Optimal-er Auctions through Attention Appendix

Dmitry Ivanov[*]
HSE University & Technion
Israel

Iskander Safiulin
Independent researcher
Russia

Igor Filippov
Independent researcher
Russia

Ksenia Balabaeva
ITMO University & BIOCAD
Russia

1 Layers and architecture of RegretFormer

Multi-head attention Popularized by Vaswani et al. [10], the attention function maps a query vector and a set of key-value vector pairs to an output vector. The procedure is typically applied to a set or a sequence of queries. The output vector is a weighted sum of the values, and each weight reflects the compatibility of the query with the corresponding key. While different attention mechanisms exist, softmax attention is the most common:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

where Q, K, and V are respectively the matrices of queries, keys, and values, and d_k is the number of keys. Self-attention is a special case of attention in which Q, K, and V are linear projections of the same inputs. Typically, layer normalization is applied to the input before projecting [1]. Vaswani et al. [10] also propose multi-head attention (MHA). In this extension, *H* different attention heads are created, and for each attention head the input matrices are projected with head-specific weight matrices QW_h^Q, KW_h^K, VW_h^V to calculate the inputs to attention:

$$\text{MHA}(Q, K, V) = \text{Concat}(head_1, ..., head_H)W^O \tag{2}$$

$$head_h = \text{Attention}(QW_h^Q, KW_h^K, VW_h^V) \tag{3}$$

Attention is equivariant to the order of elements in the input, which is a useful property when learning symmetric auctions since any optimal symmetric auction can be represented by a permutation-equivariant function [2]. In applications where this order is important (e.g. order of words in a sentence), Positional Encoding (PE) is usually applied. This technique augments the initial representation of the input data with information about the order of the elements. We demonstrate how PE can be applied to learn optimal asymmetric auctions in the main text.

Exchangeable Layers The exchangeable layer [3] is inspired by deep sets [12] and is defined as follows. A layer is specified by the number of input channels K, the number of output channels O and five learnable parameters $w_1, w_2, w_3, w_4 \in \mathbb{R}^{KxO}$ and $w_5 \in \mathbb{R}^O$. The input is a tensor *B* of size *(K, n, m)* and the output is tensor *Y* of size *(O, n, m)*. The element (i, j) of the *o*-th output channel $Y_{i,j}^{(o)}$ is given by:

[*]dimonenka@mail.ru, divanov@campus.technion.ac.il

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

$$Y_{i,j}^{(o)} = \sigma\left(\sum_{k=1}^{K}(w_1^{(k,o)} B_{i,j}^{(k)} + \frac{w_2^{(k,o)}}{n}\sum_{i'} B_{i',j}^{(k)} + \frac{w_3^{(k,o)}}{m}\sum_{j'} B_{i,j'}^{(k)} + \frac{w_4^{(k,o)}}{nm}\sum_{i',j'} B_{i',j'}^{(k)}) + w_5^{(o)}\right) \tag{4}$$

This layer constitutes the main building block of EquivariantNet [8] and is also used as the first layer of RegretFormer.

Architecture of RegretFormer There are several high-level differences between the architectures of RegretFormer and RegretNet. Whereas RegretNet uses two separate networks to calculate allocations and payments, our architecture has a single shared network with both outputs. Unlike RegretNet where the input is flattened into a vector, the input of RegretFormer is the two-dimensional matrix of bids B^{nm}. Furthermore, unlike RegretNet, n and m are not fixed in RegretFormer.

We now describe the architecture in detail. Note that each described layer except the output layers is followed by a Tanh activation.

First, we apply an exchangeable layer (4) to transform each bid into an initial vector of features that already contains information about other bids. According to our definition of the exchangeable layer, this requires adding a third dummy dimension to the bid matrix $B^{nm} \to B^{nm1}$:

$$L_1^{nmk} = \text{ExchangeableLayer}(B^{nm1}) \tag{5}$$

Then, we sequentially apply several attention-based blocks. Each block consists of two multi-head self-attention layers with residual connections, one applied item-wise and one applied participant-wise. For each layer, we accordingly reshape the input. After applying the attention layers, we concatenate their predictions and apply the same fully-connected layer (FC) to the feature vectors of each bid (to reduce the dimensionality of the feature vectors to the initial size):

$$L_{t+1,item}^{nmk} = \text{MHA}_{item}(L_t, L_t, L_t) + L_t \tag{6}$$

$$L_{t+1,part}^{nmk} = \text{MHA}_{part}(L_t, L_t, L_t) + L_t \tag{7}$$

$$L_{t+1}^{nmk} = \text{FC}_{t+1}(\text{Concat}(L_{t+1,item}, L_{t+1,part})) + L_t \tag{8}$$

After applying the attention-based block N times (in our experiments, we set N to 1 or 2), we obtain the attended feature matrix L_{N+1}^{nmk}. From this matrix, we create two separate matrices by averaging over one of the dimensions: the participant feature matrix $P_{N+1}^{nk} = \frac{1}{m}\sum_j (L_{N+1}^{njk})$ and the item feature matrix $I_{N+1}^{mk} = \frac{1}{n}\sum_i (L_{N+1}^{imk})$. These matrices are essentially embeddings of participants and items respectively and are used to compute the allocation matrix and the payment vector.

To compute the allocation matrix, we multiply the item and the participant matrices, which gives us an n by m matrix of unscaled probabilities (logits). Before scaling, we need to additionally consider the possibility of each item remaining unallocated. To this end, we introduce a dummy participant $n+1$, the unscaled probability for which is estimated for each item as a negated sum of the unscaled probabilities over the real participants. Finally, we apply the softmax function along the participants to scale the probabilities. When summarized, the allocation matrix is obtained in the following way:

$$L_{N+2}^{nm} = \text{MatMul}(P_{N+1}^{nk}, (I_{N+1}^{mk})^T) \tag{9}$$

$$L_{norm}^{(n+1)m} = \text{Concat}(L_{N+2}^{nm}, -\sum_i (L_{N+2}^{im})) \tag{10}$$

$$Z_{out}^{(n+1)m} = \text{SoftMax}(L_{norm}^{(n+1)m}) \tag{11}$$

To estimate the payment vector, we average the participant feature matrix over the feature dimension and apply the sigmoid activation to scale the output between 0 and 1:

Table 1: Neural architecture hyperparameters

Hyperparameter	1x2	2x2	2x3	2x5	3x10	multi
RegretNet						
fully-connected layers	3	3	3	6	6	6
hidden dim	200	200	200	200	200	200
EquivariantNet						
exchangeable layers	3	3	5	6	6	6
hidden dim	32	32	32	32	32	32
RegretFormer						
exchangeable layers	1	1	1	1	1	1
attention layers	1	1	1	2	2	2
attention heads	2	2	2	4	4	4
hidden dim	32	64	64	128	128	128

$$\hat{P}^n_{out} = \text{Sigmoid}(\frac{1}{k}\sum_z P^{nz}_{N+1}) \tag{12}$$

Like in RegretNet, we calculate the final payments as $p_i = \hat{p_i} \sum_{j=1}^{m} z_{ij} b_{ij}$ for $i = 1, \ldots, n$.

2 Technical details and hyperparameters

In all experiments, all networks are trained for 200000 iterations of outer optimization, each iteration corresponding to one step of the optimizer on one mini-batch. The training dataset consists of 640000 profiles (same as in the RegretNet paper) divided into 1250 mini-batches of 512 profiles. The validation dataset consists of 4096 profiles divided into 128 batches of 32 profiles. The number of inner optimization steps per one outer update equals 50 during training and 1000 during validation. The learning rate equals 0.001 for the outer optimization and 0.1 for the inner optimization. Both use separate Adam optimizers [6]. The hyperparameters related to the neural architectures are reported in Table 1. We report the sizes of neural networks in Table 7. All experiments are repeated three times and the average metrics are reported. Experiments are run on an internal cluster with V100 GPUs.

The hyperparameters related to our budget-based approach are the following. In all our experiments, we initialize $\gamma = 1$, set $\gamma_\Delta = 0.5$, set $R^{start}_{max} = 0.01$, and set such R^{mult}_{max} that R_{max} converges to R^{end}_{max} in two-thirds of the training time. We set $R^{end}_{max} = 0.001$ by default but additionally investigate the effect of choosing a lower budget $R^{end}_{max} = 0.0001$.

3 Additional results

3.1 Learning curves

We present the learning curves of the revenue, the regret, and the penalty coefficient γ for settings {1x2, 2x2, 2x3, 2x5, 3x10} in Figures 1, 2, 3, 4, and 5, respectively. The results of the same experiments are reported in Table 1 in the main text.

Note that the shapes of learning curves are a consequence of our regret budget schedule. Specifically, we provide a higher regret budget at the beginning of the training so the network finds a solution with high revenue (that does not satisfy the desirable budget), and then we tighten the regret budget (which also causes the revenue to decrease). It can also be seen that both revenue and regret start flattening at the same time as the regret penalty coefficient γ stops increasing. This happens approximately at 2/3 of the training time, in accordance with our choice of R^{mult}_{max}.

Table 2: Network distillation using networks from Table 1 in the main text, $R_{max} = 10^{-3}$

setting	metric	misreports of	RegretFormer → RegretNet		RegretFormer → EquivariantNet	
			teacher	student	teacher	student
1x2	revenue	-	0.577	0.578	0.578	0.578
	regret	teacher	0.00087	0.00098	0.00090	0.00120
		student	0.00051	0.00090	0.00024	0.00066
2x2	revenue	-	0.912	0.912	0.913	0.917
	regret	teacher	0.00057	0.00074	0.00057	0.00154
		student	0.00068	0.00148	0.00054	0.00404
2x3	revenue	-	1.412	1.412	1.414	1.419
	regret	teacher	0.00097	0.00153	0.00093	0.00183
		student	0.00129	0.00415	0.00090	0.00400
2x5	revenue	-	2.439	2.436	2.440	2.449
	regret	teacher	0.00112	0.00125	0.00110	0.00178
		student	0.00106	0.00416	0.00064	0.00577
3x10	revenue	-	6.153	6.169	6.155	6.163
	regret	teacher	0.00238	0.00386	0.00237	0.00298
		student	0.00258	0.02713	0.00220	0.01626

3.2 Network distillation

Here we elaborate on our validation procedure based on network distillation that was mentioned in the main text.

The distillation procedure is based on training the 'student' network w_s to approximate the predictions of a trained 'teacher' network w_t. We apply this procedure in five settings {1x2, 2x2, 2x3, 2x5, 3x10} given $R_{max} = 10^{-3}$ to distill RegretFormer onto RegretNet, as well as to distill RegretFormer onto EquivariantNet. If the architecture of RegretFormer for some reason impairs its ability to approximate optimal misreports, a RegretNet or an EquivariantNet trained to closely mimic the predictions of a RegretFormer may find better misreports that produce higher regret values for the RegretFormer. Specifically, since the predictions of allocation and payment modules can respectively be treated as the categorical and the Bernoulli distributions, we train the student network to minimize the KL divergence from its predictions to the predictions of the teacher network. For example, to train the allocation module, the student minimizes $KL(g(w_t), g(w_s)) = \frac{1}{|B|} \sum_{l \in B} \sum_{i,j} z_{ij}(v^l; w_t) \cdot (\log(z_{ij}(v^l; w_t)) - \log(z_{ij}(v^l; w_s)))$, and likewise for the payment module. This approach was initially proposed in Hinton et al. [4]. Furthermore, to satisfy DSIC on the whole support, the KL-divergence is minimized at both the true valuations v^l and the approximate optimal misreport for each participant $v_i'^l(w_s)$, estimated by the student network as per usual.

We report the results in Table 2. Like in the cross-misreport validation procedure, we do not find any evidence that RegretFormer approximates regret worse than the alternative architectures. This can be seen in Table 2 by comparing the teacher regret estimated on teacher misreports with the teacher regret estimated on student misreports: the latter is never substantially higher than the former.

As additional evidence of the performance gap being genuine, in the distillation experiments, both students achieve the same revenue as the teacher while consistently producing higher regret, up to a magnitude on the hardest 3x10 setting. This can have two explanations. First, the student networks get stuck in one of the multiple local optimums, which prevents them from reaching lower regrets. Moreover, this consistently happens both when learning from scratch (since in Table 1 in the main text both RegretNet and EquivariantNet achieve lower revenue than RegretFormer for the same regret budgets) and when mimicking predictions of RegretFormer. Second, the better solutions with high revenue and low regret that can be found by RegretFormer are simply absent from the space of the mechanisms that can be represented by RegretNet and EquivariantNet.

Table 3: Out-of-setting generalization

Training setting	Validation setting	EquivariantNet revenue	EquivariantNet regret	RegretFormer revenue	RegretFormer regret
1x2	2x2	0.690	0.04403	0.669	0.03776
	2x3	1.084	0.07074	1.056	0.07167
	2x5	1.917	0.12465	1.863	0.13151
	3x10	4.308	0.29291	4.197	0.29106
2x2	1x2	0.695	0.13343	0.768	0.21671
	2x3	1.350	0.00071	1.412	0.01221
	2x5	2.307	0.03169	2.402	0.03250
	3x10	5.156	0.55869	4.943	0.32002
2x3	1x2	0.686	0.14900	0.775	0.20051
	2x2	0.875	0.00116	0.904	0.00137
	2x5	2.318	0.00615	2.432	0.00938
	3x10	5.271	0.37967	4.929	0.27183
2x5	1x2	0.743	0.19830	0.816	0.23945
	2x2	0.900	0.00066	0.903	0.00072
	2x3	1.401	0.00103	1.415	0.00103
	3x10	5.517	0.24757	4.884	0.28009
3x10	1x2	0.552	0.53700	0.801	0.18780
	2x2	0.693	0.19767	1.053	0.05893
	2x3	1.099	0.33665	1.611	0.07595
	2x5	1.936	0.59959	2.754	0.10181

Table 4: Out-of-multi-setting generalization

Setting	RegretNet revenue	RegretNet regret	RegretFormer revenue	RegretFormer regret
average	2.733	0.0662	2.734	0.00478
2x4	2.115	0.083	1.943	0.00368
2x6	3.166	0.078	3.012	0.00592
3x3	1.743	0.010	1.718	0.00252
3x5	2.918	0.016	2.910	0.00313
3x7	3.722	0.144	4.086	0.00866

3.3 Out-of-setting generalization

In these experiments, we investigate how well the architectures generalize to unseen settings. It's clear that RegretNet cannot be applied out-of-domain since its layers rely on the constant input size, so we compare our network with EquivariantNet. The networks are trained in five settings {1x2, 2x2, 2x3, 2x5, 3x10} and then tested in all settings but the setting used for training.

The results are reported in Table 3. Both networks look promising when the number of objects varies and the number of bidders remains constant. However, generalization to the settings where the number of bidders varies is poor for both networks due to complex interactions between the participants. Similar results were observed by Rahme et al. [8]. Still, when the validation setting has the number of participants same as or less than the training setting, RegretFormer usually outperforms EquivariantNet by either achieving higher revenue or lower regret.

3.4 Cross-misreport validation in multi-settings

In Section 4.5 in the main text, we have mentioned performing cross-misreport validation in multi-settings. Table 5 presents the full results of this experiment. It is evident that both RegretNet and RegretFormer approximate the optimal misreports adequately since each estimates the highest

Table 5: Cross-misreport regret estimates in the multi-setting. The highest regret for a network is highlighted in bold. Notice that EquivariantNet poorly estimates misreports as its regret on misreports of RegretFormer is consistently higher than on its own misreports.

setting	regret of	misreports of		
		RegretNet	EquivariantNet	RegretFormer
2x3	RegretNet	**0.00305**	0.00083	0.00134
	EquivariantNet	0.00514	0.00258	**0.00554**
	RegretFormer	0.00128	0.00105	**0.00246**
2x4	RegretNet	**0.00341**	0.00060	0.00128
	EquivariantNet	0.00609	0.00273	**0.00925**
	RegretFormer	0.00154	0.00106	**0.00317**
2x5	RegretNet	**0.00362**	0.00050	0.00115
	EquivariantNet	0.00726	0.00309	**0.01270**
	RegretFormer	0.00188	0.00113	**0.00391**
2x6	RegretNet	**0.00425**	0.00065	0.00131
	EquivariantNet	0.00873	0.00337	**0.01597**
	RegretFormer	0.00248	0.00118	**0.00439**
2x7	RegretNet	**0.00457**	0.00048	0.00111
	EquivariantNet	0.00961	0.00356	**0.01951**
	RegretFormer	0.00242	0.00122	**0.00481**
3x3	RegretNet	**0.00322**	0.00070	0.00101
	EquivariantNet	0.00254	0.00189	**0.00358**
	RegretFormer	0.00110	0.00083	**0.00251**
3x4	RegretNet	**0.00264**	0.00042	0.00075
	EquivariantNet	0.00354	0.00214	**0.00508**
	RegretFormer	0.00138	0.00087	**0.00336**
3x5	RegretNet	**0.00277**	0.00032	0.00058
	EquivariantNet	0.00426	0.00265	**0.00709**
	RegretFormer	0.00167	0.00096	**0.00421**
3x6	RegretNet	**0.00340**	0.00030	0.00058
	EquivariantNet	0.00501	0.00277	**0.00916**
	RegretFormer	0.00171	0.00181	**0.00476**
3x7	RegretNet	**0.00430**	0.00027	0.00054
	EquivariantNet	0.00610	0.00326	**0.01101**
	RegretFormer	0.00198	0.00101	**0.00553**

regret on their own misreports. In contrast, EquivariantNet poorly approximates misreports and underestimates regret in multi-settings since both RegretNet and RegretFormer find better misreports for this network.

3.5 Out-of-multi-setting generalization

We define two subsets of settings to respectively train and validate networks:

1. *Train*: S_{train}={2x3, 2x5, 2x7, 3x4, 3x6}.
2. *Test*: S_{test}={2x4, 2x6, 3x3, 3x5, 3x7}.

We compare how networks generalize to unseen settings when trained in the multi-setting regime. To this end, we train RegretNet and RegretFormer on S_{train} and then validate them on S_{test}. Because EquivariantNet poorly underestimates regret in multi-settings (see Section 4.5 in the main text and 3.4), we do not include it for comparison.

Table 6: Average wall-clock training time, hours

setting	RegretNet	EquivariantNet	RegretFormer
1x2	2.5	5.2	12.5
2x2	2.5	5.3	12.0
2x3	2.5	8.4	11.0
2x5	3.1	9.8	28.7
3x10	4.7	11.2	82.1

Table 7: Number of parameters. In preliminary experiments, we found that RegretNet and EquivariantNet do not benefit from increasing the network sizes past what was used in the respective papers, whereas RegretFormer requires more parameters to perform optimally.

setting	RegretNet	EquivariantNet	RegretFormer
1x2	21305	4546	12705
2x2	22008	4546	49985
2x3	22711	12802	49985
2x5	84717	16930	362753
3x10	91343	16930	362753

The resulting revenue and regret values are reported in Table 4. In all out-of-domain settings, RegretNet produced poor results. On average, its regret is more than an order of magnitude larger compared to RegretFormer. In contrast, our approach stably generalizes to all unseen settings while keeping regret low. Remarkably, its revenue is as high and its regret is as low as in the experiments where all constant-sized settings are available during training (Table 5 in the main text). The superiority of RegretFormer over RegretNet is especially prominent in the 3x7 setting where our network achieves a larger revenue while producing 16 times as little regret.

3.6 Training time

We report the average wall-clock training time in hours in Table 6. It takes longer to train RegretFormer than the baselines for two reasons. First, to perform optimally RegretFormer requires a bigger network with more parameters than baselines, especially in the bigger settings. Please see Table 7 for summarized sizes of the three neural architectures in all settings. Second, the attention layers have quadratic $O(n^2)$ complexity (where n is the number of items or participants). If training time is an issue, the attention layers can be replaced with one of the multiple modifications that have $O(n \log n)$ or $O(n)$ complexity [9, 7, 11, 5].

References

[1] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] C. Daskalakis and S. M. Weinberg. Symmetries and optimal multi-dimensional mechanism design. In *Proceedings of the 13th ACM conference on Electronic commerce*, pages 370–387, 2012.

[3] J. Hartford, D. Graham, K. Leyton-Brown, and S. Ravanbakhsh. Deep models of interactions across sets. In *International Conference on Machine Learning*, pages 1909–1918. PMLR, 2018.

[4] G. Hinton, O. Vinyals, J. Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[5] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

[6] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[7] N. Kitaev, Ł. Kaiser, and A. Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

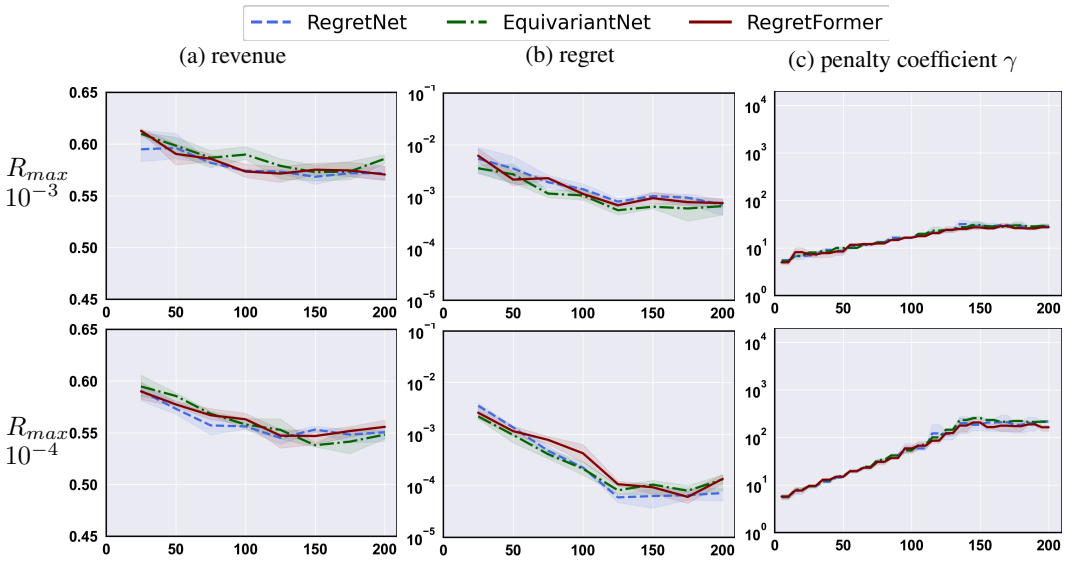

Figure 1: Learning curves in the setting 1x2. The X-axis is in the thousands of training iterations. The shaded regions correspond to the min-max spread over three random seeds.

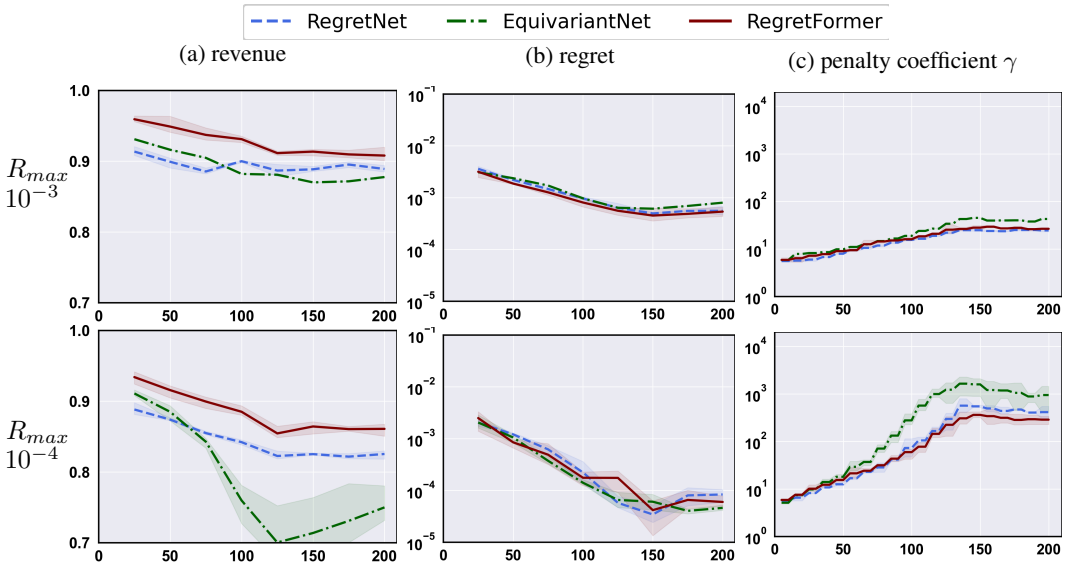

Figure 2: Learning curves in the setting 2x2. The X-axis is in the thousands of training iterations. The shaded regions correspond to the min-max spread over three random seeds.

[8] J. Rahme, S. Jelassi, J. Bruna, and S. M. Weinberg. A permutation-equivariant neural network architecture for auction design. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(6):5664–5672, May 2021. URL `https://ojs.aaai.org/index.php/AAAI/article/view/16711`.

[9] Z. Shen, M. Zhang, H. Zhao, S. Yi, and H. Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3531–3539, 2021.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[11] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

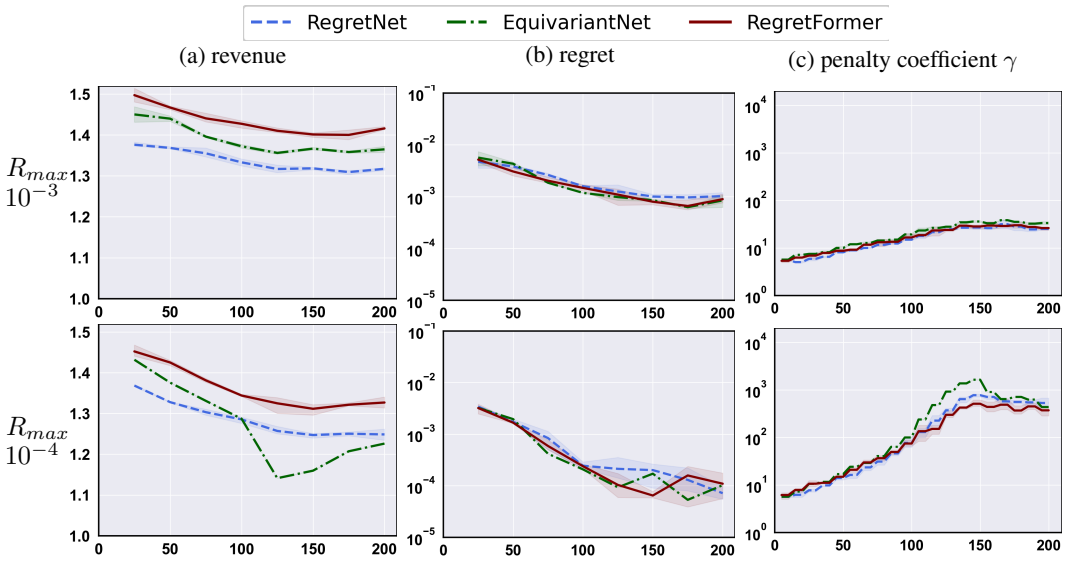

Figure 3: Learning curves in the setting 2x3. The X-axis is in the thousands of training iterations. The shaded regions correspond to the min-max spread over three random seeds.

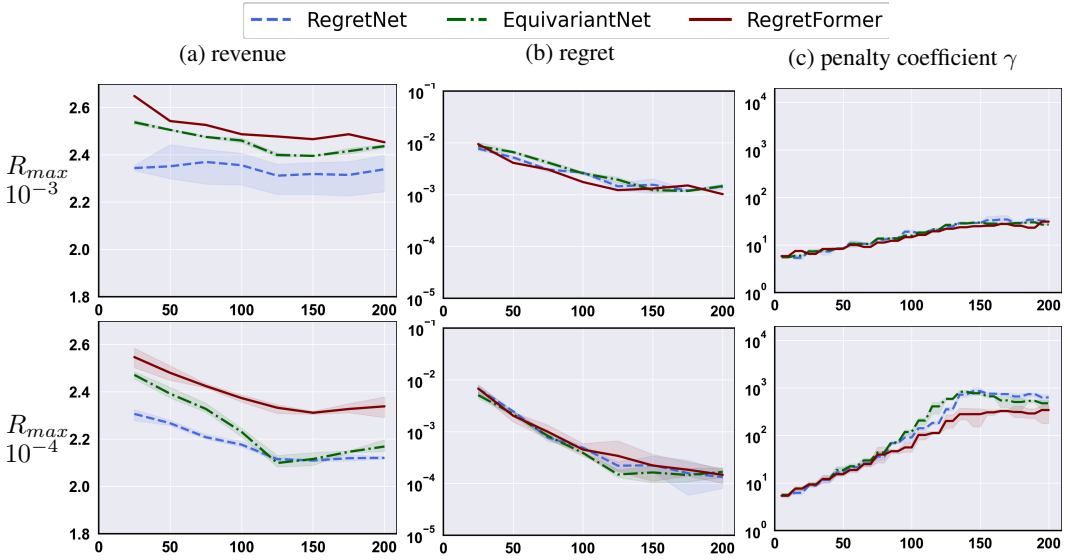

Figure 4: Learning curves in the setting 2x5. The X-axis is in the thousands of training iterations. The shaded regions correspond to the min-max spread over three random seeds.

[12] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

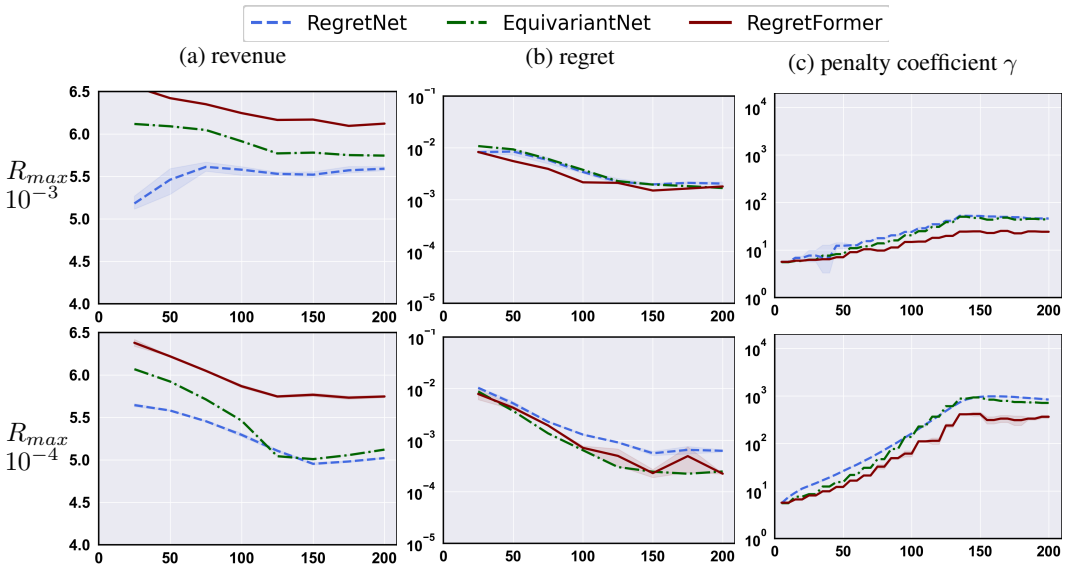

Figure 5: Learning curves in the setting 3x10. The X-axis is in the thousands of training iterations. The shaded regions correspond to the min-max spread over three random seeds.