
- $\mathcal{F} = C$: First generating $t_j \overset{\text{i.i.d.}}{\sim} U([0.25, 0.75])$, then $t_{ij} \overset{\text{i.i.d.}}{\sim} U([t_j - 0.25, t_j + 0.25])$. This is also used in Ma et al. [33]. Note that buyers' types are correlated in this distribution.

[6] A Bernoulli random variable with parameter p has prob. p to be 1 and prob. $1-p$ to be 0.

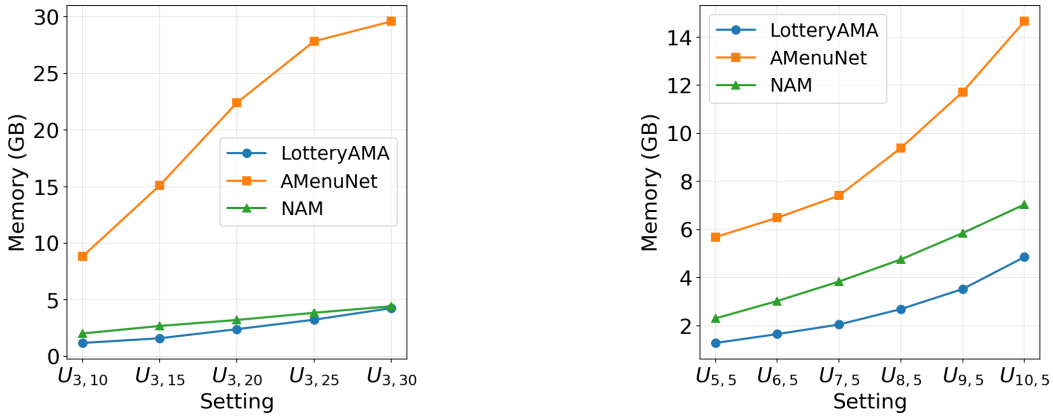
Table 4: Auctioneer's revenue with n buyers, m goods and two distributions: the Bernoulli distribution B and the correlated distribution C. '*' represents provably optimal values.

Method & Settings	$B_{2,2}$	$B_{3,10}$	$B_{3,30}$	$B_{15,10}$	$C_{2,2}$	$C_{3,10}$	$C_{3,30}$	$C_{15,10}$
NAM	0.669$_{\pm 0.087}$	4.718$_{\pm 0.104}$	14.267$_{\pm 0.152}$	4.991$_{\pm 0.144}$	0.893$_{\pm 0.001}$	5.402$_{\pm 0.002}$	**16.535**$_{\pm 0.006}$	**6.878**$_{\pm 0.007}$
LotteryAMA	0.623$_{\pm 0.066}$	2.769$_{\pm 0.231}$	7.604$_{\pm 0.017}$	1.985$_{\pm 0.008}$	0.893$_{\pm 0.001}$	5.089$_{\pm 0.014}$	14.910$_{\pm 0.223}$	5.461$_{\pm 0.022}$
AMenuNet	0.657$_{\pm 0.017}$	3.018$_{\pm 0.061}$	7.918$_{\pm 0.037}$	2.235$_{\pm 0.041}$	**0.896**$_{\pm 0.003}$	5.326$_{\pm 0.003}$	15.670$_{\pm 0.005}$	5.984$_{\pm 0.013}$
OD-VVCA	0.616$_{\pm 0.014}$	2.524$_{\pm 0.014}$	-	-	**0.893**$_{\pm 0.004}$	**5.407**$_{\pm 0.006}$	-	-
Itemwise	**0.720***	**4.880***	**14.640***	**5.367***	0.851	5.026	15.082	6.875
Bundle	0.595	2.183	6.073	1.494	0.887	5.003	14.998	5.573
VCG	0.080	1.040	3.120	1.709	0.833	5.000	15.000	6.875

There is one subtlety when Itemwise is applied in settings with correlated types, because Myerson's auction is only guaranteed optimal in independent types settings. Actually, we implement Itemwise as a second-price auction with parameterized reserve price. The optimal reserve price is found through grid search. This recovers the optimal Myerson's auction when buyers have independent and symmetric types distributions (as in all considered distributions except $\mathcal{F} = C$), and serves as a competitive baseline when buyers' types are correlated ($\mathcal{F} = C$).

We present the results on these distributions in Table 4. In $B_{n,m}$ settings, optimal auctions provably require covering all $(n+1)^m$ deterministic outcomes [33]. NAM attains $93\% \sim 97\%$ of the best value across all $B_{n,m}$ settings, whereas the learning-based baselines find highly suboptimal auctions—the best learning-based baseline achieves at most 54% of the best value in $B_{3,30}$ and 42% in $B_{15,10}$. This suggests that discretization-based approaches struggle to explore a full coverage of the deterministic outcome set. We also notice that NAM performs suboptimally in some random seeds in Bernoulli distributions, leading the high variance of NAM's performance. For correlated distribution, we can see that NAM performs optimally or competitively with all other approaches.

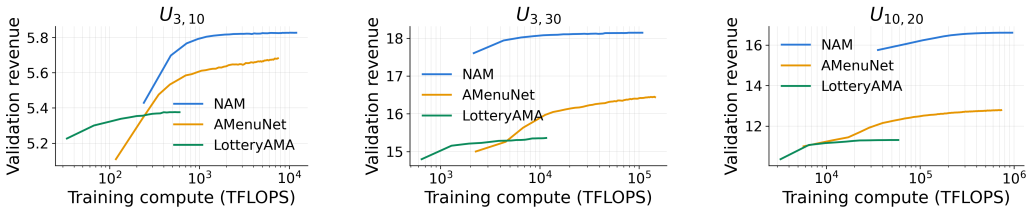
Required Memory. We show the memory used for different methods and settings in Figure 4.



(a) (b)

Figure 4: The comparison of required memory for LotteryAMA, AMenuNet and NAM.

Revenue as a function of TFLOPS. We compare the validation revenue of NAM , LotteryAMA, and AMenuNet against their cumulative training compute (in TFLOPS, logarithmic axis) in Figure 5. In $U_{3,10}$, NAM surpasses LotteryAMA and AMenuNet in every fixed compute budget. More surprisingly, in $U_{3,30}$ and $U_{10,20}$, NAM can outperform the well-trained LotteryAMA and AMenuNet with only 2% the compute budget (the most left point in NAM's curves).



(a) $U_{3,10}$ (b) $U_{3,30}$ (c) $U_{10,20}$

Figure 5: Validation revenue as a function of cumulative training compute (TFLOPS, log scale) for different settings and approaches.