EEF1-NN: Efficient and EF1 allocations through Neural Networks

Shaily Mishra, Manisha Padala, and Sujit Gujar

Machine Learning Lab, International Institute of Information Technology, Hyderabad
{shaily.mishra,manisha.padala}@research.iiit.ac.in,
sujit.gujar@iiit.ac.in

Abstract. Our goal is to allocate items to maximize efficiency while ensuring fairness. Since Envy-freeness may not always exist, we consider the relaxed notion, Envy-freeness up to one item (EF1) that is guaranteed to exist. We add the further constraint of maximizing efficiency, utilitarian social welfare (USW) among fair allocations. In general, finding USW allocations among EF1, i.e., EEF1, is an NP-Hard problem even for additive valuations. Neural networks (NNs) have shown state-of-the-art performance in designing optimal auctions as well as in learning algorithms. We design a NN inspired by U-Net for learning EEE1 allocations which we refer to as EEF1-NN. EEF1-NN is generic and scales to any number of agents and items once trained. We empirically demonstrate that EEF1-NN finds allocation with higher USW and ensures EF1 with a high probability for different distributions over input valuations.

Keywords: Resource Allocation · Neural Networks · EEF1

1 Introduction

Consider a situation where a social planner needs to allocate a set of indivisible items (goods or/and chores) among interested agents. Agents have valuations for the items, i.e., an item might be a *good* – positive valuation for one while it might be a *chore* – negative valuation for the other. The agents reveal their valuations upfront to the *social planner*. The social planner is responsible for the fair and efficient allocation of these items among the agents. For example, a Government needs to distribute resources and delegate tasks amongst its subdivisions. The subdivisions should not feel mistreated in the system. While ensuring this, the Government would like to optimally allocate items for the system's growth.

Fair division is well-explored in literature [10, 30, 34]. Economists have proposed various fairness and efficiency notions applicable in real-world settings, such as division of investments and inheritance, vaccines, tasks, etc. There are web-based applications such as Spliddit, The Fair Proposals System, Coursematch, Divide Your Rent Fairly, etc., used for credit assignment, land allocation, division of property, course allocation, and even task allotment. All these applications assure certain fairness and efficiency guarantees.

One of the most popular fairness criteria is envy-freeness (EF) [17]. An allocation is envy-free if each agent values its share at least as much as they value any other agent's share. EF is also trivially satisfied by allocating empty bundles to every agent. Hence we must also have efficiency guarantees. When we consider a complete allocation of indivisible items, EF may not exist (two agents, one good). Finding whether EF exists or not is known to be Δ_2^pcomplete [9], let alone finding an efficient allocation among EF. To overcome this limitation, we consider a prominent relaxation of EF - EF1 (Envy-freeness up to one item) [12]. Unlike EF, EF1 always exists and can be computed in polynomial time [25].

In this work, we focus on *utilitarian* social welfare (USW), i.e., the sum of utilities of individual agents. When valuations are additive, finding allocation that maximize USW (MUW) is polynomial-time solvable. While finding EF1 or MUW allocations are polynomial-time solvable, maximizing USW within EF1 allocations, i.e., *EEF1*; efficient and envy-free up to one item, is an NP-hard problem, even when valuations are additive for two agents [4, 6]. There is no known approximation algorithm for EEF1. With these theoretical limitations, we propose a data-driven learning approach, i.e., given the agents' valuations, we aim to learn EEF1. It is widely known that neural networks (NNs) outperform existing approaches in finding an optimal mapping between the given input and output data. NNs can learn algorithms [22], mechanisms [16] or solve Mixed Integer Programs [31]. Motivated by the success of NNs, we aim to learn algorithm for EEF1 using NN. We list our major challenges as follows,

Challenges. (i) In the existing integration of NNs and mechanism design, payments are at their disposal. In our work, there are no payments, and we learn discrete allocations, i.e., our solution space is binary. Whereas the output of NNs is real numbers, it can easily learn optimal fractional allocations. If we convert fractional solutions to integral, fairness guarantees no longer hold. (ii) Further, we aim to design a generalized network for any number of agents or items, even for configurations not seen during training. Most of the existing NN based approaches in EconCS train the models separately for each configuration [16, 27]. We overcome the above challenges as described below.

Contributions. To the best of our knowledge, this is the first study that integrates deep learning and fair resource allocation. In particular,

- We propose a neural network EEF1-NN inspired by U-Net to learn EEF1.
- We transform our valuations and augment them with additional channels to enhance the network's performance.
- We use a series of convolutional and up-convolutional layers to learn EEF1; EEF1-NN is generalized for any number of agents and items.
- We sample valuations from various distributions and report the expected fairness and efficiency achieved. Even for large instances, our network performs well. Moreover, [1] the network quickly computes the output; hence we can improvise this approach to be adept in practical real-time applications.
- We show that, for our setting, bagging of networks improves performance.

[1] We evaluated IP solver to solve maximizing USW w.r.t. EF1 constraints, and for 10x100, it was taking several minutes.

2 Related Work.

Fair resource allocation is well studied in the literature across various fairness and efficient notions [10, 30, 34]. When a definition of fairness is too strong or may not exist, we always look for its relaxation/approximation. There is existing work that provides approximate efficiency and fairness guarantees in [1, 7, 11, 24]. In this paper, we majorly focus on EEF1. Authors in [11] presents a framework to compute ϵ-Efficient and $\mathcal{F}$-Fair allocation, using parametric integer linear programming, which is double exponential in terms of n and m. They explored group Pareto Efficiency, which is equivalent to USW. Authors in [4] provides a pseudopolynomial-time algorithm to find MUW within EF1, which is exponential in n and polynomial in m and V, where V bounds the valuation per item.

EF1 allocations always exist and can be found in polynomial time even for general valuations [2, 14, 25]. Finding MUW allocations is also polynomial-time solvable for additive valuations, i.e., we iterate over items assign the item to the agent who values it the most. However, finding MUW allocation amongst EF1 allocations is NP-hard even for two agents with additive valuations. Also finding a truthful way for allocating EF1 is also challenging [33].

There is always a trade-off between fairness and efficiency, corresponding to the study of the price of fairness ([5, 8]). Researchers have also studied how likely a fairness notion will not exist ([15, 29, 28]). When agents' valuations are additive and drawn randomly from a uniform distribution, EF exists with a high probability when m is at least $\Omega(n \log n)$ and can be obtained by MUW allocations proved by [15]. However, the hidden constants might be high[2], and it leaves scope to explore. [29] show that RR is envy-free when $m \geq \Omega(n \log n / \log \log n)$.

Recently the EconCS community has been interested in learning mechanisms/algorithms using NN, especially in a setting of theoretical limitations [16, 21, 27, 22, 37–39]. Researchers have studied mechanism design widely [19, 18]. [16] and [21] learns optimal auctions and multi-facility mechanism using NN. [40] uses NN in the combinatorial auction for preference elicitation. [27, 38, 26] learns optimal redistribution mechanisms and MAB through NN. [39] uses NN to maximize the expected number of consumers and the expected social welfare for public projects. Additionally, [31] solves MIP on large-scale real-world application datasets and MIPLIB using a neural network that performs significantly better than the MIP solver. [32] proposed a neural network-based solution to achieve fairness in classification. Given enough data, hyper-parameter tuning, and proper training, the networks are adept at learning effective transformations. Another line of work is Reinforcement Mechanism Design, such as learning dynamic price in sponsored search auctions [13, 36].

3 Preliminaries

We consider the problem of allocating $M = [m]$ indivisible items among $N = [n]$ interested agents, where $m, n \in \mathbb{N}$. We only allow complete allocation and no

[2] Our experiments show that in the case of uniform distribution, even for 10 agents, 150 items, the probability of MUW allocation being EF1 is less than 0.5

two agents can receive the same item. That is, $A = (A_1, \ldots, A_n)$, $A \in \Pi_n(M)$ s.t., $\forall i, j \in N$, $i \neq j; A_i \cap A_j = \emptyset$ and $\bigcup_i A_i = M$. Each agent $i \in N$ has a valuation function $v_i : 2^M \rightarrow \mathbb{R}$ and $v_i(S)$ is its valuation for a $S \subseteq M$ s.t. $v_i(\emptyset) = 0$. We represent valuation profile $v = (v_1, v_2, \ldots, v_n)$. We only consider additive valuations. The valuation of an agent $i \in N$ for bundle A_i is $v_i(A_i) = \sum_{j \in A_i} v_i(\{j\})$. For an agent i, an item $j \in M$ is a *good* if, $v_i(\{j\}) \geq 0$, and a *chore* if, $v_i(\{j\}) < 0$. We consider three settings - pure goods, pure chores, and a combination of goods and chores. With this notation, we now define fairness and efficiency properties as follows.

Definition 1 (Envy-free (EF) and relaxations). *An allocation A that satisfies $\forall i, j \in N$,*

$$v_i(A_i) \geq v_i(A_j) \text{ is EF}$$
$$v_i(A_i \setminus \{k\}) \geq v_i(A_j \setminus \{k\}); \exists k \in \{A_i \cup A_j\} \text{ is EF1}$$

Definition 2 (Maximum Utilitarian Welfare (MUW)). *An allocation A^* is said to be* efficient *or MUW if it maximizes the USW, $sw(A, v) = \sum_{i \in N} v_i(A_i)$*

$$A^* \in \underset{A \in \Pi_n(M)}{arg\,max}\ sw(A, v)$$

Definition 3 (EEF1 Allocation). *We say an allocation is* EEF1 *if it satisfies EF1 fairness and maximizes USW amongst EF1 allocations.*

Given agents' valuation profile $v = (v_1, v_2, \ldots, v_n)$, we learn EEF1 allocations using a data-driven approach. We randomly draw $v_i \sim \mathcal{F}_i$ and assume $\mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2, \ldots, \times \mathcal{F}_n$ to be a known prior distribution over agents' valuations. We use the notation $n \otimes m$ to represent a setting with n agents and m items.

4 Our Approach: EEF1-NN

We construct the optimization problem for EEF1 4.1; we then formulate the Lagrangian loss function 4.2 and provide the detail of EEF1-NN 4.3 and 4.4.

4.1 Optimization Problem.

We are given a set of valuation profile $v = (v_1, v_2, \ldots, v_n)$, where v_i is drawn randomly from a distribution $\mathcal{F}_i$. Among all possible allocations, we need to find an optimal A^* that maximizes USW $sw(A, v)$ and satisfies a fairness constraint. We formulate two (generalized) fairness constraints - EF and EF1 as follows.

$$ef_i(A, v) = \sum_{k \in N} \max \left\{ 0, (v_i(A_k) - v_i(A_i)) \right\} \tag{1}$$

$$ef1_i(A,v) = \sum_{k \in N} \max \left\{ 0, (v_i(A_k) - v_i(A_i)) + \quad \min \left\{ - \max_{j \in A_k} v_i(\{j\}), \min_{j \in A_i} v_i(\{j\}) \right\} \right\} \tag{2}$$

Our goal is to maximize the expected welfare w.r.t. to the expected fairness.

$$\begin{aligned} &\text{minimize} \quad - \mathbb{E}_v \left[sw(A,v) \right] = \mathbb{E}_v [\sum_{i \in N} v_i(A_i)] \\ &\text{subject to} \ \ \mathbb{E}_v \left[\sum_{i \in N} ef_i(A,v) \right] = 0 \text{ or, } \mathbb{E}_v \left[\sum_{i \in N} ef1_i(A,v) \right] = 0 \end{aligned} \tag{3}$$

In the above optimization problem, we have 'OR' among fairness constraints, which we elaborate more on this in the Ablation Study in Section 5.1.

4.2 EEF1-NN: Lagrangian Loss Function.

EEF1-NN represents a mapping from valuation to allocation space, i.e., $\mathcal{A}^w : \mathbb{R}^{\{n \times m\}} \rightarrow \{0,1\}^{n \times m}$, where w represents the network's parameters. To learn w, we formulate our problem to optimize welfare w.r.t. to fairness constraints in Eq. 3 and formulate Lagrangian loss function ($\lambda \in \mathbb{R}_{\geq 0}$). Given $\mathcal{L}$ samples of valuation profiles $(v^1, \ldots, v^{\mathcal{L}})$ drawn from $\mathcal{F}$, the loss per sample (I_v^l) is,

$$Loss(I_v^l, w, \lambda) = \left[- sw(\mathcal{A}^w(I_v^l), v^l) + \lambda \frac{\sum_{i \in N} envy_i(\mathcal{A}^w(I_v^l), v^l)}{n} \right] \tag{4}$$

We minimize the following loss w.r.t w, $\mathbf{L}_{EEF1}(I_v^l, w, \lambda) = \frac{1}{\mathcal{L}} \sum_l Loss(v^l, w)$.

4.3 Network Details

We describe EEF1-NNś various components, including the input, architecture, and other training details in this section. EEF1-NN is a fully convolutional network (FCN) and processes input of varied sizes (i.e., height $\times$ width).

EEF1-NN: Input. We transform our valuations and augment with additional channels to enhance performance. We construct an input tensor of size $n \times m \times 6$, i.e., $I_v \in \mathbb{R}^{n \times m \times 6}$. The first channel is an $n \times m$ matrix of given valuations, i.e., $\forall i, j; I_v[i,j,1] = v_i(\{j\})$. We take a matrix X of size $n \times m$ that contains valuation for items only corresponding to the agent who values it the most, and the rest are zero. We break ties arbitrarily and expand X into five channels.

$$\forall j \in M; \ X[i.j,1] = \begin{cases} v_i(\{j\}) & \text{if } i \in argmax_i v_i(\{j\}) \\ 0 & \text{otherwise} \end{cases}$$

The next channel contains information about items indexed as $0, 5, \ldots, \lfloor m/5 \rfloor$,

$$I[i.j, 2] = \begin{cases} X[i, j, 1] & \text{if } j \in \{0, 5, 10, \ldots, \lfloor m/5 \rfloor\} \\ 0 & \text{otherwise} \end{cases}$$

The next channel contains data from the previous channel along with items indexed as $1, 6, \ldots, 1 + \lfloor m/5 \rfloor$. And so on. The last channel $I_v[i, j, 6]$ is X. We observe that single channeled input performs sub-optimal. We study the effect of input complexity on the performance in Section 5.1.

EEF1-NN: Architecture. Our architecture is inspired by U-Net architecture [35]. U-Net is a fully convolutional network built to segment bio-medical images; it also requires assigning labels to image patches and not just classifying the image as a whole. While we are working on valuation profiles rather than images, one of the primary motivations to use U-Net is to process arbitrary size images. If we use a feed-forward fully functional neural network to learn fair and efficient allocations, we need a different network for each $n \otimes m$. Moreover, just using a feed-forward functional network (multi-layer perceptron) learns EEF1 allocations for smaller values of n, but cannot learn as n increases.

EEF1-NN contains series of convolution and up-convolution layers, as given by Fig. 1. EEF1-NN has three series of Conv-UpConv layers. The convolutional layers consist of 4 repeated 3x3 convolution, each followed by a non-linear activation function, tanh. The up-convolution layers consist of 4 repeated 3x3 up-convolution, each followed by a tanh activation. Note that we are not using maxpool or skip connections as we found that they degraded the performance. We apply softmax activation function across all agents for every item to ensure each item is allocated exactly once, i.e., $\forall j \in M \ \sum_{i \in N} \mathcal{A}_i^w(\{j\}) = 1$. The final output represents the probability with which agent i receives item j. Using an FCN, we have a generalized network for $n \otimes m$; however, learning EEF1 is not easy. We need to learn integral variables, while NNs are known for learning continuous output. We describe these challenges next.

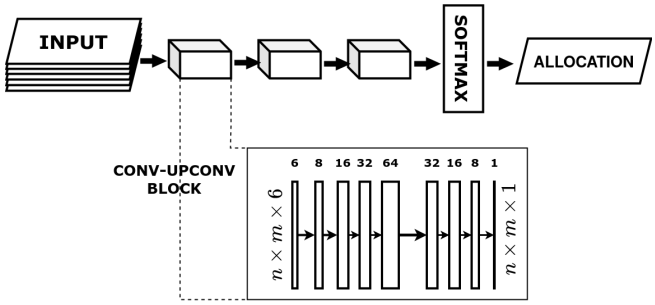

Fig. 1: EEF1-NN Architecture

4.4 Training Details

Integral Allocations. The global optima of the optimization problem in Eq. 3 might lie in a continuous allocation setting, i.e., similar to allocating divisible items. If a network learns to distribute an item equally among all agents, then the gradient vanishes. Assigning an equal partition of each item is indeed an optimum. Converting these non-integral allocation to integral is non-trivial. Hence we set a *temperature* parameter T in the softmax layer of the network to prevent getting stuck at such optima. Let $o_j = \{o_{j_1}, \ldots, o_{j_n}\}$ denote the output of our network before the final layer. The final allocation for agent i is given by, $\mathcal{A}_i^w(\{j\}) = \text{softmax}(o_{j_i}) = \frac{e^{o_{j_i}/T}}{\sum_{k=1}^{n} e^{o_{j_k}/T}}$ which represents the probability of assigning item j to all the agents. It is common to start with a large T for initial exploration and gradually reduce T to reach the global optima. While training, when we set T to 1, we get fractional allocations. As we decrease the value of T, the network outputs allocation close to discrete. The approach we want is while training, allocation output is almost discrete, but not exactly discrete. When we keep the value too low, the output is exact discrete allocations, and there is no learning because of the vanishing gradients. We appropriately choose T based on our experiments. Once the network learns, we set the parameter low enough to ensure discrete allocations.

Inefficient Local Optima. Due to the low T value, the training of EEF1-NN is highly unstable and often gets stuck at inefficient local optima. To overcome this, we use the technique of *Bootstrap Aggregation* or Bagging. It combines the predictions from multiple classifiers to produce a single classifier. We train multiple weak networks with varied hyper-parameters on the same data set, capturing different sets of local optima. While testing, the final allocation is aggregated from these networks. We pass a test sample through all networks and select the allocation that is EF1 with maximum USW. In total, we bag seven networks with varied $\lambda \in [0.1, 2]$ for increased performance. We further analyze how Bagging affects our results in the ablation study.

We implement EEF1-NN using PyTorch. We initialize the network weights using Xavier Initialization [20]. To train, we use Adam Optimizer [23] with learning rate 0.001 for 1000 epochs with $T = 0.01$. We use a batch size of 256 samples. We sample valuations from $U[0, 1]$ (goods), $U[-1, 0]$ (chores) and $U[-1, 1]$ (combination). We sample $150k$ training data for both $10 \otimes 20$ and $13 \otimes 26$ for goods, chores, and combinations, so in total, we have $300k$ training samples, and we sample $10k$ testing samples for each setting. We train seven networks with varied $\lambda \in [0.1, 2]$ and bag them for enhanced performance. The training process takes 5-6 hours to train a single network using GPU. We are training the network for $10 \otimes 20$ and $13 \otimes 26$. however, we show our test results for various $n \otimes m$. We test for network performance for $n \in [7, 15]$. We also train an individual network over different distributions such as Gaussian, Log-normal, and Exponential. We validate EEF1-NN efficacy in the next section.

5 Experiments and Results

In this section, we conduct an ablation study to set appropriate hyper-parameters and 3 types of experiments showcasing performances across different item types, distributions, and scalability. To report the network performance, we define the following two metrics: the measure of fairness (probability of an allocation to be EF1) and the other of efficiency (how close our social welfare is to optimal).
Evaluation Metrics.

1. α_{EF1}^{ALG} - It measures the probability with which an algorithm ALG outputs EF1 allocation. α_{EF1} is the ratio of the number of samples that are EF1 to the total number of samples.
2. β_{SW}^{ALG} - It measures the ratio of expected USW of an algorithm ALG by expected USW of MUW allocation. $\beta_{SW}^{ALG} = \mathbb{E}(sw^{ALG})/\mathbb{E}(sw^{MUW})$.

Note that $\beta_{SW}^{ALG} \in [0, 1]$ for goods, $\beta_{SW}^{ALG} \geq 1$ for chores, and will depend on the overall social welfare (positive/negative) for a combination of goods and chores. We will use that notation $(\alpha_{EF1}, \beta_{sw})$ to report performance.

5.1 Ablation Study

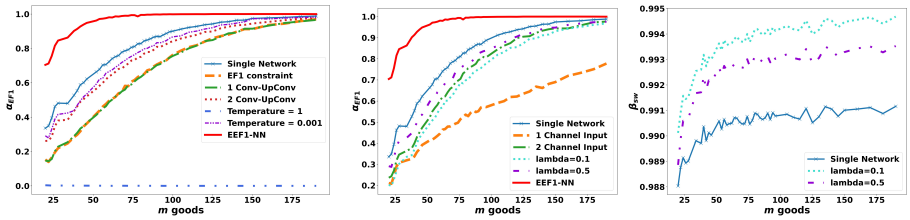

Fig. 2: Ablation Study over varied hyper-parameters

We illustrate the effect of specific hyper-parameters in the performance of EEF1-NN in Fig. 2. We set $n = 10$ goods for all the experiments. In the plots, the red line with the label EEF1-NN denotes the α_{EF1} for optimal parameters. Corresponding to EEF1-NN, a single network from this bagged network is labeled as *Single Network*. This *Single Network* trained with six-channeled input, $\lambda = 1$, and $T = 0.01$ is the baseline to compare across this ablation study. Only one parameter is changed w.r.t. the *Single Network* for the study.
(i) Effect of Temperature T. In Fig. 2(left), when $T = 1$, it converges to fractional allocation represented by the blue line at the bottom of the plot. When $T = 0.001$ (violet line), it is too low, and performs sub-optimally compared to single network. We also noticed that the performance for $T = 0.01$ and $T = 0.1$ are close to each other. We set $T = 0.01$ for all the bagged networks in EEF1-NN.

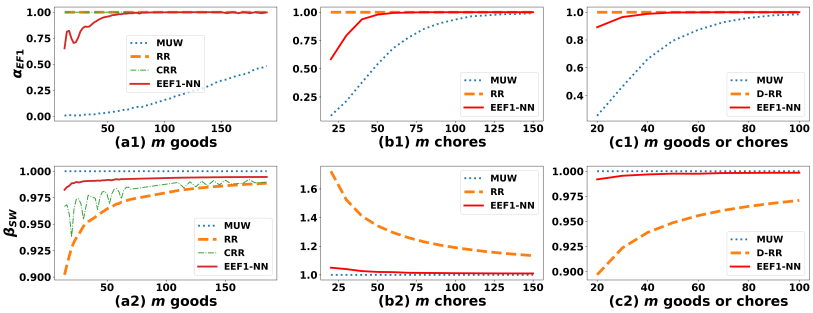

Fig. 3: Exp1 ($n = 10$, Uniform Distribution)

(ii) Effect of Series of Conv-UpConv layers. We select three series of Conv-UpConv for EEF1-NN as illustrated in Fig. 2(left). As seen from Fig. 2(left), a performance increase between 1-series (green dashed line) and 2-series (red dotted line) is significant compared to 2-series and 3-series (single network). The complexity of the network having 4-series is far more than the performance improvement.

(iii) Effect of Loss Function We empirically analyze how different envy definitions (Eq. 3) affects the training of EEF1-NN. As shown in Fig. 2(left), when we train our network using EF, i.e., Eq. 1 (*Single Network*, the network performs significantly better than when trained using EF1, i.e., Eq.2 (orange dashed line). For example, for $10 \otimes 20$, the performance of *Single Network* is (0.3358,17.9611), whereas the performance of the EF1 trained network is (0.1530,17.8708).

iv) Number of Input Channels. To enhance our network performance, we experimented with different channel inputs. For 2-channeled input, we set the first channel of input tensor as the valuation and the second to X. Like 6-channeled input, we expand X to 11 channels. As shown in Fig. 2(right), the performance of a 1-channeled network is $(0.2113, 17.8976)$, 2-channeled network is $(0.2365, 17.8991)$, *Single Network* is $(0.3358, 17.9611)$, and 11-channeled network is $(0.3925, 17.9395)$. The network cannot be generalized for 11-channeled.

v) Effect of Bagging. We bag different combination of networks, each trained for varied λ in Fig. 2(right). More the λ, more penalty is given to envy. When λ is too small, the network learns a more efficient but less fair allocation. As we increase λ up to a certain value, the network learns less efficient but more fair allocations. We observed that varying λ results in converging to the different optimum. We bagged seven networks trained with $\lambda \in [0.1, 2]$. Bagging (EEF1-NN) outperforms the performance of a single network.

5.2 Experiment Details and Observations

We conduct three types of experiments, Exp1: Different kinds of resources, Exp2: Input distributions, and Exp3: Scalability. Since approaches in [4, 11]

are exponential, we cannot report results on our experimentation configuration. We compare EEF1-NN with the following existing methods,

- MUW- We compare our results with MUW since we don't have EEF1. Note that EEF1-NN welfare is close to MUW; we can say it is also close to EEF1.
- Round Robin (RR)- [14] finds EF1 for goods or chores. *Double Round Robin* (D-RR) [2] finds EF1 for the combination of goods and chores.
- Constrained Round Robin (CRR)- We implement CRR [3] to find RB sequences to increase efficiency. An RB sequence for goods.

EXP1: Performance across differed resources for Uniform Distribution. For $n = 10$, we compare α_{EF1} in Fig. 3 (a1, b1, c1) and β_{SW} in Fig. 3 (a2, b2, c2). As m increases, all the approaches move closer to EEF1. We observe that MUW (blue dotted line) converges towards EEF1 much faster for chores or combinations than goods. While RR converges to EEF1 much faster in goods compared to chores or combinations. We discuss this convergence in detail in Table 1. We observe that EEF1-NN consistently has better α_{EF1} than MUW and β_{sw} than RR/CRR. We observe that $\alpha_{EF1}^{EEF1-NN}$ is close to 1 after a certain m. At the same time, EEF1-NN is far more efficient than *RR*. (Fig 3 (a2,b2,c2)). Note that the CRR is only for goods. We observe that compared to CRR, EEF1-NN obtains marginally better β_{SW}, in Fig. 3(a2).
EXP2: Performance across different distributions. We provide the performance of EEF1-NN when the valuations are sampled from different distributions such as Gaussian (μ=0.5,σ=1) in Fig 4(a1, a2), Log-normal (μ = 0.5,σ=1) in Fig 4(b1, b2), and Exponential ($\lambda = 1$) in Fig 4(c1, c2). We observe that in all three $\alpha_{EF1}^{EEF1-NN} > 0.99$ and $\beta_{SW}^{EEF1-NN} > 0.99$ for $m \geq 40$ in Fig. 4.
EXP3: Scalability to larger number of agents. EEF1-NN is trained only for $10 \otimes 20$ and $13 \otimes 26$. As we have seen in the previous results and in Fig. 5, the performance scales across varying m and n seamlessly. We provide the performance of EEF1-NN when $n = 7, 12, 14$ in Fig. 5[3].
Analysis of Convergence to EEF1 Allocations (Uniform Distribution)

Definition 4 ($m^\star(n)$). *For a given n, we say an algorithm converges to EEF1 allocation at $m^\star(n)$ if $\forall m > m^\star(n)$,*
(i) For goods*: $\alpha_{EF1}^{ALG} \geq 0.99$ and $\beta_{sw}^{ALG} \geq 0.99$.*
(ii) For chores*: $\alpha_{EF1}^{ALG} \geq 0.99$, and $\beta_{sw}^{ALG} \leq 1.02$.*

We empirically study $m^\star(n)$ value after which EEF1-NN, RR, and MUW start converging towards EEF1 for uniform distribution in Table 1. We don't report CRR in this; as we see fluctuations in β_{sw}, it doesn't increase smoothly in Fig. [35]; For goods, EEF1-NN reaches close to EEF1 faster than MUW and RR, and RR reaches close to EEF1 faster than MUW. EEF1-NN converges first, then MUW, and finally RR for chores in Table 1. For chores, we report the value of $m^\star$ for RR when $\beta_{sw} \leq 1.064$ since m is significantly higher than MUW and RR, concluding that RR converges after a considerably larger m. As m increases,

[3] To report performance for $n \in [7, 9]$, we reduce a Conv-UpConv layer and train accordingly with $7 \otimes 14$ and $10 \otimes 20$ valuation profiles.

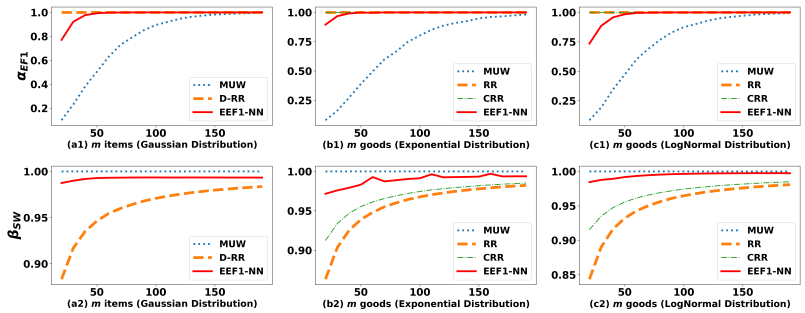

Fig. 4: Exp2 ($n = 10$, different distributions)

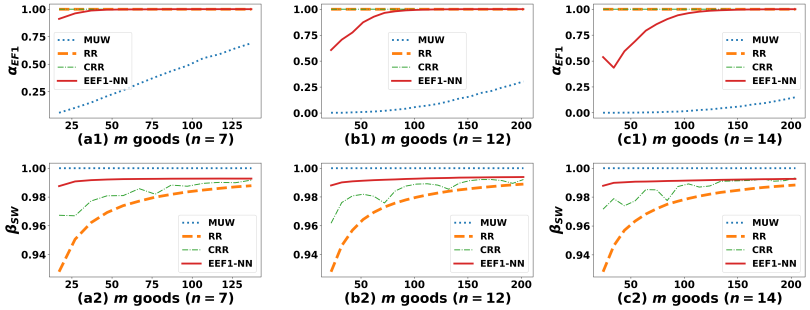

Fig. 5: Exp3 ($n = 7, 12, 14$ goods, Uniform Distribution)

$\alpha_{EF1}, \alpha_{EFX}$, and α_{EF} of MUW gets closer. Note that we do not experiment with all possible m; the actual value of $m^\star(n)$ may be slightly different from Table 1. We aim to observe a pattern among approaches to achieve EEF1.

Table 1: Value of $m^\star(n)$ as different approaches converge to EEF1 allocations

n	(m) Goods			(m) Chores		
	EEF1-NN	RR	MUW	EEF1-NN	RR	MUW
7	38	159	380	44	195	112
8	46	172	450	44	240	120
9	57	186	530	53	295	130
10	70	196	610	60	340	148
11	82	206	660	68	400	160
12	94	214	740	75	455	167
13	110	220	840	83	505	180
14	134	228	940	87	565	190

Discussion $\alpha_{EF1}^{EEF1-NN}$ reaches 1 much faster than α_{EF1}^{MUW}, and $\beta_{sw}^{EEF1-NN}$ reaches close to β_{sw}^{MUW} much faster than RR, D-RR, CRR. EEF1-NN shows a better trade-off between EF1 and efficiency than the existing approaches for different input distributions. We trained our network with fixed $n \otimes m$ for goods or/and chores, it is interesting that the performance scales for any m and a large n. We conclude that EEF1-NN effectively learns and provides a better trade-off when m is not too large or small compared to n but is in a specific range.

6 Conclusion

In this paper, we proposed a neural network EEF1-NN to find EEF1, an NP-hard problem. We designed architecture and input representation combined with other training heuristics to learn approximate EEF1 on average. We studied the effect of each proposed constituent on performance. Our experiments demonstrated the efficacy of EEF1-NN for different input distributions across various n and m over existing approaches. With theoretical limitations and the success of neural networks, we believe that the path of amalgamating deep learning and resource allocation is worth exploring further with more complex objective functions.

References

1. Amanatidis, G., Markakis, E., Nikzad, A., Saberi, A.: Approximation algorithms for computing maximin share allocations. ACM Transactions on Algorithms **13**(4), 1–28 (Dec 2017)
2. Aziz, H., Caragiannis, I., Igarashi, A.: Fair allocation of combinations of indivisible goods and chores (2018), http://arxiv.org/abs/1807.10684
3. Aziz, H., Huang, X., Mattei, N., Segal-Halevi, E.: The constrained round robin algorithm for fair and efficient allocation (2019)
4. Aziz, H., Huang, X., Mattei, N., Segal-Halevi, E.: Computing fair utilitarian allocations of indivisible goods (2020), https://arxiv.org/abs/2012.03979
5. Barman, S., Bhaskar, U., Shah, N.: Optimal bounds on the price of fairness for indivisible goods. In: Chen, X., Gravin, N., Hoefer, M., Mehta, R. (eds.) Web and Internet Economics - 16th International Conference, WINE 2020, Beijing, China, December 7-11, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12495, pp. 356–369. Springer (2020). https://doi.org/10.1007/978-3-030-64946-3_25, https://doi.org/10.1007/978-3-030-64946-3_25
6. Barman, S., Ghalme, G., Jain, S., Kulkarni, P., Narang, S.: Fair division of indivisible goods among strategic agents. p. 1811–1813. AAMAS '19 (2019)
7. Barman, S., Krishnamurthy, S.K., Vaish, R.: Finding fair and efficient allocations. p. 557–574. EC '18, Association for Computing Machinery, New York, NY, USA
8. Bei, X., Lu, X., Manurangsi, P., Suksompong, W.: The price of fairness for indivisible goods. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence. p. 81–87. IJCAI'19, AAAI Press (2019)
9. Bouveret, S.: Efficiency and envy-freeness in fair division of indivisible goods: Logical representation and complexity. Journal of Artificial Intelligence Research **32** (05 2008)

10. Bouveret, S., Chevaleyre, Y., Maudet, N., Moulin, H.: Fair Allocation of Indivisible Goods, p. 284–310. Cambridge University Press (2016)
11. Bredereck, R., Kaczmarczyk, A., Knop, D., Niedermeier, R.: High-multiplicity fair allocation: Lenstra empowered by n-fold integer programming. In: Proceedings of the 2019 ACM Conference on Economics and Computation. p. 505–523. EC '19
12. Budish, E.: The Combinatorial Assignment Problem: Approximate Competitive Equilibrium from Equal Incomes. Journal of Political Economy **119**(6), 1061–1103 (2011)
13. Cai, Q., Filos-Ratsikas, A., Tang, P., Zhang, Y.: Reinforcement mechanism design for e-commerce. In: Proceedings of the 2018 World Wide Web Conference. p. 1339–1348. WWW '18 (2018)
14. Caragiannis, I., Kurokawa, D., Moulin, H., Procaccia, A.D., Shah, N., Wang, J.: The unreasonable fairness of maximum nash welfare. In: Proceedings of the 2016 ACM Conference on Economics and Computation. p. 305–322. EC '16, Association for Computing Machinery, New York, NY, USA (2016)
15. Dickerson, J.P., Goldman, J., Karp, J., Procaccia, A.D., Sandholm, T.: The computational rise and fall of fairness. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence. p. 1405–1411. AAAI'14, AAAI Press (2014)
16. Duetting, P., Feng, Z., Narasimhan, H., Parkes, D., Ravindranath, S.S.: Optimal auctions through deep learning. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 1706–1715. PMLR (09–15 Jun 2019)
17. Foley, D.K.: Resource allocation and the public sector (1967)
18. Garg, D., Narahari, Y., Gujar, S.: Foundations of mechanism design: A tutorial part 2-advanced concepts and results. Sadhana **33**(2), 131–174 (2008)
19. Garg, D., Narahari, Y., Gujar, S.: Foundations of mechanism design: A tutorial part 1-key concepts and classical results. Sadhana **33**(2), 83–130 (2008)
20. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y.W., Titterington, M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 9, pp. 249–256. JMLR Workshop and Conference Proceedings, Chia Laguna Resort, Sardinia, Italy (13–15 May 2010), http://proceedings.mlr.press/v9/glorot10a.html
21. Golowich, N., Narasimhan, H., Parkes, D.C.: Deep learning for multi-facility location mechanism design. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence. p. 261–267. IJCAI'18, AAAI Press (2018)
22. Kim, H., Jiang, Y., Rana, R.B., Kannan, S., Oh, S., Viswanath, P.: Communication algorithms via deep learning. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=ryazCMbR-
23. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. International Conference on Learning Representations (12 2014)
24. Kurokawa, D., Procaccia, A.D., Wang, J.: Fair enough: Guaranteeing approximate maximin shares. J. ACM **65**(2) (Feb 2018)
25. Lipton, R.J., Markakis, E., Mossel, E., Saberi, A.: On approximately fair allocations of indivisible goods. In: Proceedings of the 5th ACM Conference on Electronic Commerce. p. 125–131. EC '04 (2004)
26. Manisha, P., Gujar, S.: Thompson sampling based multi-armed-bandit mechanism using neural networks. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. p. 2111–2113. AAMAS '19 (2019)

27. Manisha, P., Jawahar, C.V., Gujar, S.: Learning optimal redistribution mechanisms through neural networks. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. p. 345–353. AAMAS '18 (2018)
28. Manurangsi, P., Suksompong, W.: When do envy-free allocations exist? SIAM Journal on Discrete Mathematics **34**, 1505–1521 (01 2020)
29. Manurangsi, P., Suksompong, W.: Closing gaps in asymptotic fair division. SIAM J. Discret. Math. (2021)
30. Markakis, E.: Approximation algorithms and hardness results for fair division with indivisible goods. In: Endriss, U. (ed.) Trends in Computational Social Choice, chap. 12, pp. 231–247. AI Access (2017)
31. Nair, V., Bartunov, S., Gimeno, F., von Glehn, I., Lichocki, P., Lobov, I., O'Donoghue, B., Sonnerat, N., Tjandraatmadja, C., Wang, P., Addanki, R., Hapuarachchi, T., Keck, T., Keeling, J., Kohli, P., Ktena, I., Li, Y., Vinyals, O., Zwols, Y.: Solving mixed integer programs using neural networks. ArXiv **abs/2012.13349** (2020)
32. Padala, M., Gujar, S.: Fnnc: Achieving fairness through neural networks. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence,{IJCAI-20}, International Joint Conferences on Artificial Intelligence Organization (2020)
33. Padala, M., Gujar, S.: Mechanism design without money for fair allocations. In: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. pp. 382–389 (2021)
34. Procaccia, A.D., Moulin, H.: Cake Cutting Algorithms, p. 311–330. Cambridge University Press (2016)
35. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. vol. 9351, pp. 234–241 (10 2015)
36. Shen, W., Peng, B., Liu, H., Zhang, M., Qian, R., Hong, Y., Guo, Z., Ding, Z., Lu, P., Tang, P.: Reinforcement mechanism design, with applications to dynamic pricing in sponsored search auctions (2017), http://arxiv.org/abs/1711.10279
37. Shen, W., Tang, P., Zuo, S.: Automated mechanism design via neural networks. In: Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems. pp. 215–223 (2019)
38. Tacchetti, A., Strouse, D., Garnelo, M., Graepel, T., Bachrach, Y.: A neural architecture for designing truthful and efficient auctions (2019), http://arxiv.org/abs/1907.05181
39. Wang, G., Guo, R., Sakurai, Y., Ali Babar, M., Guo, M.: Mechanism design for public projects via neural networks. In: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems. p. 1380–1388. AAMAS '21, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2021)
40. Weissteiner, J., Seuken, S.: Deep learning—powered iterative combinatorial auctions. Proceedings of the AAAI Conference on Artificial Intelligence **34**(02), 2284–2293 (Apr 2020). https://doi.org/10.1609/aaai.v34i02.5606, http://dx.doi.org/10.1609/aaai.v34i02.5606