

# Learning Optimal Redistribution Mechanisms Through Neural Networks

P Manisha

International Institute of Information  
Technology, Hyderabad, India.  
manisha.padala@research.iiit.ac.in

C V Jawahar

International Institute of Information  
Technology, Hyderabad, India.  
jawahar@iiit.ac.in

Sujit Gujar

International Institute of Information  
Technology, Hyderabad, India.  
sujit.gujar@iiit.ac.in

## ABSTRACT

We consider a social setting where  $p$  public resources/objects are to be allocated among  $n$  competing and strategic agents so as to maximize social welfare (the objects should be allocated to those who value them the most). This is called allocative efficiency (AE). We need the agents to report their valuations for obtaining these resources, truthfully referred to as dominant strategy incentive compatibility (DSIC). Typically, we use auction-based mechanisms to achieve AE and DSIC. However, due to Green-Laffont Impossibility Theorem, we cannot ensure budget balance in the system while ensuring AE and DSIC. That is, the net transfer of money cannot be zero. This problem has been addressed by designing a redistribution mechanism so as to ensure minimum surplus of money as well as AE and DSIC. The objective could be to minimize surplus in expectation or in the worst case and these  $p$  objects could be homogeneous or heterogeneous. Designing of such mechanisms is non-trivial. Especially, designing redistribution mechanisms which perform well in expectation becomes analytically challenging for heterogeneous settings.

In this paper, we take a completely different, data-driven approach. We train a neural network to determine an optimal redistribution mechanism based on given settings with both the objectives, optimal in expectation and optimal in the worst case. We also propose a loss function to train a neural network to optimize worst case. We design neural networks with the underlying rebate functions being linear as well as nonlinear in terms of bids of the agents. Our networks' performances are same as the theoretical guarantees for the cases where it has been solved. We observe that a neural network based redistribution mechanism for homogeneous settings which uses nonlinear rebate functions outperforms linear rebate functions when the objective is optimal in expectation. Our approach also yields an optimal in expectation redistribution mechanism for heterogeneous settings.

## KEYWORDS

VCG Mechanisms; Dominant Strategy Incentive Compatibility (DSIC); Redistribution Mechanism; Neural Networks; Deep Learning

## 1 INTRODUCTION

In this paper, we address the problem of allocating public resources/objects among multiple agents who desire them. These strategic agents

have their private values for obtaining the resources. The allocation of the objects should be such that the society, as a whole, gets the maximum benefit. That is, the agents who value these resources the most should get them. This condition is referred to as *Allocative Efficient (AE)*. To achieve this, we need the true valuations of the agents, which the strategic agents may misreport for personal benefit. Thus, there is a need for an auction-based mechanism. A mechanism ensuring truthful reporting, is called *Dominant Strategy Incentive Compatible (DSIC)*. The classical Groves mechanisms [11] satisfy both of these properties.

Groves mechanisms achieve DSIC by charging each agent an appropriate amount of money known as Groves' payment rule. The most popular among Groves mechanism is VCG mechanism [3, 11, 32]. Use of VCG mechanism results in collection of money from the agents. It should be noted that our primary motive in charging the agents is to elicit their true valuations and not to make money from them as the objects are public. Hence, we need to look for the other Groves mechanisms. Moreover, the mechanism cannot fund the agents. Thus, we desire a mechanism that incurs neither deficit nor surplus of funds; it must be *Strictly Budget Balanced (SBB)*. However, due to Green-Laffont Impossibility Theorem [10], no mechanism can satisfy AE, DSIC, and SBB simultaneously. Thus, any Groves payment rule always results in either surplus or deficit of funds.

To deal with such a situation, Maskin and Laffont [26] suggested that we first execute VCG mechanism and then redistribute the surplus among the agents in a manner that does not violate DSIC. This mechanism is referred to as a *Groves' redistribution mechanism* or simply *redistribution mechanism (RM)* [12, 15] and the money returned to an agent is called its *rebate*. The rebates are determined through *rebate functions*. Thus, designing a RM is same as designing an underlying rebate function.

In the last decade, a lot of research focused on dealing with the Green-Laffont Impossibility theorem and on designing an optimal redistribution mechanism (RM) that ensures maximum possible total rebate [2, 4, 6, 12–17]. An optimal RM could be optimal in expectation or optimal in the worst case. The authors of [16, 19, 27] address the problem of finding the optimal RM when all the objects are identical (homogeneous). Guo and Conitzer [18, 19] model an optimization problem and solve for an optimal linear rebate function which guarantees maximum rebate in the worst-case (WCO) and optimal in expectation for a homogeneous setting. The authors of [12, 15] extend WCO to a heterogeneous setting where the objects are different and propose a nonlinear rebate function called as *HETERO*. Despite HETERO being proved to be optimal for unit demand in heterogeneous settings, in general, it is challenging to come up with a nonlinear RM analytically. Analytical solutions

for an optimal in expectation RMs in heterogeneous settings is elusive. Moreover, the possibility of a nonlinear rebate function which is optimal in expectation for homogeneous setting has not been explored yet. Thus, there is a need for a new approach towards designing RMs. In this paper, we propose to use neural networks and validate its usefulness.

Neural networks have been successful in learning complex, non-linear functions accurately, given adequate data [22]. There is a theoretical result which states a neural network can approximate any continuous function on compact subspace of  $\mathbb{R}^n$  [23]. However, designing such a network has been elusive until recent times. The latest theoretical developments ensure that stochastic gradient descent (SGD) converges to globally optimal solutions [24, 30]. In recent times, with advent in computing technology, neural networks have become one of the most widely used learning models. They have outperformed many of the traditional models in the tasks of classification and generation etc. [9, 21]. In the context of game theoretic mechanism design, Dütting *et.al.* [5] have proposed neural network architectures for designing optimal auctions. Converse to ours', their goal is to design the network for maximizing the expected revenue. Tang [31] uses deep reinforcement learning for optimizing a mechanism in dynamic environments. Their goal, unlike ours is not to design a mechanism, but to find the optimal parameters for the existing mechanism. Our goal in this paper is to study: Can we train neural networks to learn optimal RMs with the help of randomly generated valuation profiles, rather than analytically designing them? The following are our contributions.

*Contributions.* To the best of our knowledge, this paper is a first attempt towards learning optimal redistribution mechanisms (RM) using neural networks.

- To begin with, we train neural networks for the settings where researchers have designed RMs analytically. In particular, we train networks, OE-HO-L and OW-HO-L for optimal in expectation for homogeneous settings with linear rebate functions and optimal in the worst case for homogeneous settings with linear rebate functions respectively. Both the neural networks match the performance of theoretically optimal RMs for their respective settings.
- Next, we train a network, OW-HE-NL to model nonlinear rebate function for optimal in worst case RM in heterogeneous settings, discarding the need to solve it analytically. Note that, traditionally, neural networks have been mostly used for stochastic approximation of an expectation, but our model is also able to learn a worst-case optimal RM as well.
- Motivated by the network performance in above, we train OE-HO-NL, an optimal in expectation RM with nonlinear rebate function for the homogeneous setting. We find that this model ensures greater expected rebate than the optimal in expectation RM with linear functions, proposed by Guo and Conitzer [18].
- We also train OE-HE-NL, an optimal in expectation for heterogeneous setting with nonlinear functions and we experimentally observe that it's performance is reasonable.

*Organization.* In Section 2, we describe the related work. In Section 3, we explain the notation used in the paper and preliminaries of

Groves redistribution mechanism. We develop our neural network approach in Section 4. We discuss the training of neural networks and the experimental analysis in Section 5. We conclude the paper with the possible future directions in Section 6.

## 2 RELATED WORK

In this section, we discuss the research related to our work.

*Auction Based Mechanism.* To obtain the required private information from strategic agents truthfully, *mechanism design theory* is developed [7, 28]. The key idea is to charge the agent appropriately to make mechanisms truthful or DSIC. The most popular auction-based mechanisms are VCG and Groves mechanisms [3, 11, 32] which satisfy the desirable properties, namely, allocative efficiency (AE) and dominant strategy incentive compatibility (DSIC). Another desirable property is the net transfer of the money in the system should be zero, i.e., it should be *strictly budget balanced* (SBB). Green and Laffont [10] showed no mechanism can satisfy AE, DSIC, and SBB simultaneously. As we cannot compromise on DSIC, we must compromise on one of AE or SBB.

Faltings [6] and Guo and Conitzer [17] achieved budget balance by compromising on AE. Hartline and Roughgarden [20] proposed a mechanism that maximizes the sum of the agents' utilities in expectation. Clippel *et. al.* [4] used the idea of destroying some of the items to maximize the agents' utilities leading to approximately AE and approximately SBB. A completely orthogonal approach was proposed by Parkes *et. al.* [29], where the authors propose an optimization problem which is approximately AE, SBB and though not DSIC, it is not easy to manipulate the mechanism. However, an aggressively researched approach is to retain AE and DSIC and design mechanism that is as close to SBB as possible. These are called *redistribution mechanisms* (RM).

*Redistribution Mechanisms.* Maskin and Laffont [26] first proposed the idea of redistribution of the surplus as far as possible after preserving DSIC and AE. Bailey [1], Cavallo [2], Moulin [27], and Guo and Conitzer [16] considered a setting of allocating  $p$  homogeneous objects among  $n$  competing agents with unit demand. and Guo and Conitzer [19] generalized their work in [16] to multi-unit demand to obtain worst-case optimal (WCO) RM. In [18], Guo and Conitzer designed RM that is optimal in expectation for homogeneous settings.

Gujar and Narahari [12] proved that no linear RM can assure non-zero rebate in worst case and then generalized WCO mechanism mentioned above to heterogeneous items, namely HETERO. Their conjecture that HETERO was feasible and worst-case optimal was proved by Guo [15] for heterogeneous setting with unit-demand.

## 3 PRELIMINARIES

Let us consider a setting comprising  $p$  public resources/objects and  $n$  competing agents who assign a certain valuation to these objects. Each agent desires at most one out of these  $p$  objects. These objects could be homogeneous, in which case, agent  $i$  has valuation  $v_i = \theta_i$  for obtaining any of these  $p$  resources. It could also be the case that the objects are distinct or heterogeneous and each agent derives different valuation for obtaining different objects ( $v_i = (\theta_{i1}, \theta_{i2}, \dots, \theta_{ip})$ ). These objects are to be assigned to those

who value it the most, that is, it should be allocatively efficient (AE). The true values  $\mathbf{v} = (v_1, v_2, \dots, v_n)$  that the agents have for the objects are based on their private information  $\theta = (\theta_1, \theta_2, \dots, \theta_n) = (\theta_i, \theta_{-i})$  and the strategic agents may report them as  $(\theta'_1, \dots, \theta'_n)$ . In the absence of appropriate payments, the agents may boast their valuations. Hence, we charge agent  $i$  a payment  $m_i(\theta')$  based on the reported valuations.

We need to design a mechanism  $\mathcal{M} = (\mathcal{A}, \mathcal{P})$ , an allocation rule  $\mathcal{A}$  and a payment rule  $\mathcal{P}$ .  $\mathcal{A}$  selects an allocation  $k(\theta') \in K$  where  $K$  is the set of all feasible allocation and  $\mathcal{P}$  determines the payments. With this notation, we now explain the desirable properties of a mechanism.

### 3.1 Desirable Properties

One of our primary goals is to ensure allocative efficiency.

*Definition 3.1 (Allocative efficiency (AE)).* - A mechanism  $\mathcal{M}$  is *allocatively efficient* (AE) if it chooses in every given type profile, an allocation of objects among the agents such that sum of the valuations of the allocated agents is maximized. That is, for each  $\theta \in \Theta$ ,

$$k^*(\theta) \in \operatorname{argmax}_{k \in K} \sum_{i=1}^n v_i(k, \theta_i).$$

The most desirable property is that the agents should report their valuations truthfully to the mechanism. Formally, it is called *dominant strategy incentive compatibility* (DSIC).

*Definition 3.2 (Dominant Strategy Incentive Compatibility (DSIC)).* We say a mechanism  $\mathcal{M}$  to be *dominant strategy incentive compatible* (DSIC), if it is a best response for each agent to report their type truthfully, irrespective of the types reported by the other agents. That is,

$$v_i(k(\theta_i, \theta_{-i}), \theta_i) - m_i(\theta_i, \theta_{-i}) \geq v_i(k(\theta'_i, \theta_{-i}), \theta_i) - m_i(\theta'_i, \theta_{-i}) \\ \forall \theta'_i \in \Theta_i, \forall \theta_i \in \Theta_i, \forall \theta_{-i} \in \Theta_{-i}, \forall i \in N.$$

Given an AE allocation rule, Groves proposed a class of mechanisms known as *Groves' mechanisms* that ensure DSIC. Groves' payment rule is  $m_i(\theta) = - \sum_{j \neq i} v_j(k_*(\theta), \theta_j) + h_i(\theta_{-i})$ , where  $h_i$  is an arbitrary function of reported valuations of the agents other than  $i$ . Clarke's payment rule is a special case of Groves' payment rule where  $h_i(\theta_{-i}) = \sum_{j \neq i} v_j(k_{-i}^*(\theta_{-i}), \theta_j)$  where  $k_{-i}^*$  is an AE allocation when the agent  $i$  is not part of the system. Thus, Clarke's payment rule is give by Equation (1)

$$t_i(\theta) = \sum_{j \neq i} v_j(k_{-i}^*(\theta_{-i}), \theta_j) - \sum_{j \neq i} v_j(k^*(\theta), \theta_j) \quad \forall i = 1, \dots, n \quad (1)$$

And,  $m_i = t_i$ . This payment scheme is referred to as *VCG payment*. The total payment by all the agents is,  $t(\theta) = \sum_{i \in N} t_i(\theta)$

Another property we desire is budget balance condition.

*Definition 3.3 (Budget Balance condition or Strictly Budget balance (SBB)).* We say that a mechanism  $\mathcal{M}$  is *strictly budget balanced* (SBB) if for each  $\theta \in \Theta$ ,  $m_1(\theta), m_2(\theta), \dots, m_n(\theta)$  satisfy the condition,  $\sum_{i \in N} m_i(\theta) = 0$

It is weakly budget balanced if  $\sum_{i \in N} m_i(\theta) \geq 0$ .

One can implement AE allocation rule and charge the agents VCG payments. In the case of auctions settings, the seller collects the payments. In our setting, the goal is not to make money as these objects are public resources. However, in general, due to Green-Laffont Impossibility Theorem [10], no AE and DSIC mechanism can be strictly budget balanced. That is, the total transfer of money in the system may not be zero. So, the system will be either left with a surplus or incur deficit. Using Clarke's mechanism, we can ensure under fairly weak conditions, that there is no deficit of money (that is, the mechanism is weakly budget balanced) [3]. The idea proposed by Maskin and Laffont [26] is to design a payment rule as to first collect VCG payments and then redistribute this surplus (rebate) among the agents while ensuring DSIC. This leads to *Groves' Redistribution Mechanism*, in which the rebate is given by a rebate function  $r(\cdot)$ .

### 3.2 Groves' Redistribution Mechanism

Since SBB cannot coexist with DSIC and AE, we would like to redistribute the surplus to the participants as much as possible, preserving DSIC and AE. Such a mechanism is referred to as *Groves redistribution mechanism* or simply *redistribution mechanism*. Designing a redistribution mechanism involves designing an appropriate rebate function. We desire a rebate function which ensures maximum rebate (which is equivalent to minimum budget imbalance). In addition to DSIC, we want the redistribution mechanism to have following properties:

- (1) Feasibility (F): the total payment to the agents should be less than or equal to the total received payment.
- (2) Individual Rationality (IR): each agent's utility by participating in the mechanism should be non-negative.
- (3) Anonymity: rebate function is same for all the agents,  $r_i(\cdot) = r_j(\cdot) = r(\cdot)$ . This may still result in different redistribution payments as the input to the function may be very different.

While designing redistribution mechanism for either homogeneous or heterogeneous objects, we may have linear or nonlinear rebate function of the following form,

**THEOREM 3.4.** [12] *In the Groves redistribution mechanism, any deterministic, anonymous rebate function  $f$  is DSIC iff,*

$$r_i = f(v_1, v_2, \dots, v_{i-1}, v_{i+1}, \dots, v_n) \quad \forall i \in N$$

where,  $v_1 \geq v_2 \geq \dots \geq v_n$ .

*Definition 3.5. [Linear Rebate Function]* The rebates to an agent follow a *linear rebate* function if the rebate is a linear combination of bid vectors of all the remaining agents. Thus,  $r_i(\theta, i) = c_0 + c_1 v_{-i,1} + \dots + c_{n-1} v_{-i,n-1}$ .

There may exist a family of redistribution mechanisms which satisfy the above constraints, but the aim is to identify the one mechanism that redistributes the greatest fraction of the total VCG payment. To measure the performance of redistribution mechanism [19], defines redistribution index,

*Definition 3.6. [Redistribution Index]* The *redistribution index* of a redistribution mechanism is defined to be the worst case fraction of VCG surplus that gets redistributed among the agents. That is,

$$e^{ow} = \inf_{\theta: t(\theta) \neq 0} \frac{\sum r_i(\theta_{-i})}{t(\theta)}$$

**Table 1: Optimization problem formulation**

	OE	OW
Variables :	$c_0, c_1, \dots, c_{n-1}$	$e^{ow}, c_0, c_1, \dots, c_{n-1}$
Maximize :	$\mathbb{E} \sum_{i=1}^n r_i$	$e^{ow}$
Feasibility :	$\sum_{i=1}^n r_i \leq t$	$\sum_{i=1}^n r_i \leq t$
Other constraints		
For worst-case :		$\sum_{i=1}^n r_i \geq e^{ow} t$
IR :		$r_i \geq 0$

With the notation defined above and Green-Laffont Impossibility theorem in backdrop, we now explain existing redistribution mechanisms in the following subsection.

### 3.3 Optimal Redistribution Mechanisms

It may happen that, one mechanism might redistribute higher rebate at  $\theta_1$  and another mechanism at  $\theta_2$ . Hence, we use the two kinds of evaluation metrics defined to select a mechanism. One metric compares the rebate functions based on maximum expected total redistribution, to find the mechanism which is optimal in expectation. The other metric finds the optimal in worst-case redistribution mechanism, based on the lowest redistribution index it guarantees

**3.3.1 Optimal in Expectation. (OE):** If the prior distributions over agents' valuations are available we can compare the mechanisms based on total expected redistribution. In [18] the authors derive the mechanism and prove its optimality for homogeneous setting with linear rebate function. We define a redistribution index for OE setting as follows:

$$e^{oe} = \frac{\mathbb{E} \sum_i r_i(\theta_{-i})}{\mathbb{E} \sum_{\theta} t(\theta)}$$

Maximizing  $e^{oe}$  is equivalent to maximizing the expected total rebate. The authors formulated the problem as given in Table 1.

The OE objective for heterogeneous objects as well as with nonlinear rebate function has not been addressed yet.

**3.3.2 Optimal in Worst-case. (OW) :** The redistribution mechanism is better if it ensures higher rebate to the agents on average. In the absence of distributional information, we would evaluate a mechanism by considering the worst redistribution index that it guarantees. In [19] the authors gave the following model and analytically solved it for homogeneous setting with linear rebate functions. They also claim the worst-case optimal mechanism is optimal among all redistribution mechanisms that are deterministic, anonymous and satisfy DSIC, AE and F. The optimization problem is formulated as given in Table 1.

For heterogeneous setting, [12] defines a nonlinear redistribution mechanism which is called HETERO and [15] proves the optimality of HETERO. There is no optimal mechanism with linear rebate function for heterogeneous setting as established by the following theorem,

**THEOREM 3.7.** [12] *If a redistribution mechanism is feasible and individually rational, then there cannot exist a linear rebate function which is simultaneously DSIC, deterministic, anonymous and has non-zero redistribution index.*

Equipped with the knowledge of the existing approaches and above theorem, we describe our approach for designing the optimal rebate function using neural networks.

### 3.4 Our Approach

As mentioned, for a homogeneous setting the linear rebate functions that are OE and OW can be analytically found by formulating redistribution mechanism as a linear program. However, for heterogeneous settings, linear redistribution mechanism need not be a good choice (Theorem 3.7). Even though [15] has solved for redistribution mechanisms in heterogeneous by proving HETERO to be OW, an OE mechanism for heterogeneous settings has not been formulated yet. Moreover, OW mechanism (HETERO) is not simple to describe. In addition, for homogeneous settings, it is not known whether nonlinear redistribution mechanisms can do better than linear for OE objective.

In this paper, we address these issues, using a novel data-driven approach to approximate a rebate function for a given setting, without analytically solving for it. That is, we generate a large number of bid profiles randomly and train a neural network to determine the rebates for the agents so as to achieve the given objective, either OE or OW. We consider both the rebate functions, linear and nonlinear for homogeneous objects as well as heterogeneous objects. The choice of neural networks is largely motivated by the universal approximation theorem, which states that a feed-forward network with single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets.

## 4 DESIGN OF NEURAL NETWORKS

Neural Networks are biologically inspired paradigms which learn optimal functions from data. The main components that customize such a network for a specific task are its architecture and the objective function which guides its training. To design a rebate function which is OE or OW, we define an appropriate neural network in Section 4.2. We begin by describing an artificial neuron which is the fundamental processing unit of a neural network.

### 4.1 Basic Structure

An artificial neuron receives inputs  $x_0$  to  $x_m$ . If necessary, we apply nonlinear activation function  $\phi$  to obtain the output ( $y$ ) from the neuron. The activation function is for thresholding the output to introduce nonlinearity in the network. Thus the output of a neuron is,

$$y = \phi\left(\sum_{i=0}^m w_i x_i\right)$$

where  $w_i$  is the weight for  $i^{th}$  input.

In any general network, we connect the neurons together in a specific and useful manner. The neurons are grouped into primarily three layers, input layer, hidden layer, and output layer. The weights are randomly initialized before training. Given a set of training input and output pairs, the model compares its own output with the desired output and tries to learn the optimal set of weights by back-propagating the error through the network. In our case, we do not have a desired output, but we have an objective function that is to be maximized. That is, we need to determine optimal

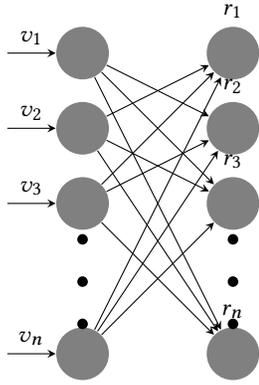


Figure 1: Network model

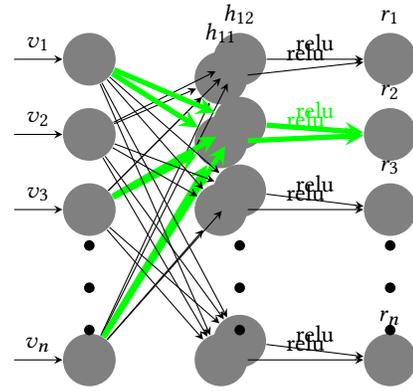


Figure 2: Nonlinear Network model

weights such that the rebate function is OE or OW. In addition, our mechanism should be Feasible and Individual Rational. The total VCG payment by the agents is  $t$ , and the neural network parameters are  $(w, b)$  then,

- Feasibility :  $g(w, b) : t - \sum_{i=1}^n r_i(w, b) \geq 0$
- Individual Rationality :  $g'(w, b) : r_i(w, b) \geq 0, \forall i \in N$

The above inequality constraints are added to the loss function during training. Having defined a general network and the constraints, we define the specific design of the network that we use.

## 4.2 Neural Network Architecture

**4.2.1 Linear rebate function :** To model the linear rebate function as given by Definition 3.5, we use a network consisting of neurons with  $n$  input and  $n$  output nodes without any activation function. The input nodes represent agents' valuations and output nodes represent their rebate. As required by Theorem 3.4, a rebate function for an agent  $i$  should depend only on valuations of the remaining agents. Hence, we connect the  $i^{th}$  output node to all the input nodes except  $i^{th}$  input node as shown in Figure 1. We used a total of  $n-1$  weights and 1 bias. Since the weights and the bias used is same for calculating the rebate of each agent (represented by each node in the output layer) we ensure that the  $r()$  is anonymous. In addition to the weights ( $w$ ) which model the  $c_1$  to  $c_n$  in  $r()$ , there is a same bias added ( $b$ ), which models the  $c_0$ , to each output.

$$r_i = \sum_{j=1}^{n-1} v_j w_j + b, \forall i = 1 \rightarrow n$$

**4.2.2 Nonlinear rebate function :** The network consists of neurons with  $n$  input and  $n$  output nodes and one hidden layer. The input nodes represent agent valuations and output nodes represent the rebate. The neurons in the input and hidden layer use ReLU activation which returns 0 if the output of that neuron is negative else returns the output itself. [12] defines the optimal nonlinear rebate function as the combination of marginal payments. We believe that the rebate functions though nonlinear, should only contain first degree terms for bid values as payments do not have higher order terms, making the function piece-wise linear. Hence, we use ReLU as our activation function.

As required by Theorem 3.4, a rebate function for an agent  $i$  should depend only on valuations of the remaining agents. Hence, we connect the  $i^{th}$  output node to the  $i^{th}$  layer of hidden nodes which are connected to all the input nodes except the  $i^{th}$  input node as shown in Figure 2. The redistribution function being anonymous, the weights of the connections entering each hidden layer and each output are the same. The green thick lines in Figure 2 represent a unique set of weights which are connected to the second agents' output. The same weights are used for calculating the rebate of the other agents as well. The first set of weights ( $w$ ) connect the input nodes to the hidden nodes and bias ( $b$ ) is added to the hidden nodes. The second set of weights ( $w'$ ) connect the hidden nodes to the output nodes and same bias added ( $b'$ ) to each output node.

$$r_i = \sum_{k=1}^h \text{relu}\left(\sum_{j=1}^{n-1} v_j w_{jk} + b\right) w'_k + b', \forall i = 1 \text{ to } n,$$

$h$ : Number of hidden neurons,  $\text{relu}(x) = \max(0, x)$

The defined network architectures can model different functions depending on the weights. Hence, the training of the network guides the network to learn appropriate weights. Prior to training, we must recall Theorem 3.4 which necessitates the ordering of the input valuations. Besides the ordering, the constraints defined in Section 4.1 require the evaluation of  $t$  the VCG Payment. In the following section, we mention the details about the same.

## 4.3 Ordering of Inputs and Payments

The ordering and calculation of VCG payment in both homogeneous and heterogeneous cases are independent of the neural network.

**4.3.1 Homogeneous Objects :** All the given  $p$  objects are similar and each agent desires at most one object. The bids submitted are  $\theta$  where,  $\theta \in \mathbb{R}^n$ . We order the bids such that  $v_1 \geq v_2 \geq \dots \geq v_n$ . Payment by agent  $i$ ,

$$t_i = \begin{cases} v_{p+1} & i \leq p \\ 0 & i > p \end{cases}$$

Hence,  $t = pv_{p+1}$ .

**4.3.2 Heterogeneous Objects :** All the  $p$  objects are different, each agent will submit his valuation for each of the objects. The bids

submitted are  $\theta$  where  $\theta \in \mathbb{R}^{p \times n}$ . We define a particular ordering among these vectors based on the overall utility of each agent and the marginal valuations they have for each item. The allocation of the goods is similar to a weighted graph matching problem and is solved using Hungarian Algorithm. Once we get the allocation say,  $k^*$ , we proceed to calculate the payments using the VCG payment  $t = \sum_{i \in N} t_i$ , where each  $t_i$  is given by Equation (1). The ordering of bids for the winning  $p$  agents is determined based on their utilities. The utility  $u_i$  of agent  $i$  is given by,

$$u_i = \sum_{j \in N} v_j(k_*(\theta), \theta_j) - \sum_{j \neq i} v_j(k_{-i}^*(\theta_{-i}), \theta_j), \forall i = 1, \dots, n.$$

If two agents have same utility, their ordering is determined by their marginal values for the first item, and if it is same, then by second item and so on. Once their ordering is determined, we remove the  $p$  agents and then run the VCG mechanism to get the next  $p$  winning agents and calculate the ordering using the same procedure as above. If the remaining agents are less than  $p$  we can still find the allocation and hence order the remaining agents till none are left or one is left. The time complexity of this ordering is polynomial in  $np$ .

With the given ordering of the inputs and payments, we use the specified network models in both homogeneous and heterogeneous settings. For each setting, we model either OE or OW mechanism. For both OE and OW the network architecture remains same whereas the objective changes as defined in the following section.

## 4.4 Objective Function

During the forward pass the input valuations are multiplied by the network weights and the corresponding rebate for each agent is calculated. The initial weights being random, the rebate calculated will not be optimal. In order to adjust the weights to obtain optimal rebate function, we add an objective at the end of the network which maximizes the rebate in both OW and OE. The loss function essentially is the negative total rebate of all the agents. The objective also takes care of the Feasibility 4.1 and Individual Rationality condition 4.1

### 4.4.1 Optimal in Expectation (OE).

- Given that we need to maximize the total expected rebate, the loss is defined as:

$$l(w, b) : \frac{1}{T} \sum_{j=1}^T \sum_{i=1}^n -r_i^j,$$

$T$  =total number of training samples

- Given Inequality constraint for Feasibility 4.1 we modify it to equality as:

$$G^j(w, b) = \max(-g(w, b), 0), \forall j = 1, 2, \dots, T$$

- The overall loss function:

$$L(w, b) = l(w, b) + \frac{\rho}{2} \sum_{j=1}^T G^j(w, b)^2 \quad (2)$$

### 4.4.2 Worst case Optimal (OW).

- Given that in OW we are trying to maximize the worst possible redistribution index, as defined in Definition 3.6, the loss is given by:

$$l(k) : -k$$

$$\text{such that } g_3 : \sum_{i=1}^n r_i - kt \geq 0$$

$T$  = total number of training samples

- Given inequality constraint for Feasibility 4.1 we modify it to equality as:

$$G_1^j(w, b) = \max(-g(w, b), 0)$$

Given inequality constraint for IR 4.1 we modify it to equality as:

$$G_2^j(w, b) = \max(-g'(w, b), 0)$$

The inequality condition for finding the worst case optimal is modified as follows

$$G_3^j(w, b) = \max(-g_3(w, b), 0)$$

$\forall j = 1, 2, \dots, T$

- The overall loss function for worst case:

$$L(w, b, k) = l(k) + \frac{\rho}{2} \sum_{j=1}^T [G_1^j(w, b)^2 + G_2^j(w, b)^2 + G_3^j(w, b)^2] \quad (3)$$

The network and objective together can be used to model any mechanism which is either OW or OE. Taking the cue from Section 3.3 we conduct few experiments in order to learn an optimal mechanism which was analytically solved in theory for homogeneous settings. Further, our experiments for heterogeneous settings with objective OW, try to model HETERO [12] and also OE nonlinear mechanisms for homogeneous setting.

## 5 IMPLEMENTATION DETAILS AND EXPERIMENTAL ANALYSIS

The proper training of neural networks is very crucial for its convergence. Xavier initialization and Adam optimization guide us in choosing appropriate initialization and optimizer which are crucial for stabilizing the network. Given that we have two different networks and two different objectives, we experimented on various combinations of these to validate the data-driven approach with existing results. In the following subsections, we specify the implementation details.

### 5.1 Initialization and Optimizer

**5.1.1 Xavier initialization:** [8] The right way of initialization of a neural network is to have weights that are able to produce outputs that follow a similar distribution across all neurons. This will greatly help convergence during training and we will be able to train faster and effectively. Xavier initialization tries to scale the random normal initialized weights with a factor  $\alpha$ , such that there is unit variance in the output.  $\alpha = \frac{1}{\sqrt{n}}$ , where  $n$  is the number

**Table 2:  $e^{oe}$  for homogeneous and heterogeneous setting.**

$n, p$	Homogeneous			Heterogeneous	
	OE-HO Theoretical	OE-HO Linear NN	OE-HO Nonlinear NN	OE-HE Linear NN	OE-HE Nonlinear NN
3,1	0.667	0.668	0.835	0.667	0.835
4,1	0.833	0.836	0.916	0.834	0.920
5,1	0.899	0.901	0.961	0.900	0.969
6,1	0.933	0.933	0.973	0.934	0.970
3,2	0.667	0.665	0.839	0.458	0.774
4,2	0.625	0.626	0.862	0.637	0.855
5,2	0.800	0.802	0.897	0.727	0.930
6,2	0.875	0.875	0.935	0.756	0.954
10,1	0.995	0.996	0.995	0.995	0.995
10,3	0.943	0.945	0.976	0.779	0.923
10,5	0.880	0.880	0.947	0.791	0.897
10,7	0.943	0.944	0.976	0.781	0.857
10,9	0.995	0.997	0.996	0.681	0.720

of input connection entering in that particular node.  $\alpha = \sqrt{\frac{2}{n}}$  for ReLU.

**5.1.2 Adam optimizer:** [25] Adam is a first order gradient-based optimization of stochastic objective functions. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. The default values provided in the tensorflow library is used for  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$  and  $\epsilon = 1e - 08$ . The learning rates are different for different cases as mentioned in their respective sections.

## 5.2 Different Settings for Training

**5.2.1 Optimal in Expectation for Homogeneous Objects (OE-HO):** The inputs form a matrix of  $(S \times n)$ , where  $S$  is the batch size, the values are sampled from a uniform random distribution  $U[0, 1]$ . The batch size is set to be as large as possible, for  $n < 10$ ,  $S = 10000$  and  $n = 10$ ,  $S = 50000$ . After that we apply the ordering and calculate payments for the given input valuations as defined in Section 4.3.1. Next, we feed it to the linear network model (Figure 1) whose parameters are initialized using Xavier initialization (Section 5.1.1). The objective function given by Equation (2) is applied to the output of the network and parameters are updated using Adam optimizer (Section 5.1.2), learning rate set to 0.0001. The nonlinear model (Figure 2) is also trained in the similar manner. We used 1000 nodes in the hidden layer and the network was trained with a learning rate of  $10e - 4$ .

**5.2.2 Optimal in Worst-case for Homogeneous Objects (OW-HO):** As in the OE case, linear network model is used and similar procedure is followed. The input is sorted as given in Section 4.3.1 and along with the calculated payments is fed to the linear network. The objective given in Equation (3) is optimized with the learning rate set to 0.0001 and the training is carried till the loss decreases and saturates which happens when the redistribution index is optimal. As discussed in Section 3.3.2 for homogeneous setting linear rebate functions are optimal among all possible deterministic functions

**Table 3:  $e^{ow}$  for Homogeneous and Heterogeneous setting.**

$n, p$	Homogeneous		Heterogeneous	
	OW-HO theoretical	OW-HO Linear NN	OW-HE <sup>1</sup> Linear NN	OW-HE Nonlinear NN
3,1	0.333	0.336	0.332	0.333
4,1	0.571	0.575	0.571	0.571
4,2	0.250	0.250	0.0	0.250
5,1	0.733	0.739	0.733	0.732
5,2	0.454	0.460	0.0	0.454
5,3	0.200	0.200	0.0	0.199
6,1	0.839	0.847	0.839	0.838
6,2	0.615	0.620	0.0	0.614
6,3	0.375	0.378	0.0	0.375
7,1	0.905	0.910	0.905	0.904
7,2	0.737	0.746	0.0	0.736
7,3	0.524	0.538	0.0	0.523
8,1	0.945	0.949	0.945	0.943
8,2	0.825	0.834	0.0	0.825
9,1	0.969	0.972	0.968	0.968
9,2	0.887	0.894	0.0	0.886
10,1	0.982	0.985	0.982	0.982
10,2	0.928	0.936	0.0	0.927

which are DSIC and AE, hence we did not use nonlinear model for this case.

**5.2.3 Optimal in Expectation for Heterogeneous Objects (OE-HE):** The inputs are again randomly sampled from a uniform distribution  $U[0, 1]$ , the input matrix is of the form,  $(S \times n \times p)$ . Then the inputs are ordered as defined in Section 4.3.2. Both the networks Figure 1, Figure 2 are used for finding the optimal in expectation mechanism. Just like in the homogeneous case, network parameters are initialized using Xavier initialization. For the payment calculation, we use the scipy library for linear sum assignment. This library assigns objects such that the cost is minimized, whereas we want the valuation to be maximized as per AE, hence we negate the bids before passing it to the function. Besides, the Hungarian algorithm for assignment works only when the number of objects to be assigned is same as the agents, hence we introduce dummy agents or dummy objects with zero valuation so that the input matrix is a square matrix. The objective function 2 is optimized using Adam optimizer (described in Section 5.1.2) with learning rate  $10e - 4$  for both the linear and nonlinear models. In the nonlinear network, the number of nodes in the hidden layer was set to 1000.

**5.2.4 Optimal in Worst-case for Heterogeneous Objects (OW-HE):** For designing this particular mechanism, we use the same inputs that we used in the OE setting for heterogeneous items. The networks used and their initialization is also same. The only difference is the objective function which is given by Equation (3) and the optimizer used is Adam. For the linear network learning rate is  $10e - 4$ . In the nonlinear network, the number of hidden nodes used were 100 and learning rate of  $10e - 4$  for all values of  $n$ , with  $p = 1$ . When the values of  $p > 1$ , the number of hidden nodes was increased to 1000 and a learning rate of  $10e - 5$  was used.

<sup>1</sup>All values below  $10e-3$  are considered to be 0.0

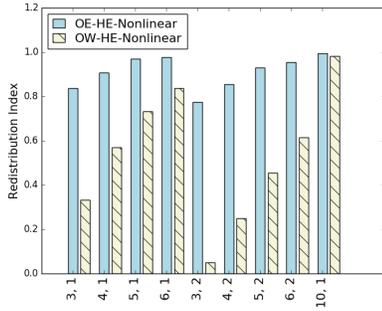


Figure 3: OE-HE-Nonlinear Vs OW-HE-Nonlinear

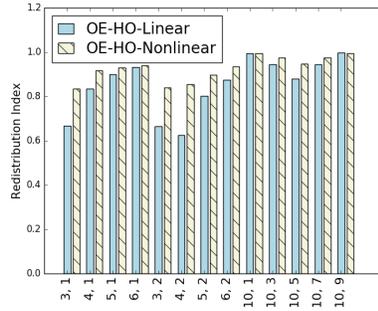


Figure 4: OE-HO-Linear vs OE-HO-Nonlinear

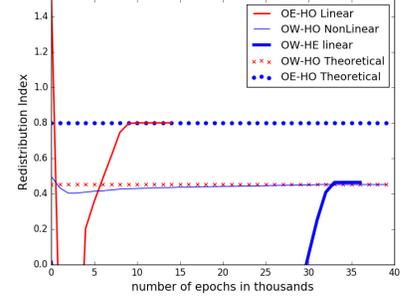


Figure 5: RI values with change in epoch for  $n = 5, p = 2$

### 5.3 Specific parameters used for training

- We used tensorflow library for all the implementations. We used GPUs: Tesla K40c and GeForce GTX Titan X. The network training time varies from few minutes to a whole day, for the different  $n, p$  values.
- The number of input samples,  $T$  for binary settings ideally should be  $2^{np}$  and way more than this for the real value cases. So we use 10000 samples where  $np \leq 13$ , 70000 for  $np \leq 16$  and 100000 for the rest of the cases.
- All the experiments are run for a maximum of 400000 epochs and the constant  $\rho$  as given in the overall loss functions (2, 3) is set to 1000.
- The number of nodes in the hidden layer for the network given in Figure 2 is 1000 ideally. Even having 100 is sufficient for the cases where the product of  $n$  and  $p$  is less than 10.

### 5.4 Results and Discussion

We now describe the results obtained from the networks trained as discussed above. Table 2 compares the values of redistribution index ( $e^{oe}$ ) for all the experiments with OE objective for different  $(n,p)$  values under homogeneous and heterogeneous settings. The first column indicates theoretical bounds on OE RMs with linear rebate functions. In the column heterogeneous, we compare the redistribution indexes obtained by our OE-HE networks with linear and nonlinear networks. Similarly, Table 3 compares the OW redistribution index ( $e^{ow}$ ) for all the networks under consideration and theoretical values for different  $(n,p)$  values. With these tables, the following are our observations:

*Achieving the analytically solved bounds:* For the OE objective in homogeneous setting, our network OE-HO-Linear achieves the theoretical values of  $e^{oe}$  proposed in [18]. Similarly, the theoretical  $e^{ow}$  values are achieved by the network OW-HO Linear.

*OW nonlinear rebate function for heterogeneous setting :* The values from OE-HE-Linear NN illustrate the impossibility theorem stated in [12] that there cannot be a linear rebate function with non-zero redistribution index for heterogeneous settings. For  $p = 1$ , there being no difference in homogeneous or heterogeneous, the values remain the same, but are zero for  $p > 1$ . The network OW-HE-Nonlinear achieves the theoretical values given in OW-HO theoretical Table 3.

*OE nonlinear rebate function for homogeneous setting :* [18] have only tried to find the OE linear rebate function. OE-HO Nonlinear NN outperforms the linear counterpart. The graph in Figure 3 illustrates the comparison. This indicates the existence of a nonlinear rebate function which guarantees higher  $e^{oe}$  than the linear rebate function for homogeneous setting.

*OE nonlinear rebate function for heterogeneous setting :* The values from OE-HE-Linear NN is shows same results as OE-HO Linear for  $p = 1$  and different otherwise. OE-HE-Nonlinear NN outperforms the linear network The graph in Figure 4 compares between the OW and OE performance of the nonlinear network for heterogeneous settings.

Figure 3 illustrates the significantly better performance of Optimal in Expectation RM wrt Optimal in Worst Case for heterogeneous settings for different  $(n, p)$  values. For reference, it also has OE performance for homogeneous settings. We also observe that performance of OE mechanisms with nonlinear rebates is significantly better as compared to OE mechanism with linear rebates (Figure 4). The convergence of neural networks for  $n = 5, p = 2$  with number of epochs while training can be found in Figure 5. Typically, most of the networks studied here converge in less than 80000 epochs for  $n \leq 10$ . (Whenever the objective value of a neural network drops below zero, we skip them to plot).

## 6 CONCLUSIONS

In this paper, we considered a problem of designing an optimal redistribution mechanism. We proposed a novel approach to this problem: train a neural network with randomly generated valuation profiles for a desired goal. Our neural network design takes care of problem specific constraints. We showed that, it can learn optimal redistribution mechanisms with proper initialization and a suitably defined ordering over valuation profiles. Our analysis shows that one can design nonlinear rebate functions for homogeneous settings that perform better than optimal in expectation linear rebate functions. We could design optimal in expectation rebate functions for heterogeneous objects which is not solved analytically.

There are many challenges to be handled here. For example, can we design a vanilla neural network that can learn linear, nonlinear rebate functions without explicitly designing such architectures, based on the problem specific needs? Can we come up with training strategies independent of  $(n, p)$ ?

## REFERENCES

- [1] Martin J Bailey. 1997. The Demand Revealing Process: To Distribute the Surplus. *Public Choice* 91, 2 (1997), 107–26. <https://EconPapers.repec.org/RePEc:kap:pubcho:v:91:y:1997:i:2:p:107-26>
- [2] Ruggiero Cavallo. 2006. Optimal Decision-Making With Minimal Waste: Strategyproof Redistribution of VCG Payments. In *Proc. of the 5th Int. Joint Conf. on Autonomous Agents and Multi Agent Systems (AAMAS'06)*. Hakodate, Japan, 882–889. <http://econcs.seas.harvard.edu/files/econcs/files/cavallo-redis.pdf>
- [3] Edward Clarke. 1971. Multipart pricing of public goods. *Public Choice* 11, 1 (1971), 17–33. <https://EconPapers.repec.org/RePEc:kap:pubcho:v:11:y:1971:i:1:p:17-33>
- [4] Geoffroy de Clippel, Victor Naroditskiy, Maria Polukarov, Amy Greenwald, and Nicholas R. Jennings. 2014. Destroy to save. *Games and Economic Behavior* 86, C (2014), 392–404.
- [5] Paul Dütting, Zhe Feng, Harikrishna Narasimhan, and David C. Parkes. 2017. Optimal Auctions through Deep Learning. *CoRR* abs/1706.03459 (2017). <http://arxiv.org/abs/1706.03459>
- [6] Boi Faltings. 2005. A Budget-balanced, Incentive-compatible Scheme for Social Choice. In *Proceedings of the 6th AAMAS International Conference on Agent-Mediated Electronic Commerce: Theories for and Engineering of Distributed Mechanisms and Systems (AAMAS'04)*. Springer-Verlag, Berlin, Heidelberg, 30–43. DOI: [http://dx.doi.org/10.1007/11575726\\_3](http://dx.doi.org/10.1007/11575726_3)
- [7] Dinesh Garg, Y Narahari, and Sujit Gujar. 2008. Foundations of Mechanism Design: A Tutorial - Part 1: Key Concepts and Classical Results. *Sadhana - Indian Academy Proceedings in Engineering Sciences* 33, Part 2 (April 2008), 83–130.
- [8] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*, Yee Whye Teh and Mike Titterton (Eds.), Vol. 9. PMLR, Chia Laguna Resort, Sardinia, Italy, 249–256. <http://proceedings.mlr.press/v9/glorot10a.html>
- [9] Alex Graves. 2013. Generating Sequences With Recurrent Neural Networks. *CoRR* abs/1308.0850 (2013).
- [10] Jerry R. Green and Jean-Jacques Laffont. 1979. *Incentives in Public Decision Making*. North-Holland, Amsterdam.
- [11] Theodore Groves. 1973. Incentives in Teams. *Econometrica* 41, 4 (1973), 617–31. <https://EconPapers.repec.org/RePEc:emetrp:v:41:y:1973:i:4:p:617-31>
- [12] Sujit Gujar and Y. Narahari. 2011. Redistribution Mechanisms for Assignment of Heterogeneous Objects. *J. Artif. Int. Res.* 41, 2 (May 2011), 131–154. <http://dl.acm.org/citation.cfm?id=2051237.2051242>
- [13] Faruk Gul and Ennio Stacchetti. 1999. Walrasian Equilibrium with Gross Substitutes. *Journal of Economic Theory* 87, 1 (1999), 95–124. <https://EconPapers.repec.org/RePEc:eee:jetheo:v:87:y:1999:i:1:p:95-124>
- [14] Mingyu Guo. 2011. VCG Redistribution with Gross Substitutes. (2011).
- [15] Mingyu Guo. 2012. Worst-case Optimal Redistribution of VCG Payments in Heterogeneous-item Auctions with Unit Demand. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2 (AAMAS '12)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 745–752. <http://dl.acm.org/citation.cfm?id=2343776.2343803>
- [16] Mingyu Guo and Vincent Conitzer. 2007. Worst-case Optimal Redistribution of VCG Payments. In *Proceedings of the 8th ACM Conference on Electronic Commerce (EC '07)*. ACM, New York, NY, USA, 30–39. DOI: <http://dx.doi.org/10.1145/1250910.1250915>
- [17] Mingyu Guo and Vincent Conitzer. 2008. Better Redistribution with Inefficient Allocation in Multi-unit Auctions with Unit Demand. In *Proceedings of the 9th ACM Conference on Electronic Commerce (EC '08)*. ACM, New York, NY, USA, 210–219. DOI: <http://dx.doi.org/10.1145/1386790.1386825>
- [18] Mingyu Guo and Vincent Conitzer. 2008. Optimal-in-expectation Redistribution Mechanisms. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2 (AAMAS '08)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1047–1054. <http://dl.acm.org/citation.cfm?id=1402298.1402367>
- [19] Mingyu Guo and Vincent Conitzer. 2009. Worst-case optimal redistribution of VCG payments in multi-unit auctions. *Games and Economic Behavior* 67, 1 (2009), 69–98.
- [20] Jason D. Hartline and Tim Roughgarden. 2008. Optimal Mechanism Design and Money Burning. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing (STOC '08)*. ACM, New York, NY, USA, 75–84. DOI: <http://dx.doi.org/10.1145/1374376.1374390>
- [21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity Mappings in Deep Residual Networks. *CoRR* abs/1603.05027 (2016). <http://arxiv.org/abs/1603.05027>
- [22] Kurt Hornik. 1991. Approximation Capabilities of Multilayer Feedforward Networks. *Neural Networks* 4 (1991), 251–257. DOI: [http://dx.doi.org/10.1016/0893-6080\(91\)90009-T](http://dx.doi.org/10.1016/0893-6080(91)90009-T)
- [23] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2, 5 (1989), 359–366.
- [24] Kenji Kawaguchi. 2016. Deep Learning without Poor Local Minima. In *Advances in Neural Information Processing Systems* 29, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 586–594. <http://papers.nips.cc/paper/6112-deep-learning-without-poor-local-minima.pdf>
- [25] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). <http://arxiv.org/abs/1412.6980>
- [26] Eric Maskin, J. J. Laffont, and J. J. Laffont. 1979. *A Differential Approach to Expected Utility Maximizing Mechanisms*. North Holland, 289–308.
- [27] Herve Moulin. 2009. Almost budget-balanced VCG mechanisms to assign multiple objects. *Journal of Economic Theory* 144, 1 (2009), 96–119.
- [28] Noam Nisan. 2007. Introduction to Mechanism Design (for Computer Scientist). In *Algorithmic game theory*, Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay Vazirani (Eds.). Cambridge University Press, 209–242.
- [29] David C. Parkes, Jayant R. Kalagnanam, and Marta Eso. 2001. Achieving Budget-Balance with Vickrey-Based Payment Schemes in Exchanges. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*. 1161–1168. <http://econcs.seas.harvard.edu/files/econcs/files/combexch01.pdf>
- [30] Daniel Soudry and Yair Carmon. 2016. No bad local minima: Data independent training error guarantees for multilayer neural networks. *CoRR* abs/1605.08361 (2016). <http://arxiv.org/abs/1605.08361>
- [31] Pingzhong Tang. 2017. Reinforcement mechanism design. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 5146–5150. DOI: <http://dx.doi.org/10.24963/ijcai.2017/739>
- [32] W. Vickrey. 1961. Counterspeculation, Auctions, and Competitive Sealed Tenders. *Journal of Finance* 16, 1 (March 1961), 8–37.