Bayesian Optimization-Based Combinatorial Assignment[*]

Jakob Weissteiner[1,3†], **Jakob Heiss**[2,3†], **Julien Siems**[1†] and **Sven Seuken**[1,3]
[1]University of Zurich
[2]ETH Zurich
[3]ETH AI Center
weissteiner@ifi.uzh.ch, jakob.heiss@math.ethz.ch, juliensiems@gmail.com, seuken@ifi.uzh.ch

Abstract

We study the combinatorial assignment domain, which includes combinatorial auctions and course allocation. The main challenge in this domain is that the bundle space grows exponentially in the number of items. To address this, several papers have recently proposed *machine learning-based preference elicitation* algorithms that aim to elicit only the most important information from agents. However, the main shortcoming of this prior work is that it does not model a mechanism's uncertainty over values for not yet elicited bundles. In this paper, we address this shortcoming by presenting a *Bayesian optimization-based combinatorial assignment (BOCA)* mechanism. Our key technical contribution is to integrate a method for capturing model uncertainty into an iterative combinatorial auction mechanism. Concretely, we design a new method for estimating an *upper uncertainty bound* that can be used to define an acquisition function to determine the next query to the agents. This enables the mechanism to properly *explore* (and not just *exploit*) the bundle space during its preference elicitation phase. We run computational experiments in several spectrum auction domains to evaluate BOCA's performance. Our results show that BOCA achieves higher allocative efficiency than state-of-the-art approaches.

1 Introduction

Many economic problems require finding an efficient combinatorial assignment of multiple indivisible items to multiple agents. Popular examples include *combinatorial auctions (CAs)*, *combinatorial exchanges (CEs)*, and *combinatorial course allocation*. In CAs, heterogeneous items are allocated among a set of bidders, e.g., for the sale of spectrum licenses (Cramton 2013). In CEs, items are allocated among agents who can be sellers *and* buyers at the same time, e.g., for the reallocation of catch shares (Bichler, Fux, and Goeree 2019). In course allocation, course seats are allocated among students at universities (Budish 2011).

In all these domains, agents have preferences over *bundles* of items. In particular, agents' preferences may exhibit *complementarities* and *substitutabilities* among items. A mechanism that allows agents to report values for bundles rather than just for individual items can achieve significantly higher efficiency. However, this also implies that agents' preferences are exponentially-sized (i.e., for m items there are 2^m different bundles), which implies that agents cannot report values for all bundles. Therefore, the key challenge in combinatorial assignment is the design of a *preference elicitation (PE)* algorithm that is (i) *practically feasible* w.r.t. elicitation costs and (ii) *smart*, i.e., it should elicit the information that is "most useful" for achieving high efficiency.

1.1 Iterative Combinatorial Auctions (ICAs)

While the PE challenge is common to all combinatorial assignment problems, it has been studied most intensely in the CA domain (Sandholm and Boutilier 2006). In CAs with general valuations, the amount of communication needed to guarantee full efficiency is exponential in the number of items (Nisan and Segal 2006). Thus, practical CAs cannot provide efficiency guarantees. In practice, *iterative combinatorial auctions (ICAs)* are therefore employed, where the auctioneer interacts with bidders over rounds, eliciting a *limited* (and thus *practically feasible*) amount of information, aiming to find a highly efficient allocation. ICAs are widely used; e.g., for the sale of licenses to build offshore wind farms (Ausubel and Cramton 2011). The provision of spectrum licenses via the *combinatorial clock auction (CCA) (Ausubel, Cramton, and Milgrom 2006)* has generated more than \$20 billion in total revenue (Ausubel and Baranov 2017). Thus, increasing the efficiency of such real-world ICAs by only 1% point translates into monetary gains of hundreds of millions of dollars.

1.2 ML-Powered Preference Elicitation

In recent years, researchers have used machine learning (ML) to design smart PE algorithms. Most related to this paper is the work by Brero, Lubin, and Seuken (2018, 2021), who developed the first practical ML-powered ICA that outperforms the CCA. The main idea of their mechanism is two-fold: first, they train a separate *support vector regression* model to learn each bidder's full value function from a small set of bids; second, they solve an *ML-based winner determination problem (WDP)* to determine the allocation with the highest predicted social welfare, and they use this allocation to generate the next set of queries to all bidders.

[*]The full paper including appendix is available on arXiv via: https://arxiv.org/abs/2208.14698.

[†]These authors contributed equally.

This process repeats in an iterative fashion until a fixed number of queries has been asked. Thus, their ML-powered ICA can be interpreted as a form of combinatorial *Bayesian optimization (BO)* (see Appendix C).

In several follow-up papers, this work has been extended by developing more sophisticated ML methods for this problem. Weissteiner and Seuken (2020) integrated *neural networks (NN)* in their ICA and further increased efficiency. Weissteiner et al. (2022b) used *Fourier transforms (FTs)* to leverage different notions of sparsity of value functions. Finally, Weissteiner et al. (2022a) achieved state-of-the-art (SOTA) performance using *monotone-value NNs (MVNNs)*, which incorporate important prior domain knowledge.

The main shortcoming of this prior work is that all of these approaches are *myopic* in the sense that the resulting mechanisms simply query the allocation with the highest predicted welfare. In particular, the mechanisms do not have any model of *uncertainty* over bidders' values for not yet elicited bundles, although handling uncertainty in a principled manner is one of the key requirements of a smart PE algorithm (Bonilla, Guo, and Sanner 2010). Thus, the mechanisms cannot properly control the *exploration-exploitation trade-off* inherent to BO. For ML-based ICAs, this means that the mechanisms may get stuck in local minima, repeatedly querying one part of the allocation space while not exploring other, potentially more efficient allocations.

1.3 Our Contributions

In this paper, we address this main shortcoming of prior work and show how to integrate a notion of *model uncertainty* (i.e., epistemic uncertainty) over agents' preferences into iterative combinatorial assignment. Concretely, we design a *Bayesian optimization-based combinatorial assignment (BOCA)*[1] mechanism that makes use of model uncertainty in its query generation module. The main technical challenge is to design a new method for estimating an *upper uncertainty bound* that can be used to define an acquisition function to determine the next query. For this we combine *MVNNs* (Weissteiner et al. 2022a) with *neural optimization-based model uncertainty (NOMU)* (Heiss et al. 2022), a recently introduced method to estimate model uncertainty for NNs. In detail, we make the following contributions:

1. We present a modified NOMU algorithm (Section 3.1), tailored to CAs, exploiting monotonicity of agents' preferences and the discrete (finite) nature of this setting.
2. We show that generic parameter initialization for monotone NNs can dramatically fail and propose a new initialization method for MVNNs based on uniform mixture distributions (Section 3.2).
3. We present a more succinct mixed integer linear program for MVNNs to solve the ML-based WDP (Section 3.3).
4. We experimentally show that BOCA outperforms prior approaches in terms of efficiency (Section 4).

Although our contribution applies to any combinatorial assignment setting, we focus on CAs to simplify the notation and because there exist well-studied preference generators for CAs that we use for our experiments.

Our source code is publicly available on GitHub via: https://github.com/marketdesignresearch/BOCA.

1.4 Related Work on Model Uncertainty

Estimating model uncertainty for NNs is an active area of research in AI and ML, with a plethora of new methods proposed every year. Classic methods can be broadly categorized into *(i) ensemble methods*: training multiple different NNs to estimate model uncertainty (Gal and Ghahramani 2016; Lakshminarayanan, Pritzel, and Blundell 2017; Wenzel et al. 2020) and *(ii) Bayesian NNs (BNNs)*: assuming a prior distribution over parameters and then estimating model uncertainty by approximating the intractable posterior (Graves 2011; Blundell et al. 2015; Hernández-Lobato and Adams 2015; Ober and Rasmussen 2019). However, for ML-based iterative combinatorial assignment, a key requirement is to be able to efficiently solve the ML-based WDP based on these uncertainty estimates. As there is no known computationally tractable method to perform combinatorial optimization over ensembles or BNNs, we cannot use these approaches for ML-based ICAs. In contrast, NOMU (Heiss et al. 2022) enables the computationally efficient optimization over its uncertainty predictions, which is why we use it as a building block for BOCA.

2 Preliminaries

In this section, we present our formal model (Section 2.1), review the ML-based ICA by Brero, Lubin, and Seuken (2021) (Section 2.2), briefly review Bayesian optimization (BO) (Section 2.3), and review *monotone-value neural networks (MVNNs)* by Weissteiner et al. (2022a) (Section 2.4) as well as *neural optimization-based model uncertainty (NOMU)* by Heiss et al. (2022) (Section 2.5).

2.1 Formal Model for ICAs

We consider a CA with n bidders and m indivisible items. Let $N = \{1, \ldots, n\}$ and $M = \{1, \ldots, m\}$ denote the set of bidders and items. We denote by $x \in \mathcal{X} = \{0, 1\}^m$ a bundle of items represented as an indicator vector, where $x_j = 1$ iff item $j \in M$ is contained in x. Bidders' true preferences over bundles are represented by their (private) value functions $v_i : \mathcal{X} \to \mathbb{R}_+,\ i \in N$, i.e., $v_i(x)$ represents bidder i's true value for bundle $x \in \mathcal{X}$.

By $a = (a_1, \ldots, a_n) \in \mathcal{X}^n$ we denote an allocation of bundles to bidders, where a_i is the bundle bidder i obtains. We denote the set of *feasible* allocations by $\mathcal{F} = \left\{a \in \mathcal{X}^n : \sum_{i \in N} a_{ij} \leq 1,\ \forall j \in M\right\}$. We let $p \in \mathbb{R}^n_+$ denote the bidders' payments. We assume that bidders have quasilinear utility functions u_i of the form $u_i(a, p) = v_i(a_i) - p_i$. This implies that the (true) *social welfare* $V(a)$ of an allocation a is equal to the sum of all bidders' values $\sum_{i \in N} v_i(a_i)$. We let $a^* \in \operatorname{argmax}_{a \in \mathcal{F}} V(a)$ denote a social-welfare maximizing, i.e., *efficient*, allocation. The *efficiency* of any allocation $a \in \mathcal{F}$ is determined as $V(a)/V(a^*)$.

[1]The acronym BOCA has also been used for a different method, namely for *Bayesian optimisation with continuous approximations* by Kandasamy et al. (2017).

An ICA *mechanism* defines how the bidders interact with the auctioneer and how the allocation and payments are determined. We denote a bidder's (possibly untruthful) reported value function by $\hat{v}_i : \mathcal{X} \to \mathbb{R}_+$. In this paper, we consider ICAs that ask bidders to iteratively report their values $\hat{v}_i(x)$ for bundles x selected by the mechanism. A finite set of reported bundle-value pairs of bidder i is denoted as $R_i = \left\{\left(x^{(l)}, \hat{v}_i(x^{(l)})\right)\right\}, x^{(l)} \in \mathcal{X}$. Let $R = (R_1, \ldots, R_n)$ be the tuple of reported bundle-value pairs obtained from all bidders. We define the *reported social welfare* of an allocation a given R as $\widehat{V}(a|R) := \sum_{i \in N:\, (a_i, \hat{v}_i(a_i)) \in R_i} \hat{v}_i(a_i)$, where $(a_i, \hat{v}_i(a_i)) \in R_i$ ensures that only values for reported bundles contribute. The ICA's optimal allocation $a^*_R \in \mathcal{F}$ and payments $p(R) \in \mathbb{R}^n_+$ are computed based on the elicited reports R *only*. More formally, $a^*_R \in \mathcal{F}$ given reports R is defined as

$$a^*_R \in \operatorname*{argmax}_{a \in \mathcal{F}} \widehat{V}(a|R). \tag{1}$$

As the auctioneer can only query a limited number of bundles $|R_i| \leq Q^{\max}$ (e.g., $Q^{\max} = 100$), an ICA needs a practically feasible and smart PE algorithm.

2.2 A Machine Learning-Powered ICA

We now provide a high-level review of the *machine learning-powered combinatorial auction (MLCA)* by Brero, Lubin, and Seuken (2021) (please see Appendix A for further details). MLCA proceeds in rounds until a maximum number of value queries per bidder $Q^{\max}$ is reached. In each round, for every bidder i, an ML model $\mathcal{A}_i$ is trained on the bidder's reports R_i to learn an approximation of bidders' value functions. Next, MLCA generates new value queries by computing the allocation with the highest predicted social welfare. Concretely, it computes $q^{\text{new}} = (q^{\text{new}}_i)^n_{i=1}$ with $q^{\text{new}}_i \in \mathcal{X} \setminus R_i$ by solving an ML-based WDP:

$$q^{\text{new}} \in \operatorname*{argmax}_{a \in \mathcal{F}} \sum_{i \in N} \mathcal{A}_i(a_i) \tag{2}$$

The idea is the following: if the $\mathcal{A}_i$'s are good surrogate models of the bidders' value functions, then the efficiency of q^{new} is likely to be high as well. Thus, in each round, bidders are providing value reports on bundles that are guaranteed to fit into a feasible allocation and that together are predicted to have high social welfare. Additionally, bidders are also allowed to submit "push-bids," enabling them to submit information to the auctioneer that they deem useful, even if they are not explicitly queried about it. At the end of each round, MLCA receives reports R^{new} from all bidders for the newly generated queries q^{new}, and updates the overall elicited reports R. When $Q^{\max}$ is reached, MLCA computes an allocation a^*_R that maximizes the *reported* social welfare (Equation (1)) and determines VCG payments $p(R)$ based on all reports (see Appendix Definition B.1).

Remark 1 (IR, No-Deficit, and Incentives of MLCA)**.** *Brero, Lubin, and Seuken (2021) showed that MLCA satisfies* individual rationality (IR) *and* no-deficit*, with* any *ML algorithm. They also studied MLCA's incentive properties; this is important, since manipulations may lower efficiency. Like all deployed ICAs (including the CCA), MLCA is not strategyproof. However, they argued that it has good incentives in practice; and given two additional assumptions, bidding truthfully is an ex-post Nash equilibrium. We present a detailed summary of their incentive analysis in Appendix B.*

2.3 Bayesian Optimization Background

In this section, we briefly review Bayesian optimization (BO). BO refers to a class of *machine learning-based gradient-free* optimization methods, which, for a given black-box objective function $f : X \to \mathbb{R}$, aim to solve

$$\max_{x \in X} f(x) \tag{3}$$

in an *iterative* manner. Specifically, given a budget of T queries (i.e., function evaluations of f), a BO algorithm generates queries $\{x^{(1)}, \ldots, x^{(T)}\}$ with the aim that

$$\max_{x \in \{x^{(1)}, \ldots, x^{(T)}\}} f(x) \approx \max_{x \in X} f(x). \tag{4}$$

In each BO step t, the algorithm selects a new input point $x^{(t)} \in X$ and observes a (potentially noisy) output

$$y^{(t)} = f(x^{(t)}) + \varepsilon^{(t)}, \tag{5}$$

where $\varepsilon^{(t)}$ is typically assumed to be i.i.d. Gaussian, i.e., $\varepsilon^{(t)} \sim \mathcal{N}(0, \sigma^2)$.[2] The BO algorithm's decision rule for selecting the query $x^{(t)}$ is based on

1. A *probabilistic model* representing an (approximate) posterior distribution over f (e.g., Gaussian processes, NOMU, ensembles, BNNs, etc.).
2. An *acquisition function* $\mathcal{A} : X \to \mathbb{R}$ that uses this probabilistic model to determine the next query $x^{(t)} \in \operatorname{argmax}_{x \in X} \mathcal{A}(x)$ by properly trading off *exploration* and *exploitation*. See Appendix C.3 for popular examples of acquisition functions including:
   - *Upper uncertainty bound* (uUB) (aka *upper confidence bound (UCB)*) (Srinivas et al. 2012)
   - *Expected improvement* (Frazier 2018, Section 4.1)
   - *Thompson sampling* (Chapelle and Li 2011)

Remark 2. *MLCA (Section 2.2) can be seen as a combinatorial BO algorithm with acquisition function $\mathcal{A}(a) := \sum_{i \in N} \mathcal{A}_i(a_i)$ (see Appendix C for a discussion).*

2.4 MVNNs: Monotone-Value Neural Networks

MVNNs (Weissteiner et al. 2022a) are a new class of NNs specifically designed to represent *monotone combinatorial* valuations. First, we reprint the definition of MVNNs and then discuss their desirable properties.

Definition 1 (MVNN, Weissteiner et al. (2022a))**.** *An MVNN $\mathcal{M}^\theta_i : \mathcal{X} \to \mathbb{R}_+$ for bidder $i \in N$ is defined as*

$$\mathcal{M}^\theta_i(x) := W^{i,K_i} \varphi_{0,t^{i,K_i-1}} \left(\ldots \varphi_{0,t^{i,1}}(W^{i,1}x + b^{i,1}) \ldots\right) \tag{6}$$

- $K_i + 1 \in \mathbb{N}$ *is the number of layers ($K_i - 1$ hidden layers),*

[2] In this paper, we assume that $\sigma^2 = 0$.

- $\{\varphi_{0,t^{i,k}}\}_{k=1}^{K_i-1}$ *are the MVNN-specific activation functions with cutoff* $t^{i,k} > 0$, *called* bounded ReLU (bReLU)*:*

$$\varphi_{0,t^{i,k}}(\cdot) := \min(t^{i,k}, \max(0, \cdot)) \tag{7}$$

- $W^i := (W^{i,k})_{k=1}^{K_i}$ *with* $W^{i,k} \geq 0$ *and* $b^i := (b^{i,k})_{k=1}^{K_i-1}$ *with* $b^{i,k} \leq 0$ *are the* non-negative *weights and* non-positive *biases of dimensions* $d^{i,k} \times d^{i,k-1}$ *and* $d^{i,k}$*, whose parameters are stored in* $\theta = (W^i, b^i)$.

MVNNs are particularly well suited for the design of combinatorial assignment mechanism for two reasons. First, MVNNs are *universal* in the set of monotone and normalized value functions (Weissteiner et al. 2022a, Theorem 1), i.e., *any* $\hat{v}_i : \mathcal{X} \to \mathbb{R}_+$ that satisfies the following two properties can be represented *exactly* as an MVNN $\mathcal{M}_i^\theta$:

1. **Monotonicity (M)** (*"additional items increase value"*): For $A, B \in 2^M$: if $A \subseteq B$ it holds that $\hat{v}_i(A) \leq \hat{v}_i(B)$
2. **Normalization (N)** (*"no value for empty bundle"*): $\hat{v}_i(\emptyset) = \hat{v}_i((0, \dots, 0)) := 0$,

Second, Weissteiner et al. (2022a) showed that an MVNN-based WDP, i.e., $\underset{a \in \mathcal{F}}{\operatorname{argmax}} \sum_{i \in N} \mathcal{M}_i^\theta(a_i)$, can be succinctly encoded as a MILP, which is key for the design of MVNN-based iterative combinatorial assignment mechanisms. Finally, Weissteiner et al. (2022a) experimentally showed that using MVNNs as $\mathcal{A}_i$ in MLCA leads to SOTA performance.

2.5 NOMU

Recently, Heiss et al. (2022) introduced a novel method to estimate model uncertainty for NNs: *neural optimization-based model uncertainty (NOMU)*. In contrast to other methods (e.g., ensembles), NOMU represents an *upper uncertainty bound (uUB)* as a *single* and *MILP-formalizable* NN. Thus, NOMU is particularly well suited for iterative combinatorial assignment, where uUB-based *winner determination problems (WDPs)* need to be solved hundreds of times to generate new informative queries. This, together with NOMU's strong performance in noiseless BO, is the reason why we build on it and define a modified NOMU algorithm tailored to iterative combinatorial assignment (Section 3.1).

3 Bayesian Optimization-Based ICA

In this section, we describe the design of our Bayesian optimization-based combinatorial assignment (BOCA) mechanism. While the design is general, we here present it for the CA setting, leading to a BO-based ICA. Recall that MLCA generates new value queries by solving the ML-based WDP $q^{\text{new}} \in \underset{a \in \mathcal{F}}{\operatorname{argmax}} \sum\limits_{i \in N} \mathcal{A}_i(a_i)$ (see Section 2.2). For the design of BOCA, we integrate a proper notion of uncertainty into MLCA by using a bidder-specific *upper uncertainty bound (uUB)*, taking the role of the ML model $\mathcal{A}_i$, to define our acquisition function $\mathcal{A}(a) := \sum_{i \in N} \mathcal{A}_i(a_i)$. To define our uUB and make it amenable to MLCA, we proceed in three steps: First, we combine MVNNs with a modified NOMU algorithm that is tailored to the characteristics of combinatorial assignment (Section 3.1). Second, we highlight the importance of proper parameter initialization for MVNNs and propose a more robust method (Section 3.2). Third, we present a more succinct MILP for MVNNs (Section 3.3). In the remainder of the paper, we make the following assumption:

Assumption 1. *For all agents* $i \in N$*, the true and reported value functions* v_i *and* $\hat{v}_i$ *fulfill the* ***Monotonicity (M)*** *and* ***Normalization (N)*** *property (see Section 2.4).*

3.1 Model Uncertainty for Monotone NNs

We propose a modified NOMU architecture and loss that is specifically tailored to combinatorial assignment. Concretely, our algorithm is based on the following two key characteristics of combinatorial assignment: (i) since agents' value functions are monotonically increasing, the uUBs need to be monotonically increasing too, and (ii) due to the (finite) discrete input space, one can derive a closed-form expression of the 100%-uUB as an MVNN. Before we present our modified NOMU architecture and loss, we introduce the MVNN-based 100%-uUB.

Let $\mathcal{H}$ denote a *hypothesis class* of functions $f : X \to \mathbb{R}$ for some input space X and let $\mathcal{H}_{D^{\text{train}}} := \{f \in \mathcal{H} : f(x^{(l)}) = y^{(l)}, l = 1, \dots, n^{\text{train}}\}$ denote the set of all functions from $\mathcal{H}$ that fit exactly through training points $D^{\text{train}} = \left\{\left(x^{(l)}, f(x^{(l)})\right)\right\}_{l=1}^{n^{\text{train}}}$.

Definition 2 (100%-uUB)**.** *For a hypothesis class* $\mathcal{H}$ *and a training set* D^{train}*, we define the 100%-uUB as* $f^{\text{100\%-uUB}}(x) := \sup_{f \in \mathcal{H}_{D^{train}}} f(x)$ *for every* $x \in X$.

In the following, let

$$\mathcal{V} := \{\hat{v} : \mathcal{X} \to \mathbb{R}_+ | \text{ satisfy } \textbf{(N)} \text{ and } \textbf{(M)}\} \tag{8}$$

denote the set of all value functions that satisfy the *normalization* and *monotonicity* property. Next, we define the 100%-uUB. In Theorem 1, we show that for $\mathcal{H} = \mathcal{V}$ the 100%-uUB can be explicitly represented as an MVNN.

Theorem 1 (MVNN-based 100%-uUB)**.** *Let* $((1, \dots, 1), \hat{v}_i(1, \dots, 1)) \in D^{train}$*. Then for* $\mathcal{H} = \mathcal{V}$ *it holds that* $f^{\text{100\%-uUB}}(x) = \max_{f \in \mathcal{V}_{D^{train}}} f(x)$ *for all* $x \in \mathcal{X}$ *and* $f^{\text{100\%-uUB}} \in \mathcal{V}_{D^{train}}$ *can be represented as a two hidden layer MVNN with* n^{train} *neurons per layer, which we denote as* $\mathcal{M}_i^{\text{100\%-uUB}}$ *going forward.*[3]

Proof. The proof for Theorem 1 is provided in Appendix D.1. It follows a similar idea as the universality proof in (Weissteiner et al. 2022a, Theorem 1). In particular, Equation (27) in Appendix D.1 provides the closed-form expression of $f^{\text{100\%-uUB}}$ as MVNN $\mathcal{M}_i^{\text{100\%-uUB}}$. □

Using the MVNN-based 100%-uUB $\mathcal{M}_i^{\text{100\%-uUB}}$, we can now define our modified NOMU architecture and loss.

The Architecture. Towards defining the architecture, we first observe that if the true function is monotonically increasing, the corresponding uUB needs to be monotonically increasing as well (Proposition 1 and 2 in Appendix D.2).

[3]Note that $\mathcal{M}_i^{\text{100\%-uUB}}(\cdot)$ depends on a training set D^{train}, but we omit this dependency in our notation to improve readability.

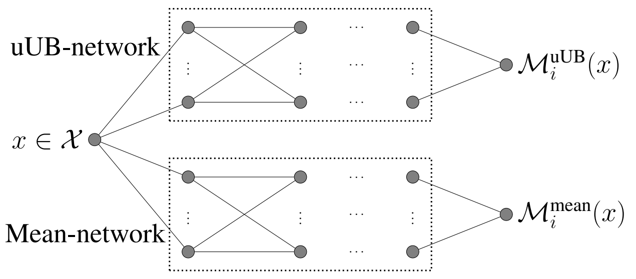
uUB-network
$x \in \mathcal{X}$
Mean-network
$\mathcal{M}_i^{\text{uUB}}(x)$
$\mathcal{M}_i^{\text{mean}}(x)$

Figure 1: $\mathcal{M}_i^{\text{NOMU}}$: a modification of NOMU's original architecture for the combinatorial assignment domain.

Given that bidders' value functions are monotone (Assumption 1), this implies that our uUB must also be monotonically increasing. Thus, instead of the original NOMU architecture that outputs the (raw) uncertainty (i.e., an estimate of the posterior standard deviation) which is *not* monotone, we can modify NOMU's architecture and directly output the monotone uUB. Given this, we propose the following architecture $\mathcal{M}_i^{\text{NOMU}}$ to estimate the uUB for bidder $i \in N$. $\mathcal{M}_i^{\text{NOMU}}$ consists of two sub-MVNNs with two outputs: the mean prediction $\mathcal{M}_i^{\text{mean}} : \mathcal{X} \to \mathbb{R}$ and the estimated uUB $\mathcal{M}_i^{\text{uUB}} : \mathcal{X} \to \mathbb{R}$. In Figure 1, we provide a schematic representation of $\mathcal{M}_i^{\text{NOMU}}$ (see Appendix D.2 for details).

The Loss. Next, we formulate a new NOMU loss function L^π tailored to combinatorial assignment. Since we have a closed-form expression of the 100%-uUB as MVNN $\mathcal{M}_i^{\text{100\%-uUB}}$ (Theorem 1), we are able to enforce that $\mathcal{M}_i^{\text{mean}} \leq \mathcal{M}_i^{\text{uUB}} \leq \mathcal{M}_i^{\text{100\%-uUB}}$ via the design of our new loss function. Let $\mathcal{M}_i^{\text{mean}}$ be a trained mean-MVNN with a standard loss (e.g., MAE and L2-regularization). Using $\mathcal{M}_i^{\text{mean}}$ and $\mathcal{M}_i^{\text{100\%-uUB}}$, we then only train the parameters θ of $\mathcal{M}_i^{\text{uUB}}$ with loss L^π and L2-regularization parameter $\lambda > 0$, i.e., minimizing $L^\pi(\mathcal{M}_i^{\text{uUB}}) + \lambda \|\theta\|_2^2$ via gradient descent. In particular, the parameters of $\mathcal{M}_i^{\text{100\%-uUB}}$ and $\mathcal{M}_i^{\text{mean}}$ are not influenced by the training of $\mathcal{M}_i^{\text{uUB}}$ (see Appendix D.3 for details on the loss and training procedure).

Definition 3 (NOMU Loss Tailored to Combinatorial Assignment)**.** *Let* $\pi = (\pi_{sqr}, \pi_{exp}, c_{exp}, \overline{\pi}, \underline{\pi}) \in \mathbb{R}_+^5$ *be a tuple of hyperparameters and let* $s(\mathcal{M}_i^{uUB}, x) := \min\{\mathcal{M}_i^{uUB}(x), \mathcal{M}_i^{100\%\text{-}uUB}(x)\} - \mathcal{M}_i^{mean}(x)$ *for all* $x \in \mathcal{X}$*. For a training set* D^{train}*,* L^π *is defined as*

$$L^\pi(\mathcal{M}_i^{uUB}) := \pi_{sqr} \sum_{l=1}^{n^{train}} L_1^\beta\left(\mathcal{M}_i^{uUB}(x^{(l)}), y^{(l)}\right) \tag{9a}$$

$$+ \pi_{exp} \int_{[0,1]^m} g\left(-c_{exp} s(\mathcal{M}_i^{uUB}, x)\right) \, dx \tag{9b}$$

$$+ \pi_{exp} c_{exp} \overline{\pi} \int_{[0,1]^m} L_1^\beta\left((\mathcal{M}_i^{uUB}(x) - \mathcal{M}_i^{100\%\text{-}uUB}(x))^+\right) \, dx \tag{9c}$$

$$+ \pi_{exp} c_{exp} \underline{\pi} \int_{[0,1]^m} L_1^\beta\left((\mathcal{M}_i^{mean}(x) - \mathcal{M}_i^{uUB}(x))^+\right) \, dx \, , \tag{9d}$$

where L_1^β *is the smooth L1-loss with threshold* β *(see Appendix Definition D.1),* $(\cdot)^+$ *the positive part, and* $g := 1 + ELU$[4] *is convex monotonically increasing with ELU being the* exponential linear unit *(see Appendix Definition D.2).*

The interpretations of the four terms are as follows:

(9a) enforces that $\mathcal{M}_i^{\text{uUB}}$ fits through the training data.

(9b) pushes $\mathcal{M}_i^{\text{uUB}}$ up as long as it is below the 100%-uUB $\mathcal{M}_i^{\text{100\%-uUB}}$. This force gets weaker the further $\mathcal{M}_i^{\text{uUB}}$ is above the mean $\mathcal{M}_i^{\text{mean}}$ (especially if c_{exp} is large). π_{exp} controls the overall strength of (9b) and c_{exp} controls how fast this force increases when $\mathcal{M}_i^{\text{uUB}} \to \mathcal{M}_i^{\text{mean}}$. Thus, increasing π_{exp} increases the uUB and increasing c_{exp} increases the uUBs in regions where it is close to $\mathcal{M}_i^{\text{mean}}$. Weakening (9b) (i.e., $\pi_{\text{exp}} c_{\text{exp}} \to 0$) leads to $\mathcal{M}_i^{\text{uUB}} \approx \mathcal{M}_i^{\text{mean}}$. Strengthening (9b) by increasing $\pi_{\text{exp}} c_{\text{exp}}$ in relation to regularization[5] leads to $\mathcal{M}_i^{\text{uUB}} \approx \mathcal{M}_i^{\text{100\%-uUB}}$.

(9c) enforces that $\mathcal{M}_i^{\text{uUB}} \leq \mathcal{M}_i^{\text{100\%-uUB}}$. The strength of this term is determined by $\overline{\pi} \cdot (\pi_{\text{exp}} c_{\text{exp}})$, where $\overline{\pi}$ is the (9c)-specific hyperparameter and $\pi_{\text{exp}} c_{\text{exp}}$ adjusts the strength of (9c) to (9b).

(9d) enforces $\mathcal{M}_i^{\text{uUB}} \geq \mathcal{M}_i^{\text{mean}}$. The interpretation of $\underline{\pi}$ and $\pi_{\text{exp}} c_{\text{exp}}$ is analogous to (9c).

As in (Heiss et al. 2022), in the implementation of L^π, we approximate Equations (9b) to (9d) via Monte Carlo integration using additional, *artificial input points* $D^{\text{art}} := \left\{x^{(l)}\right\}_{l=1}^{n^{\text{art}}} \overset{i.i.d.}{\sim} \text{Unif}([0,1]^m)$.

Visualization of the uUB. In Figure 2, we present a visualization of the output of $\mathcal{M}_i^{\text{NOMU}}$ (i.e., $\mathcal{M}_i^{\text{mean}}$ and $\mathcal{M}_i^{\text{uUB}}$) and $\mathcal{M}_i^{\text{100\%-uUB}}$ for the national bidder in the LSVM domain of the spectrum auction test suite (SATS) (Weiss, Lubin, and Seuken 2017). In noiseless regression, uncertainty should vanish at observed training points, but (model) uncertainty should remain about value predictions for bundles that are very different from the bundles observed in training. Figure 2 shows that our uUB $\mathcal{M}_i^{\text{uUB}}$ nicely fulfills this. Moreover, we have shown in Appendix D.2 that $\mathcal{M}_i^{\text{uUB}}$ is monotonically increasing, since we assume that value functions fulfill the monotonicity property. This implies that once we observe a value for the full bundle, we obtain a globally bounded 100%-uUB, i.e., see $\mathcal{M}_i^{\text{100\%-uUB}}$ in Figure 2. Furthermore, we see that $\mathcal{M}_i^{\text{100\%-uUB}}$ jumps to a high value when only a single item is added to an already queried bundle, but then often stays constant (e.g., $|x| = 12, \ldots, 18$ in Figure 2). Thus, using such a 100%-uUB in our acquisition function, BOCA would only add a single item to an already queried bundle to have more items left for the other bidders instead of properly exploring the bundle space. Our uUB $\mathcal{M}_i^{\text{uUB}}$ circumvents this via implicit and explicit regularization and yields a useful uUB.

[4]In our notation, $g(\cdot)$ is the analog of the function $e^{(\cdot)}$ used in the original NOMU loss in (Heiss et al. 2022).

[5]Regularization can be early stopping or a small number of neurons (implicit) or L2-regularization on the parameters (explicit).

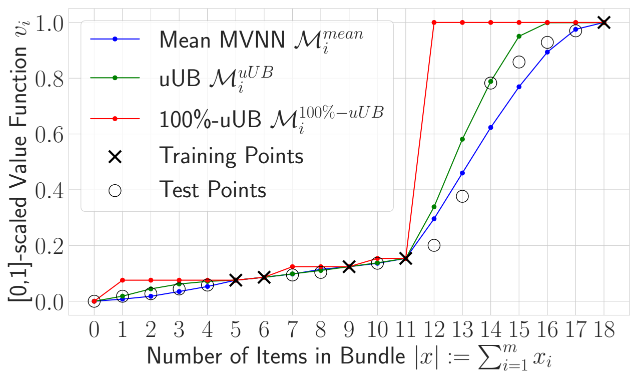

Figure 2: $\mathcal{M}_i^{\text{mean}}$, $\mathcal{M}_i^{\text{uUB}}$ and $\mathcal{M}_i^{\text{100\%-uUB}}$ along an increasing 1D subset-path (i.e., for all bundles $x^{(j)}, x^{(k)}$ on the x-axis it holds that for $j \leq k : x^{(j)} \subset x^{(k)}$).

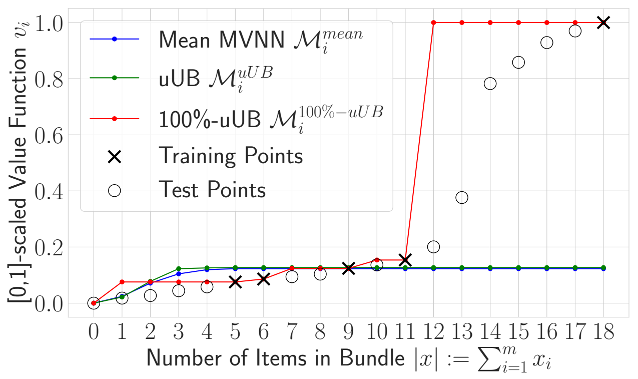

Figure 3: In contrast to our proposed initialization (see Figure 2), training fails with generic initialization already for relatively small [64,64]-architectures that were used here.

3.2 Parameter Initialization for MVNNs

We now discuss how to properly initialize parameters for MVNNs. Importantly, the *MVNN-based uUBs* $\mathcal{M}_i^{\text{uUB}}$ are MVNNs. As we will show next, to achieve the best performance of BOCA (in fact of any MVNN training), an adjusted, non-generic parameter initialization is important.

Generic Initialization. For standard NNs, it is most common to use a parameter initialization with zero mean $\mu_k := \mathbb{E}\left[W_{j,l}^{i,k}\right] = 0$ and non-zero variance $\sigma_k^2 := \mathbb{V}\left[W_{j,l}^{i,k}\right] \neq 0$. Then the mean of each pre-activated neuron of the first hidden layer is zero and the variance $\mathbb{V}\left[\left(W^{i,1}x\right)_j\right] = d^{i,0}\sigma_1^2\overline{x^2}$, if $\left(W_{j,l}^{i,1}\right)_{l=1}^{d^{i,0}}$ are i.i.d., where $\overline{x^2} = \frac{1}{d^{i,0}}\sum_{l=1}^{d^{i,0}} x_l^2$.[6] Analogously, one can compute the *conditional* mean and the *conditional* variance of a pre-activated neuron in any layer k by replacing x by the output $z^{i,k-1}$ of the previous layer, i.e., $\mathbb{E}\left[\left(W^{i,k}z^{i,k-1}\right)_j \middle| z^{i,k-1}\right] = 0$ and $\mathbb{V}\left[\left(W^{i,k}z^{i,k-1}\right)_j \middle| z^{i,k-1}\right] = d^{i,k-1}\sigma_k^2\overline{(z^{i,k-1})^2}$. For $\sigma_k \propto \frac{1}{\sqrt{d^{i,k-1}}}$, the conditional mean and variance do not depend on the layer dimensions $d^{i,k}$, which is why generic initialization methods scale the initial distribution by $s_k \propto \frac{1}{\sqrt{d^{i,k-1}}}$.

Problem. Unfortunately, this generic initialization approach can dramatically fail for MVNNs: For any non-zero initialization, the non-negativity constraint of the weights implies that the mean $\mu_k > 0$. This implies that the mean of a pre-activated neuron in the first hidden layer is $\mathbb{E}\left[\left(W^{i,1}x\right)_j\right] = d^{i,0}\mu_1\bar{x}$. For a generic scaling s_k one would obtain $\mu_k \propto \frac{1}{\sqrt{d^{i,k-1}}}$ and thus the mean $\mathbb{E}\left[\left(W^{i,1}x\right)_j\right] \propto d^{i,0}\frac{1}{\sqrt{d^{i,0}}}\bar{x} = \sqrt{d^{i,0}}\bar{x}$ of the pre-activated neurons diverges to infinity with a rate of $\sqrt{d^{i,0}}$ as $d^{i,0} \to \infty$. Analogously, the pre-activated neurons of every layer diverge to infinity as $d^{i,k-1} \to \infty$. This is particularly problematic for bReLUs (as used in MVNNs) as their gradient is zero on $[0, t^{i,k}]^c$. Figure 3 shows that both MVNNs $\mathcal{M}_i^{\text{mean}}$ and $\mathcal{M}_i^{\text{uUB}}$ get "stuck." This happens because already at initialization, every neuron in the first hidden layer has a pre-activation that is larger than $t^{i,1}$ for every training point.

This could be solved by scaling down the initial weights even more, e.g., $W_{j,l}^{i,k} \sim \text{Unif}[0, \frac{2}{d^{i,k-1}}]$ resulting in $\mu_k = \frac{1}{d^{i,k-1}}$. However, since for $W_{j,l}^{i,k} \sim \text{Unif}[0, \frac{2}{d^{i,k-1}}]$ it holds that $\sigma_k^2 \propto \frac{1}{(d^{i,k-1})^2}$, this induces a new problem of vanishing conditional variance $\mathbb{V}\left[\left(W^{i,k}z^{i,k-1}\right)_j \middle| z^{i,k-1}\right]$ with a rate of $\mathcal{O}(\frac{1}{d^{i,k-1}})$ for wide (i.e., $d^{i,k-1}$ large) MVNNs. Overall, it is impossible to simultaneously solve both problems by just scaling the distribution by a factor s_k, because the conditional mean $\mathbb{E}\left[\left(W^{i,k}z^{i,k-1}\right)_j \middle| z^{i,k-1}\right]$ scales with $s_k \cdot d^{i,k-1}$ and the conditional variance $\mathbb{V}\left[\left(W^{i,k}z^{i,k-1}\right)_j \middle| z^{i,k-1}\right]$ scales with $s_k^2 \cdot d^{i,k-1}$. Thus, for wide MVNNs, one of those two problems (i.e., either diverging expectation or vanishing variance) would persist.

Solution. We introduce a new initialization method that solves *both* problems at the same time. For this, we propose a mixture distribution of two different uniform distributions (see Appendix Definition E.1). For each layer k, we independently sample all weights $W_{jl}^{i,k}$ i.i.d. with probability $(1 - p_k)$ from Unif$[0, A_k]$, and with probability p_k from Unif$[0, B_k]$. If we choose p_k and A_k small enough, we can get arbitrarily small μ_k while not reducing σ_k too much. In Appendix E, we provide formulas for how to choose A_k, B_k and p_k depending on $d^{i,k-1}$. In Theorem 3 in Appendix E, we prove that, if the parameters are chosen in this way, then the conditional mean and conditional variance neither explode nor vanish with increasing $d^{i,k-1}$ but rather stay con-

[6]We assume that the biases $b^{i,k} = 0$ are all initialized to zero throughout Section 3.2 to keep the notation simpler, while we formulate everything for the general case including random biases in Appendix E and in our code.

stant for large $d^{i,k-1}$. Note that, in Figure 2, for $\mathcal{M}_i^{\text{mean}}$ and $\mathcal{M}_i^{\text{uUB}}$, we used our proposed initialization method for suitable A_k, B_k and p_k, such that the problem induced by a generic initialization from Figure 3 is resolved.

3.3 Mixed Integer Linear Program (MILP)

A key step in ML-powered iterative combinatorial assignment mechanisms is finding the (predicted) social welfare-maximizing allocation, i.e., solving the *ML-based WDP*. Thus, a key requirement posed on any acquisition function $\mathcal{A}$ in such a mechanism is to be able to efficiently solve $\max_{a\in\mathcal{F}} \mathcal{A}(a)$. Recall that, to define our acquisition function $\mathcal{A}$, we use $\mathcal{A}(a) = \sum_{i\in N} \mathcal{A}_i(a_i)$ where the $\mathcal{A}_i$'s are bidder-specific upper uncertainty bounds. Thus, the ML-based WDP becomes

$$\max_{a\in\mathcal{F}} \sum_{i\in N} \mathcal{A}_i(a_i). \tag{10}$$

Weissteiner et al. (2022a) proposed a MILP for MVNNs with $\mathcal{A}_i := \mathcal{M}_i^\theta$ to efficiently solve eq. (10). Their MILP was based on a reformulation of the $\min(\cdot,\cdot)$ and $\max(\cdot,\cdot)$ in the bReLU activation $\min(\max(\cdot, 0), t)$. Thus, it required twice the number of binary variables *and* linear constraints as for a plain ReLU-NN. Since we use an MVNN-based uUB $\mathcal{A}_i := \mathcal{M}_i^{\text{uUB}}$ to define our acquisition function, we could directly use their MILP formulation. However, instead, we propose a new MILP, which is significantly more succinct. For this, let $o^{i,k} := W^{i,k}z^{i,k-1} + b^{i,k}$ be the *pre*-activated output and $z^{i,k} := \varphi_{0,t^{i,k}}(o^{i,k})$ be the output of the kth layer with $l^{i,k} \leq o^{i,k} \leq u^{i,k}$, where the tight lower (upper) bound $l^{i,k}$ ($u^{i,k}$) is derived by forward-propagating the empty (full) bundle (Weissteiner et al. 2022a, Fact 1). In Theorem 2, we state our new MILP (see Appendix F.1 for the proof).[7]

Theorem 2 (MVNN MILP Tailored to Combinatorial Assignment)**.** *Let $\mathcal{A}_i = \mathcal{M}_i^{uUB}$ be our MVNN-based uUBs. The ML-based WDP* (10) *can be formulated as the following MILP:*

$$\max_{a\in\mathcal{F}, z^{i,k}, \alpha^{i,k}, \beta^{i,k}} \left\{ \sum_{i\in N} W^{i,K_i} z^{i,K_i-1} \right\} \tag{11}$$

s.t. for $i \in N$ and $k \in \{1, \ldots, K_i - 1\}$

$$z^{i,0} = a_i \tag{12}$$

$$z^{i,k} \leq \alpha^{i,k} \cdot t^{i,k} \tag{13}$$

$$z^{i,k} \leq o^{i,k} - l^{i,k} \cdot (1 - \alpha^{i,k}) \tag{14}$$

$$z^{i,k} \geq \beta^{i,k} \cdot t^{i,k} \tag{15}$$

$$z^{i,k} \geq o^{i,k} + (t^{i,k} - u)\beta^{i,k} \tag{16}$$

$$\alpha^{i,k} \in \{0,1\}^{d^{i,k}}, \ \beta^{i,k} \in \{0,1\}^{d^{i,k}} \tag{17}$$

Note that for each neuron of $\mathcal{A}_i = \mathcal{M}_i^{\text{uUB}}$, our new MILP has only 4 linear constraints, i.e., respective components of eqs. (13) to (16), compared to 8 in (Weissteiner et al. 2022a). Moreover, in contrast to the MILP in (Weissteiner et al. 2022a), our MILP does not make use of any "big-M" constraints, which are known to be numerically unstable.

[7]All vector inequalities should be understood component-wise.

4 Experiments

In this section, we experimentally evaluate the performance of BOCA in CAs. To this end, we equip the MLCA mechanism (see Section 2.2) with our new acquisition function $\mathcal{A}(a) = \sum_{i\in N} \mathcal{M}_i^{\text{uUB}}(a_i)$. We compare the efficiency of BOCA against the previously proposed MVNN-based and NN-based MLCA from (Weissteiner et al. 2022a) which do not explicitly model the mechanism's uncertainty over values for not yet elicited bundles.[8] We use our new parameter initialization method (Section 3.2) for $\mathcal{M}_i^{\text{uUB}}$, and we use our new MILP (Theorem 2) for solving the WDPs.

Experiment Setup. To generate synthetic CA instances, we use the following three domains from the spectrum auction test suite (SATS) (Weiss, Lubin, and Seuken 2017): LSVM, SRVM, and MRVM (see Appendix G.1 for details).[9] SATS gives us access to the true optimal allocation a^*, which we use to measure the *efficiency loss*, i.e., $1 - V(a^*_R)/V(a^*)$ when eliciting reports R via MLCA. We report efficiency loss (and not revenue), as spectrum auctions are government-run, with a mandate to maximize welfare (Cramton 2013). See Appendix G.6 for a discussion of the corresponding results on revenue. To enable a fair comparison against prior work, for each domain, we use $Q^{\text{init}} = 40$ initial random queries (including the full bundle for the calculation of $\mathcal{M}_i^{100\%\text{-uUB}}$) and set the query budget to $Q^{\max} = 100$ (see Appendix G.8 for results for $Q^{\text{init}} = 20$). We terminate any mechanism in an intermediate iteration if it already found an allocation with 0% efficiency loss.

Hyperparameter Optimization (HPO). We use *random search (RS)* (Bergstra and Bengio 2012) to optimize the hyperparameters of the mean MVNN $\mathcal{M}_i^{\text{mean}}$ and of our MVNN-based uUB $\mathcal{M}_i^{\text{uUB}}$. The HPO includes the NN-architecture parameters, training parameters, NOMU parameters, and initialization parameters (see Section 3.2). RS was carried out independently for each bidder type and SATS domain with a budget of 500 configurations, where each configuration was evaluated on 100 SATS instances. For each instance, the MVNNs $\mathcal{M}_i^{\text{mean}}$ and $\mathcal{M}_i^{\text{uUB}}$ were trained on uniformly at random chosen bundle-value pairs D^{train} and evaluated on a disjoint test set of different bundle-value pairs D^{test}. To select the winner configuration, we consider as evaluation metric the quantile-loss on the test set *and* the MAE on the training set, i.e., for each configuration and instance we calculate

$$\frac{1}{|D^{\text{test}}|} \sum_{(x,y)\in D^{\text{test}}} \max\{(y - \mathcal{M}_i^{\text{uUB}}(x))q, (\mathcal{M}_i^{\text{uUB}}(x) - y)(1-q)\}$$
$$+ \text{MAE}(D^{\text{train}}), \tag{18}$$

which we then average over all 100 instances. We used four quantile parameters $q \in \{0.6, 0.75, 0.9, 0.95\}$ in eq. (18)

[8]In these methods, uncertainty over not yet elicited bundles is only modeled via the retraining of the (MV)NNs in each round, i.e., the random parameter initialization of the (MV)NNs. This can be seen as simple form of Thompson sampling (see last paragraph in Appendix C).

[9]We do not use GSVM, as Weissteiner et al. (2022a) already achieved 0% efficiency loss in GSVM via MVNN-based MLCA.

Domain	$Q^{\max}$	Efficiency Loss in % ↓					T-Test for Efficiency:	
		BOCA	MVNN-MLCA	NN-MLCA	FT-MLCA	RS	$\mathcal{H}_0 : \mu_{\text{MVNN-MLCA}} \leq \mu_{\text{BOCA}}$	$\mathcal{H}_0 : \mu_{\text{NN-MLCA}} \leq \mu_{\text{BOCA}}$
LSVM	100	0.39±0.30	00.70±0.40	02.91±1.44	01.54 ±0.65	31.73±2.15	$p_{\text{VAL}} = 9\text{e}{-2}$	$p_{\text{VAL}} = 3\text{e}{-4}$
SRVM	100	0.06±0.02	00.23±0.06	01.13±0.22	00.72±0.16	28.56±1.74	$p_{\text{VAL}} = 5\text{e}{-6}$	$p_{\text{VAL}} = 2\text{e}{-13}$
MRVM	100	7.77±0.34	08.16±0.41	09.05±0.53	10.37±0.57	48.79±1.13	$p_{\text{VAL}} = 8\text{e}{-2}$	$p_{\text{VAL}} = 2\text{e}{-5}$

Table 1: BOCA vs MVNN-MLCA, NN-MLCA, Fourier transform (FT)-MLCA and random search (RS). Shown are averages and a 95% CI on a test set of 50 instances. Winners based on a t-test with significance level of 1% are marked in grey.

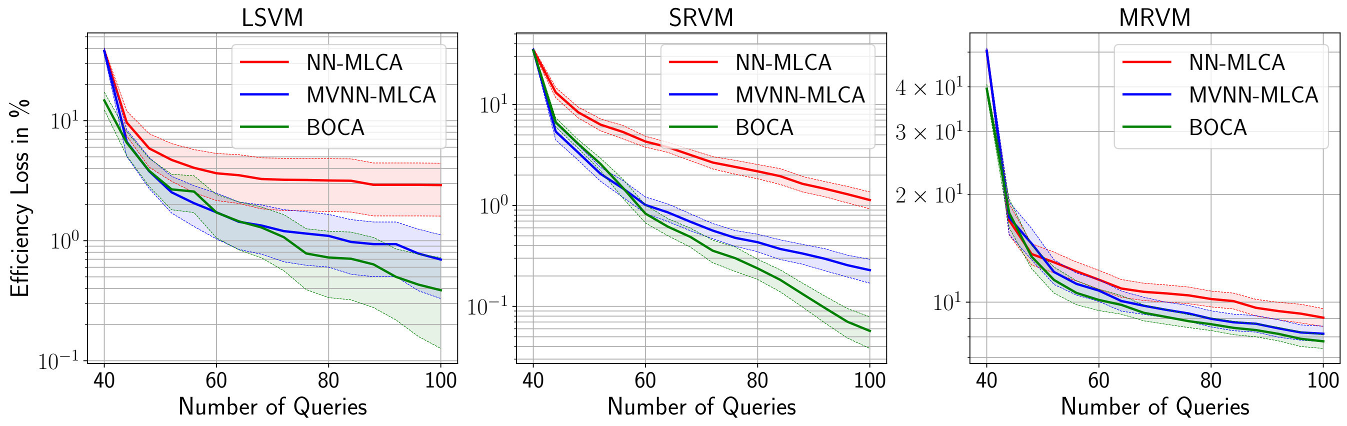

Figure 4: Efficiency loss paths (i.e., regret plots) of BOCA compared to the results from Weissteiner et al. (2022a) of MVNN-MLCA and NN-MLCA without any notion of uncertainty. Shown are averages with 95% CIs over 50 CA instances.

to achieve different levels of exploration (i.e., the resulting uUBs become larger the more we increase q in eq. (18)). This evaluation metric *simultaneously* measures the quality of the uUB on the test data (via the quantile-loss) as well as the quality of the uUB predictions on the training data (via the MAE). For each quantile q and SATS domain, we then proceed with the winner configuration of $\mathcal{M}_i^{\text{uUB}}$ and evaluate the efficiency of BOCA on a separate set of 50 instances. Details on hyperparameter ranges and the training procedure are provided in Appendices G.2 and G.3.

Results. In Table 1, we show the average efficiency loss of each approach after $Q^{\max} = 100$ queries (see Appendix G.5 for details). We see that BOCA significantly outperforms MVNN-MLCA (Weissteiner et al. 2022a) in SRVM, and it performs on-par in LSVM and MRVM, with a better average performance. Since MVNNs previously achieved SOTA performance, BOCA also outperforms the other benchmarks (i.e., NN (Weissteiner and Seuken 2020) and FT-MLCA (Weissteiner et al. 2022b)). RS's poor performance highlights the intrinsic difficulty of this task. The amount of exploration needed is domain dependent (e.g., multi-modality of the objective), which explains why the significance of BOCA's improvement varies across domains. However, our results also show that using an uUB (as in BOCA) instead of just a mean prediction (as in MVNN-MLCA) never hurts.

Figure 4 shows the efficiency loss path for all domains. We see that the superior (average) performance of $\mathcal{M}_i^{\text{uUB}}$ does not only hold at the end of the auction (at $Q^{\max} = 100$), but also for a large range of queries: in LSVM, BOCA is better for [70,100]; in SRVM, BOCA is significantly better for [70,100]; in MRVM, BOCA is better for [50,100]. See Appendix G.6 for results on revenue where BOCA significantly outperforms MVNN-MLCA also for MRVM. In Appendix G.7, we study to what degree BOCA's performance increase is due to (a) our uncertainty model (Section 3.1) versus (b) our new parameter initialization method (Section 3.2). Finally, in Appendix G.8, we provide further experiments for a reduced number of $Q^{\text{init}} = 20$ initial queries, which lead to similar results as shown in Table 1.

5 Conclusion

In this paper, we have proposed a Bayesian optimization-based combinatorial assignment (BOCA) mechanism. On a conceptual level, our main contribution was the integration of model uncertainty over agents' preferences into ML-based preference elicitation. On a technical level, we have designed a new method for estimating an upper uncertainty bound that exploits the monotonicity of agents' preferences in the combinatorial assignment domain and the finite nature of this setting. Our experiments have shown that BOCA performs as good or better than the SOTA in terms of efficiency. An interesting direction for future work is the evaluation of BOCA in other combinatorial assignment domains, such as combinatorial exchanges or course allocation (e.g., see (Soumalias et al. 2023)). Finally, it would also be interesting to apply BOCA's conceptual idea in the combinatorial BO settings outside of combinatorial assignment.

Acknowledgments

We thank the anonymous reviewers for helpful comments. This paper is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant agreement No. 805542).

References

Ausubel, L.; and Cramton, P. 2011. Auction design for wind rights. *Report to Bureau of Ocean Energy Management, Regulation and Enforcement*.

Ausubel, L. M.; and Baranov, O. 2017. A practical guide to the combinatorial clock auction. *Economic Journal*, 127(605): F334–F350.

Ausubel, L. M.; Cramton, P.; and Milgrom, P. 2006. The clock-proxy auction: A practical combinatorial auction design. In Cramton, P.; Shoham, Y.; and Steinberg, R., eds., *Combinatorial Auctions*, 115–138. MIT Press.

Bergstra, J.; and Bengio, Y. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2).

Bichler, M.; Fux, V.; and Goeree, J. K. 2019. Designing combinatorial exchanges for the reallocation of resource rights. *Proceedings of the National Academy of Sciences*, 116(3): 786–791.

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight uncertainty in neural networks. In *32nd International Conference on Machine Learning (ICML)*.

Bonilla, E. V.; Guo, S.; and Sanner, S. 2010. Gaussian process preference elicitation. *Advances in neural information processing systems*, 23.

Brero, G.; Lubin, B.; and Seuken, S. 2018. Combinatorial Auctions via Machine Learning-based Preference Elicitation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*.

Brero, G.; Lubin, B.; and Seuken, S. 2021. Machine Learning-powered Iterative Combinatorial Auctions. *arXiv preprint arXiv:1911.08042*.

Budish, E. 2011. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy*, 119(6): 1061–1103.

Chapelle, O.; and Li, L. 2011. An Empirical Evaluation of Thompson Sampling. In *NIPS*.

Cramton, P. 2013. Spectrum auction design. *Review of Industrial Organization*, 42(2): 161–190.

Frazier, P. I. 2018. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*.

Gal, Y.; and Ghahramani, Z. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *33rd International Conference on Machine Learning (ICML)*, 1050–1059.

Graves, A. 2011. Practical variational inference for neural networks. In *Advances in neural information processing systems*, 2348–2356.

Heiss, J. M.; Weissteiner, J.; Wutte, H. S.; Seuken, S.; and Teichmann, J. 2022. NOMU: Neural Optimization-based Model Uncertainty. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 8708–8758. PMLR.

Hernández-Lobato, J. M.; and Adams, R. 2015. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, 1861–1869.

Kandasamy, K.; Dasarathy, G.; Schneider, J.; and Póczos, B. 2017. Multi-fidelity bayesian optimisation with continuous approximations. In *International Conference on Machine Learning*, 1799–1808. PMLR.

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, 6402–6413.

Nisan, N.; and Segal, I. 2006. The communication requirements of efficient allocations and supporting prices. *Journal of Economic Theory*, 129(1): 192–224.

Ober, S. W.; and Rasmussen, C. E. 2019. Benchmarking the neural linear model for regression. *arXiv preprint arXiv:1912.08416*.

Sandholm, T.; and Boutilier, C. 2006. Preference elicitation in combinatorial auctions. *Combinatorial auctions*, 10.

Soumalias, E.; Zamanlooy, B.; Weissteiner, J.; and Seuken, S. 2023. Machine Learning-powered Course Allocation. *arXiv preprint arXiv:2210.00954*.

Srinivas, N.; Krause, A.; Kakade, S. M.; and Seeger, M. W. 2012. Information-Theoretic Regret Bounds for Gaussian Process Optimization in the Bandit Setting. *IEEE Transactions on Information Theory*, 58(5): 3250–3265.

Weiss, M.; Lubin, B.; and Seuken, S. 2017. Sats: A universal spectrum auction test suite. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, 51–59.

Weissteiner, J.; Heiss, J.; Siems, J.; and Seuken, S. 2022a. Monotone-Value Neural Networks: Exploiting Preference Monotonicity in Combinatorial Assignment. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 541–548. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Weissteiner, J.; and Seuken, S. 2020. Deep Learning—Powered Iterative Combinatorial Auctions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02): 2284–2293.

Weissteiner, J.; Wendler, C.; Seuken, S.; Lubin, B.; and Püschel, M. 2022b. Fourier Analysis-based Iterative Combinatorial Auctions. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 549–556. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Wenzel, F.; Snoek, J.; Tran, D.; and Jenatton, R. 2020. Hyperparameter ensembles for robustness and uncertainty quantification. *arXiv preprint arXiv:2006.13570*.