EEF1-NN: Efficient and EF1 allocations through Neural Networks

Shaily Mishra
International Institute of Information Technology (IIIT)
Hyderabad, India
shaily.mishra@research.iiit.ac.in

Manisha Padala
International Institute of Information Technology (IIIT)
Hyderabad, India
manisha.padala@research.iiit.ac.in

Sujit Gujar
International Institute of Information Technology (IIIT)
Hyderabad, India
sujit.gujar@iiit.ac.in

ABSTRACT

Neural networks have shown state-of-the-art performance in designing auctions, where the network learns the optimal allocations and payment rule to ensure desirable properties. Motivated by the same, we focus on learning fair division of resources, with no payments involved. Our goal is to allocate the items, goods and/or chores efficiently among the fair allocations. By fair, we mean an allocation that is Envy-free (EF). However, such an allocation may not always exist for indivisible resources. Therefore, we consider the relaxed notion, Envy-freeness up to one item (EF1) that is guaranteed to exist. However, it is not enough to guarantee EF1 since the allocation of empty bundles is also EF1. Hence, we add the further constraint of efficiency, maximum utilitarian social welfare (USW).

In general finding, USW allocations among EF1 is an NP-Hard problem even when valuations are additive. In this work, we design a network for this task which we refer to as EEF1-NN. We propose an UNet inspired architecture, Lagrangian loss function, and training procedure to obtain desired results. We show that EEF1-NN finds allocation close to optimal USW allocation and ensures EF1 with a high probability for different distributions over input valuations. Compared to existing approaches EEF1-NN empirically guarantees higher USW. Moreover, EEF1-NN is scalable and determines the allocations much faster than solving it as a constrained optimization problem.

KEYWORDS

Fairness, Efficiency, and Neural Networks

ACM Reference Format:

Shaily Mishra, Manisha Padala, and Sujit Gujar. 2018. EEF1-NN: Efficient and EF1 allocations through Neural Networks. In Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY, ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

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 or/and delegate tasks amongst its subdivisions. The subdivisions should not feel mistreated in the system (fairness), and while ensuring fairness, the Government would like to further allocate items optimally for the system’s growth (efficiency).

The fair division of items is well-explored in literature [13, 40, 42, 48]. Economists have proposed various fairness notions (Envy-freeness [27], Equitable [24], Proportional [46]) and efficiency notions (Utilitarian Welfare, Nash Welfare, Egalitarian Welfare, Pareto Efficiency). These are 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.

Fairness Notion. One of the most popular fairness criteria is envy-freeness (EF)[27]. An allocation is envy-free if each agent values its share at least as much as they value any other agent’s share, i.e., no agent is envious. EF is also trivially satisfied by allocating empty bundles to every agent. Hence we must also have efficiency/optimality guarantees. When we consider a complete allocation of indivisible items, Envy-free may not exist—for example, two agents, one good. The agent who doesn’t get the good will always envy the one who receives it. Finding whether EF allocation exists or not is known to be $\Delta$-complete [12], 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) [11]. Unlike EF, EF1 always exists and can be computed in polynomial time [36].

Efficiency Notion. The notion of Pareto efficiency $^1$ is widely studied in fair resource allocation, i.e., PE and fair allocations [3, 9, 28]. In this work, we instead focus on utilitarian social welfare (USW), i.e., the sum of utilities of individual agents. When valuations are additive, finding allocations that maximize utilitarian welfare (MUW) is polynomial-time solvable. While finding EF1 or MUW allocations are polynomial-time solvable, maximizing utilitarian welfare within EF1 allocation is an NP-hard problem [2–4, 8, 22] even in additive valuation settings. There is existing work that provides approximate efficiency and fairness guarantees in [1, 9, 15, 35]. But to find allocations that are MUW among EF allocations is an NP-Hard problem even when valuations are additive for two agents [5, 8].

---

$^1$ An allocation $A$ is said to be PE if it is not Pareto dominated by any other allocation, i.e., there is no other allocation $A'$ such that $\forall i \in N, v_i(A'_i) \geq v_i(A_i)$, and $\exists i \in N, v_i(A'_i) > v_i(A_i)$. 

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Woodstock '18, June 03–05, 2018, Woodstock, NY © 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06. . . $15.00
https://doi.org/10.1145/1122445.1122456
Aziz et al. [5] provide a pseudopolynomial-time algorithm for finding MUW within EF1, which is exponential in the number of agents and polynomial in the number of items and $V$, where $V$ bounds the valuation per item. When agents valuations are additive and drawn randomly from a uniform distribution, envy-free allocation exists with a high probability when the number of items $m$ is at least $\Omega(n \log n)$ and can be obtained by MUW allocations proven by Dickerson et al. [23]. This guarantee is achieved only for significantly large values of $m$. However, the hidden constants might be high\(^2\), and it leaves scope to explore.

With these theoretical limitations, in this paper, we focus on a data-driven approach, i.e., given the agents’ valuations, we aim to learn allocations that maximize USW amongst EF1, which we call $EEF1$; efficient and envy-free up to one item. It is widely known that neural networks outperform existing approaches in finding an optimal mapping (e.g., mechanisms, algorithms) between the input and output data [19, 20, 25, 31, 33, 37, 45, 47, 49, 50, 52]. Given enough data, hyper-parameter tuning, and proper training, the networks are adept at learning effective transformations. Dutting et al. [25] learn a mechanism from input valuation space to allocations and payments that provide maximum revenue and ensure truthful valuation elicitation. Motivated by the success of NNS in resource allocations and the theoretical limitations, we propose a learning-based approach for fair and efficient resource allocations. Note that payments are at their disposal in most previous NN-based allocation approaches, and the main challenge is to learn payments. In our work, there are no payments, and we are learning allocations via NNS. The major challenges are as follows.

**Challenges.** To the best of our knowledge, this is the first study that integrates deep learning and fair resource allocation. It entails addressing the following challenges. (i) Allocations for indivisible items are in the discrete space, whereas the output of NNS being real numbers, it can easily learn optimal fractional allocation, i.e., give each agent an equal proportion of an item. If we convert fractional solutions to integral solutions in our settings, fairness guarantees no longer hold. (ii) Further, we aim to design a generalized network that should work 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, for example, in papers [25, 37].

**Contributions.** For a given distribution, there is a certain likelihood for MUW allocation to be EF1. It increases when the number of items is very large, w.r.t. the number of agents. We train the NN for the cases where MUW allocation is rarely EF1. In order to provide competitive results, we address the above challenges by introducing the following novelty,

- We formulate a Lagrangian loss function to find allocations that are *Envy Free up to one item Efficient* (maximal USW) and EF1.
- We show that, for our setting, the Bagging of multiple networks improves performance.
- We sample valuations from various distributions and report the expected fairness and efficiency achieved. Our network performs well even for large instances with more than 10 agents and 100 items. Moreover, compared to any integer optimization solver, the network quickly computes the output; hence we can improvise this approach to be adept in practical real-time applications.

2 RELATED WORK.

Fair resource allocation is well studied in the literature across various fairness and efficient notions [13, 26, 40, 42, 48]. When a definition of fairness is too strong or may not exist, we always look for its relaxation/approximation; researchers have also studied how likely it is that a fairness notion will not exist.

In this paper, we are majorly concerned with EF1 and USW. EF1 allocations always exist and can be found in polynomial time. When agents have additive valuations, the round-robin algorithm always guarantees EF1 for (pure) goods or chores and the double round-robin algorithm for the combination of goods and chores. [3, 21]. When agents have general valuations, we can find EF1 allocations in $O(mn^3)$ using a cycle-elimination algorithm. [36]. 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 [2–5, 8, 22].

Authors in [14] present a framework to compute $\epsilon$-Efficient and $F$-Fair allocation, using parametric integer linear programming, which is double exponential in terms of the number of agents and items. In [14], they explore group Pareto Efficiency, which is equivalent to USW. Authors in [5] provide a pseudopolynomial-time algorithm for finding MUW within EF1 for any fixed number of agents for goods, which is exponential in the number of agents. In the paper, [41], the authors present an approximately optimal round-robin order that gives highly efficient (USW) EF1 allocations in the Reviewer Assignment setting; however, the setting is quite different from ours, as we are not concerned with the multiplicity of items.

In further related work, papers [3, 9] explore PE and EF1 allocations, [28, 29] explore PE and EQ1 allocations, [6] explore PE and ProP1 allocations for various items (goods or/and chores). There will always be a tradeoff between fairness and efficiency, corresponding to the study of the price of fairness [7, 10]. Alongside, Researchers have also studied how likely it is that a fairness notion will not exist [23, 38, 39]. In [39], the authors show that Round Robin allocation 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 neural networks, esp. in a setting of theoretical limitations. For, e.g., In paper [32], the authors provide a strategy-proof, multi-facility mechanism that minimizes expected social cost via NN. Authors in [17], the authors integrate machine

---

\(^2\)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

- We formulate a Lagrangian loss function to find allocations that are *Envy Free up to one item Efficient* (maximal USW) and EF1.
- We show that, for our setting, the Bagging of multiple networks improves performance.
- We sample valuations from various distributions and report the expected fairness and efficiency achieved. Our network performs well even for large instances with more than 10 agents and 100 items. Moreover, compared to any integer optimization solver, the network quickly computes the output; hence we can improvise this approach to be adept in practical real-time applications.

2 RELATED WORK.

Fair resource allocation is well studied in the literature across various fairness and efficient notions [13, 26, 40, 42, 48]. When a definition of fairness is too strong or may not exist, we always look for its relaxation/approximation; researchers have also studied how likely it is that a fairness notion will not exist.

In this paper, we are majorly concerned with EF1 and USW. EF1 allocations always exist and can be found in polynomial time. When agents have additive valuations, the round-robin algorithm always guarantees EF1 for (pure) goods or chores and the double round-robin algorithm for the combination of goods and chores. [3, 21]. When agents have general valuations, we can find EF1 allocations in $O(mn^3)$ using a cycle-elimination algorithm. [36]. 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 [2–5, 8, 22].

Authors in [14] present a framework to compute $\epsilon$-Efficient and $F$-Fair allocation, using parametric integer linear programming, which is double exponential in terms of the number of agents and items. In [14], they explore group Pareto Efficiency, which is equivalent to USW. Authors in [5] provide a pseudopolynomial-time algorithm for finding MUW within EF1 for any fixed number of agents for goods, which is exponential in the number of agents. In the paper, [41], the authors present an approximately optimal round-robin order that gives highly efficient (USW) EF1 allocations in the Reviewer Assignment setting; however, the setting is quite different from ours, as we are not concerned with the multiplicity of items.

In further related work, papers [3, 9] explore PE and EF1 allocations, [28, 29] explore PE and EQ1 allocations, [6] explore PE and ProP1 allocations for various items (goods or/and chores). There will always be a tradeoff between fairness and efficiency, corresponding to the study of the price of fairness [7, 10]. Alongside, Researchers have also studied how likely it is that a fairness notion will not exist [23, 38, 39]. In [39], the authors show that Round Robin allocation 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 neural networks, esp. in a setting of theoretical limitations. For, e.g., In paper [32], the authors provide a strategy-proof, multi-facility mechanism that minimizes expected social cost via NN. Authors in [17], the authors integrate machine
learning in the combinatorial auction for preference elicitation. Further, in [52], authors use a neural network to improve it and reformulate WDP into a mixed-integer program. Authors in [37, 47] learn optimal redistribution mechanisms through NNs. Another line of work is Reinforcement Mechanism Design, such as learning dynamic price in sponsored search auctions [19, 44]. In [51], the authors use NN to maximize the expected number of consumers and the expected social welfare for public projects.

3 PRELIMINARIES

We consider the problem of allocating $M = \{m\}$ indivisible items among $N = \{n\}$ interested agents. 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 consider three settings - pure goods, pure chores, and a combination of goods and chores. In combination, an item may be good for one agent and a chore for another. 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 represent valuation profile $v = (v_1, v_2, \ldots, v_n)$. We consider additive valuations. The valuation of an agent $i \in N$ for bundle $A_i$ is $v_i(A_i) = \sum_{j \in A_i} v_i(j)$.

Utilitarian Social Welfare (USW) is defined as $sw(A, v) = \sum_{i \in N} v_i(A_i)$.

We assume $F = F_1 \times F_2 \times \ldots \times F_n$ to be a known prior distribution over agents’ valuations. We randomly draw $v_i \sim F_i$. An allocation $A \in \{0, 1\}^{m \times n}$ is an $m$ way partition of $M$ into $N$. Here, $A_i \in \{0, 1\}$ is the bundle assigned to the agent $i$ and $A_i \cap A_k = \emptyset, i, k \in N$ and $i \neq k$. We consider a complete allocation of items, i.e., $\cup_i A_i = M$. We use the notation $n \times m$, for a problem setting with $n$ agents for $m$ items. Given a valuation profile of agents $v = (v_1, v_2, \ldots, v_n)$, our goal is to find a fair and efficient allocation. First, we define fairness and efficiency properties.

Definition 3.1 (Envy-free (EF)). An allocation $A$ is said to be EF, if no agent envies another agent, i.e., $\forall i, j \in N, v_i(A_i) \geq v_i(A_j)$.

As EF allocation may not always exist for indivisible goods, we consider a generalized version of relaxation of the EF defined by Budish [18].

Definition 3.2 (Envy-free up to one item (EF1)). An allocation $A$ is said to be EF1 if envy of any agent can be eliminated by either removing any good from the envied agent’s allocation or removing any chore from the agent’s allocation. i.e., when either of the following is true $\forall i, k \in N$,

1. $\exists j \in A_k \text{ s.t. } v_i(A_i) \geq v_i(A_k \setminus \{j\})$
2. $\exists j \in A_j \text{ s.t. } v_i(A_i \setminus \{j\}) \geq v_i(A_k)$

Definition 3.3 (Maximum Utilitarian Welfare (MUW)). An allocation $A^*$ is said to be efficient or MUW if it maximizes the USW, i.e.

$A^* \in \text{argmax}_{A \in \{0, 1\}^{m \times n}} sw(A, v)$

Definition 3.4 (EEF1 Allocation). We say an allocation is EEF1 if it satisfies EF1 fairness and maximizes USW amongst EF1 allocations.

4 OUR APPROACH: EEF1-NN

EEF1-NN represents a mapping from valuation space 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 weights. To learn the network parameters, we formulate our problem to optimize social welfare w.r.t. fairness constraints in Section 4.1. We construct the Lagrangian Loss function of this optimization problem for the training of EEF1-NN. We explain our architecture in Section 4.2 and training details in Section 4.3. Note that we represent EEF1-NN by $\mathcal{A}^w$ and an allocation by $A$.

4.1 Optimization Problem.

Consider $n$ interested agents and $m$ indivisible items; it can be good $v_i(\{j\}) \geq 0$, or chore $v_i(\{j\}) < 0$. 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 $F_i$. Among all possible allocations $A \in \{0, 1\}^{n \times m}$, we need to find an optimal $A^*$ that maximizes utilitarian social welfare $sw(A, v)$ and satisfies a fairness constraint. We formulate two fairness constraints - EF and EF1. In Definition 3.1, the envy of an agent $i \in N$ according to EF is as follows,

$\text{envy}_i(A, v) = \sum_{k \in N} \max \left\{ 0, (v_i(A_k) - v_i(A_j)) \right\}$

In Definition 3.2, the $\text{ef}_1A_i$ of an agent $i \in N$, according to EF1 is as follows:

$\text{ef}_1A_i(A, v) = \sum_{k \in N} \max \left\{ 0, (v_i(A_k) - v_i(A_j)) + \min \left\{ -\max_{j \in A_k} v_i(\{j\}), \min_{j \in A_i} v_i(\{j\}) \right\} \right\}$

The above constraints are generalized formulations for both goods and chores. Given a set of valuation profiles, our goal is to maximize the expected welfare w.r.t. the expected fairness.

In the above optimization problem, we have ‘OR’ among fairness constraints, which we will elaborate on in the Ablation Study in Section 5.1.

EEF1-NN: Lagrangian Loss Function. We now formulate the objective function given by Eq. 3 using the Lagrangian multiplier method. We use the Lagrangian multiplier $\lambda \in \mathbb{R}_{\geq 0}$ to combine the objective and constraints. Given $L$ samples of valuation profiles $(v_1, \ldots, v_L)$ drawn from $F$, we have the corresponding input $I^w$ and the loss for each sample is given by,

$\text{Loss}(I^w, \lambda, \mathcal{L}) = \frac{1}{n \times m} \left| \text{sw}(\mathcal{A}^w(I^w), v^j) + \sum_{i \in N} \text{enjoy}_i(\mathcal{A}^w(I^w), v^j) \right|$

We minimize the following loss w.r.t $\mathcal{L}$,

$L_{\text{EEF1}}(I^w, \mathcal{L}, \lambda) = \frac{1}{L} \sum_{l} \text{Loss}(v^j, \mathcal{L})$
4.2 Network Details

We describe EEF1-NN’s 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 × width). Because of using an FCN, EEF1-NN runs independently of n and m.

EEF1-NN: Input. We construct an input tensor of size $n \times m \times 6$, i.e., the input to the network is a six channeled input $I_0 \in \mathbb{R}^{n \times m \times 6}$. The first channel is an $n \times m$ matrix of given valuations, i.e., $\forall i, j; I_0[i, j, 1] = v_i(\{j\})$. Note that we sample the valuation from a distribution $\mathcal{F}$. 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 elements are zeros, i.e., It takes a value $v_i(\{j\})$ for each item (column) for the agent (row) having maximum value for it, i.e.,

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

We break ties arbitrarily. We expand this $n \times m$ matrix into five channels, such that the first one will contain information about items indexed as $0, 5, 10, \ldots, [m/5]$, i.e.,

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

The next channel will have data from the previous channel and along with that about items indexed as $1, 6, 11, \ldots, 1 + [m/5]$.

$$
I[i, j, 3] = I[i, j, 2] + \begin{cases}
  X[i, j, 1] & \text{if } j \in \{1, 6, 11, \ldots, 1 + [m/5] \} \\
  0 & \text{otherwise}
\end{cases}
$$

And so on. The last channel, $I_6[i, j, 6]$ will be equal to $X$. We observe that giving single channeled input of only valuations, i.e., tensor of size $(n \times m \times 1)$, performs sub-optimal as opposed to the six-channel. We evaluate the performance of six channeled input across various other inputs in Section 5.1. With this representation, the network learns better.

EEF1-NN: Architecture. Our architecture is inspired by U-Net architecture [43]. U-Net is a fully convolutional network built to segment bio-medial images; it also requires assigning labels to image patches and not just classifying the image as a whole. Traditionally, a fully convolutional network is used for image segmentation. 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 $n \times m$. Moreover, just using a feed-forward functional network (multi-layer perceptron) learns EEF1 allocations for smaller values of $n$, cannot learn as $n$ increases; we will briefly mention this in our Ablation Study in Section 5.1.

EEF1-NN contains series of convolution (contracting) and up-convolution (expanding) 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, i.e., tanh, which is applied element-wise. 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. We found that using a pooling layer or skip connections degraded the network performance. The final output represents the probability with which agent $i$ will receive item $j$. We apply softmax activation function across all agents for each item to ensure each item is allocated exactly once, i.e., $\forall j \in M \sum_{i \in N} A_i^\omega(\{j\}) = 1$. In total, we have 24 layers (convolution + up-convolution).

Using an FCN structure, we have a generalized network for $n \times m$; however, learning EEF1 allocations is not easy. We need to learn discrete variables, while neural networks are known for learning continuous output. We will describe these challenges in the next Section.

![Figure 1: EEF1-NN Architecture](image)

4.3 Training Details

There are certain challenges with network training, especially in the setting of indivisible resource allocation.

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. The training starts if a network learns to distribute an item equally among all agents, and the gradient vanishes. For a fair allocation, assigning an equal partition of each item is indeed an optimum. Converting these non-integral probabilities to integral allocation is non-trivial. Hence we set a temperature parameter in the softmax layer of the network to prevent getting stuck at such optima. Let $p_{j} = \{p_{j1}, \ldots, p_{jn}\}$ denote the output of our network before the final layer which represents the probability of assigning item $j$ to all the agents. The final allocation for agent $i$, i.e., $A_i^\omega(\{j\})$ is given by,

$$
A_i^\omega(\{j\}) = \text{softmax}(p_{j}) = \frac{e^{p_{ji}/T}}{\sum_{k=1}^{n} e^{p_{jk}/T}}
$$

It is common to start with a large temperature value for initial exploration and gradually reduce the temperature to reach the global optima. While training, when we set the temperature value 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 of $T$ too low, the output is exact discrete allocations, and there is no learning because of the vanishing gradients [11]. So we appropriately choose $T$ based on our experiments. Once the network learns, we set the parameter low enough to ensure discrete allocations.
We train seven networks with varied $\lambda$ with different training data sets, however in our case, varying $(\alpha, \beta)$ and chores. We will use that notation for all social welfare (positive/negative) for a combination of goods and chores. We set the temperature parameter to 0.01. We train our network for 1000 epochs. We use channelled input and feed to the network. We sample $10^3$ training samples, and we sample $10^2$ testing samples for each setting. We transform these valuations into six-channelled input and feed to the network. We set the temperature parameter to 0.01. We train our network for 1000 epochs. We use our Lagrangian loss as the objective function to train our network. We train seven networks with varied $\lambda \in [0.1, 2]$ and bag them for enhanced performance. The training process took 5-6 hours to train a single network using GPU. We are training the network for $10 \times 20$ and $13 \times 26$ for goods, chores, and combinations, so in total, we have 300k training samples, and we sample 10k testing samples for each setting. We illustrate the effect of specific hyper-parameters in the per-

5 EXPERIMENTS AND RESULTS

For reporting the network performance on the test set, we define the following two metrics, one is the measure of fairness and the other of efficiency.

Evaluation Metrics.

1. $\alpha_{\text{EF1}}^{\text{ALG}}$ - It measures the probability with which an algorithm $\text{ALG}$ outputs, EF1 allocation. $\alpha_{\text{EF1}}$ is the ratio of the number of samples that are EF1 to the total number of samples.

2. $\beta_{\text{USW}}^{\text{ALG}}$ - It measures the ratio of expected USW of an algorithm $\text{ALG}$ by expected USW of MUW allocation. $\beta_{\text{USW}}^{\text{ALG}} = \mathbb{E}(\text{sw}^{\text{ALG}})/\mathbb{E}(\text{sw}^{\text{MUW}})$.

Using the above metrics, we conduct the following ablation study to set appropriate hyper-parameters. Our network is trained across various types of items (goods or chores) and types of distribution. The test set consists of 10k samples. Note that $\beta_{\text{USW}}^{\text{ALG}} \in [0, 1]$ for goods, $\beta_{\text{USW}}^{\text{ALG}} \geq 1$ for chores, and will depend on the overall social welfare (positive/negative) for a combination of goods and chores. We will use that notation $(\alpha_{\text{EF1}}, \beta_{\text{USW}})$ to write network/algorithms’s performance.

5.1 Ablation Study

We illustrate the effect of specific hyper-parameters in the performance of EEF1-NN in Fig. [2,3]. We sample the valuations from the uniform distribution, set $n = 10$, only goods, for all the ablation study experiments, and observe the $\alpha_{\text{EF1}}$ as $m$ increases. In the plots, the red line with the label EEF1-NN denotes the $\alpha_{\text{EF1}}$ for optimal parameters. Corresponding to EEF1-NN, a single network from this bagged network is labeled as Single Network in the graph. This Single Network trained with six-channelled input, $\lambda = 1$, and temperature $T = 0.01$ is the baseline to compare across this ablation study. Only one parameter is changed w.r.t. the Single Network training for all the networks used in this study. We only compare $\beta_{\text{USW}}$ for the network across varied $\lambda$ values, as $\alpha_{\text{EF1}}$ values are close to the Single Network.

(i) Effect of Temperature $T$. Keeping other parameters fixed, we vary the $T = \{1, 0.1, 0.001\}$ in Fig. 2. When $T = 1$, our network converges to global optima, i.e., fractional allocation, unable to learn EEF1 discrete allocation represented by the blue line at the bottom of the plot. Also, we empirically observed that when networks learn to allocate an equal fraction of an item among agents (0.1 in case of 10 agents), the gradient vanishes, thus stuck in a bad local optimum. When $T = 0.001$ (violet line), it is too low, and performs sub-optimally compared to when $T = 0.01$ in single network (no bagging) (light blue line). We also noticed that the network’s 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.
(ii) Effect of series of Conv-UpConv layers. We select three series of Conv-UpConv for EEF1-NN as illustrated in Fig. 2. We plot the $\alpha_{EF1}$ for one Conv-UpConv series green dashed line. It is less than what we obtain for a 2-series red dotted line, which is less than the optimal 3-series (light blue line). As seen from Fig. 2, an increase between 1-series and 2-series is significant compared to 2-series and 3-series (single network without bagging). The complexity of the network having 4-series is far more than the performance improvement. We have limitations over the number of layers in Conv-UpConv, as we are working with a low dimensional matrix, such as $10 \times 20$, and having such a series increases performance. Briefly stating, while training a 4-layered feed-forward fully functional network for $10 \times 20$, $\alpha_{EF1}$ was roughly 0.008.

(iii) Effect of loss function. We analyze how different envy definitions in our loss function represented in Eq. 3 affects the training of EEF1-NN. As shown in Fig. 2, when we train our network using EF, i.e., Eq. 1 (Single Network, light blue line), the network performs significantly better than when trained using EF1, i.e., Eq. 2 (orange dashed line). For example, for $10 \times 20$, the performance of Single Network is (0.3358, 17.9611), whereas the performance of the EF1 trained network is (0.1530, 17.8708). Given a distribution, one way of interpreting this behavior can be that when we train the neural network to maximize social welfare w.r.t. to Envy-free, the best fairness it can have is EEF1 while maintaining efficiency close to MUW.

iv) Number of Input Channels. When training with just one channel input, i.e., valuations, the results we obtained were quite poor; for $10 \times 20$, we get (0.2113, 17.8976) as shown in Fig 3. Thus we changed our input representation into more channels. We tested for 2, 6, and $(n+1)$ channels for $n=10$. For two channel input, we set the first channel of input tensor as the valuation matrix while training a 4-layered feed-forward fully functional network for $10 \times 20$, $\alpha_{EF1}$ was roughly 0.008.

v) Effect of Bagging. We try different combinations of networks, each trained for varied $\lambda$ values. The Lagrangian Loss, as described in Eq. 4, $\lambda$ corresponds to the fairness constraint. More the value of $\lambda$, more penalty is given to envy in the loss. When $\lambda$ is too small, the penalty for allocating an unfair allocation is less, so the network learns a more efficient but less fair allocation. As we increase $\lambda$ up to a certain value, the network learns less efficient but more fair allocations. Beyond a certain value, if we increase $\lambda$, we get a degraded performance overall. In Fig. 3, we compare $\alpha_{EF1}$ and $\beta_{SW}$ of Single Network ($\lambda=1$), lambda=0.5, and lambda=0.1. We observed that varying $\lambda$ value results in converging to the different optimum. We bagged seven networks trained on with $\lambda$ values of $\lambda \in [0.1, 2]$. We choose a mix of (low efficiency, high fairness) and (high efficiency, low fairness). We feed six-channeled input to EEF1-NN, and it outputs the fairest and efficient, i.e., if more than one network gives EF1 allocation, then it will select the one with maximum social welfare. In Fig. 2, we find combining the networks (red line) outperforms the performance of a single network (light blue line).

5.2 Experiment Details and Observations

We select the best training parameters for EEF1-NN based on the above ablation study for the following experiments. We conduct three types of experiments to compare existing approaches across, Exp1: Different kinds of resources, Exp2: Different input distributions, and Exp3: Scalability to samples with large $n$. In all three experiments, we compare EEF1-NN with the following existing methods,

- MUW Since we don’t have Optimal EEF1 allocations to compare our results, we compare our results with MUW allocations. We also analyze after which value of $m$, an MUW allocation is likely to be EF1.
- Round Robin (RR) [21] finds EF1 allocations under additive valuations for pure goods and pure chores. Double Round Robin (D-RR) [3] extends RR to find EF1 allocation for the case with a combination of goods and chores under additive valuations.
- CRR Based on paper [4], we implement CRR to find RB sequence such that it allocate items to the agent who values it the most for goods. As mentioned in [4], an RB sequence is EF1 when all items have positive valuations.

Note that approaches like using parametric ILP solver [14] and the algorithm provided by [5] are exponential. Therefore, it is infeasible to use these for the configurations we report our results on, so we do not compare them.

Further, in Table 1, we study the convergence of different approaches towards EF1 for uniform distribution, i.e., after which value of $m$, a method converges to EF1.

EXP1: Performance across different resources for Uniform Distribution. For $n=10$, we compare the $\alpha_{EF1}$ in Fig. 4 (a1, b1, c1) and $\beta_{SW}$ in Fig. 4 (a2, b2, c2) as $m$ increases across the existing approach.

Irrespective of the resource type, as the number of items increases, all the approaches will move closer to EF1. We observe that MUW allocation (blue dotted line) converges towards EF1 allocations much faster for chores or combinations than goods. While Round Robin converges to EF1 allocations 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 $\alpha_{EF1}$ than MUW allocation and better $\beta_{SW}$ than RR/CRR.

The baseline RR is designed to find EF1 allocations and hence, by construct has $\alpha_{EF1}^{RR} = 1$. We observe that $\alpha_{EF1}^{CRR} = 1$ is close to $\alpha_{EF1}^{RR}$ after a certain $m$. At the same time, the allocation returned by EEF1-NN is far more efficient than RR as represented by the $\beta_{SW}$ values. (Fig 4 (a2, b2, c2)). Though $\alpha_{EF1}^{CRR} = 1$ (Fig. 4 a1), note that the baseline CRR only works for goods. Even for goods, we observe that compared to CRR, EEF1-NN obtains marginally better $\beta_{SW}$, in Fig. 4(a2).

EXP2: Performance across different distributions. We provide the performance of EEF1-NN when the valuations are sampled from different distributions such as Gaussian ($\mu=0.5, \sigma=1$) in Fig 5(a1, a2),
Analysis of Convergence to EEF1 Allocations (Uniform Distribution)

Definition 5.1 ($m^*(n)$). For a given $n$, we say an algorithm converges to EEF1 allocation at $m^*(n)$ if \( m > m^*(n) \),

(i) For goods: \( \alpha^\text{ALG}_{\text{EF1}} \geq 0.99 \) and \( \beta^\text{ALG}_{SW} \geq 0.99 \).

(ii) For chores: \( \alpha^\text{ALG}_{\text{EF1}} \leq 1.02 \).

We empirically study the value of $m^*(n)$ after which EEF1-NN, RR, and MUW start converging towards EEF1 for goods/chores for uniform distribution in Table 1. We don’t report CRR in this Table; as we see fluctuations in \( \beta^\text{CRR}_{SW} \), it doesn’t increase smoothly in Fig. [46]; However, note that CRR results better than RR for goods, and EEF1-NN performs marginally better than CRR. We observe that in the case of goods, EEF1-NN reaches close to EEF1 allocations faster than MUW and RR, and RR reaches close to EEF1 faster than MUW.

As seen in Table 1, EEF1-NN converges first, then MUW, and finally RR in the case of chores. We report the value of $m^*$ for RR we use \( \beta^\text{CRR}_{SW} \leq 1.064 \) since these $m$ values are already significantly high than MUW and RR, concluding that RR will converge after a considerably larger $m$. We also observed as $m$ increases, \( \alpha^\text{EF1}, \alpha^\text{EFX} \), and \( \alpha^\text{EF} \) of MUW keeps getting closer. For example, for $9 \times 530$ goods uniform distribution, \( \alpha^\text{EF1} = 0.989, \alpha^\text{EFX} = 0.9834 \), and \( \alpha^\text{EF} = 0.9834 \); while for $9 \times 200$ goods uniform distribution, \( \alpha^\text{EF1} = 0.6436, \alpha^\text{EFX} = 0.5086 \), and \( \alpha^\text{EF} = 0.5032 \).

Note that the actual value of $m^*(n)$ may be slightly different from the exact point of convergence mentioned in Table as we do not perform experiments for all possible values of $m$. Our goal here is to observe a pattern among approaches to compare the different approaches to achieve EEF1.

Table 1: Value of $m^*(n)$ 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 $\alpha^\text{EEF1-NN} \leq 1$ much faster than $\alpha^\text{MUW}_{\text{EEF1}}$, and $\beta^\text{EEF1-NN}_{SW} \leq \beta^\text{MUW}_{SW}$ much faster than RR, D-RR, CRR. So the results from EEF1-NN show a better trade-off between EEF1 and efficiency than the existing approaches for different input distributions. We trained our network with fixed $n \times m$ and goods or/and chores, still the performance scales for any $m$ and a large $n$. For smaller $n$ and $m$, one can use integer programming or any
pseudo-polynomial approach, and when \( m \gg n \). We observed that MUW converges towards EEF1 faster than RR in goods, while in chores, it’s the other way around. Hence we conclude that our approach effectively learns and provides a better trade-off when \( m \) is not too large or very small compared to the \( n \) but is in a specific range.

6 CONCLUSION

In this paper, we addressed finding fair and efficient allocations for goods, chores, or combinations. In general, the problem is NP-hard. We proposed a neural network EEF1-NN to find EEF1 allocations. To overcome the issue of finding optimal discrete allocations, we designed appropriate architecture and input representation combined with other training heuristics. We studied the effect each proposed constituent has on performance. Our experiments demonstrated the efficacy of EEF1-NN for different input distributions over existing approaches. It finds reasonably fair and close to optimal solutions in real-time. Can we improve it further?

REFERENCES

[1] Georgios Amanatidis, Evangelos Markakis, Afshin Nikzad, and Amin Saberi. 2017. Approximation Algorithms for Computing Maximin Share Allocations. ACM Transactions on Algorithms 13, 4 (Dec 2017), 1–28.

[2] Haris Aziz, Páter Biró, Jérôme Lang, Julien Lesca, and Jérôme Monnot. 2016. Optimal Reallocation under Additive and Ordinal Preferences. In Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems (Singapore, Singapore) (AAMAS ’16). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 402–410.

[3] Haris Aziz, Ioannis Caragiannis, and Ayumi Igarashi. 2018. Fair allocation of combinations of indivisible goods and chores. (2018). http://arxiv.org/abs/1807.10684

[4] Haris Aziz, Xin Huang, Nicholas Mattei, and Erel Segal-Halevi. 2019. The Constrained Round Robin Algorithm for Fair and Efficient Allocation. arXiv:1908.00161 [cs.GT]

[5] Haris Aziz, Xin Huang, Nicholas Mattei, and Erel Segal-Halevi. 2020. Computing Fair Utilitarian Allocations of Indivisible Goods. (2020). https://arxiv.org/abs/2012.03979

[6] Haris Aziz, Hervé Moulin, and Fedor Sandomirskiy. 2019. A polynomial-time algorithm for computing a Pareto optimal and almost proportional allocation.
