INFORMS Journal on Computing

Online Supplement "Learning Equilibria in Asymmetric Auction Games"

Martin Bichler*, Stefan Heidekrüger, Nils Kohring
Technical University of Munich, Department of Computer Science, 85748 Garching, Germany

bichler@in.tum.de

---

1. Performance and Runtime Analysis

In the following, we look at the influence of some of the hyperparameters on the performance and runtime of the computation. We will illustrate that based on the example of the FPSB LLLLGG auction format.

1.1. Influence of batch size

The batch size corresponds to the number of auction games that are simultaneously played under the current strategies. Strategy updates are solely based on any changes in the utility estimate averaged over these batches, thus making the batch size one of the most crucial parameters of NPGA. Except for the parameter of interest (the batch size here), we leave all other parameters unchanged from their default values from Section 6 in the main paper.

Results for different batch sizes can be seen in Figure 1 for the first 500 iterations. As expected, lower batch sizes dramatically increase the variance in the utility estimates (see left plot) and thus prevent quick learning (see right plot).

Furthermore, the average time per iteration increases slowly as to be expected by vectorized GPU implementation. The average time per iteration for batch sizes of 64, 1,024, and 262,144 are 0.3717 ($\pm$0.0878), 0.3741 ($\pm$0.0757), and 0.4681 ($\pm$0.0283) seconds, respectively. Only once the available memory and available cores run out, the computation would have to be done in a sequential manner, which would lead to a drastic increase in computation time.

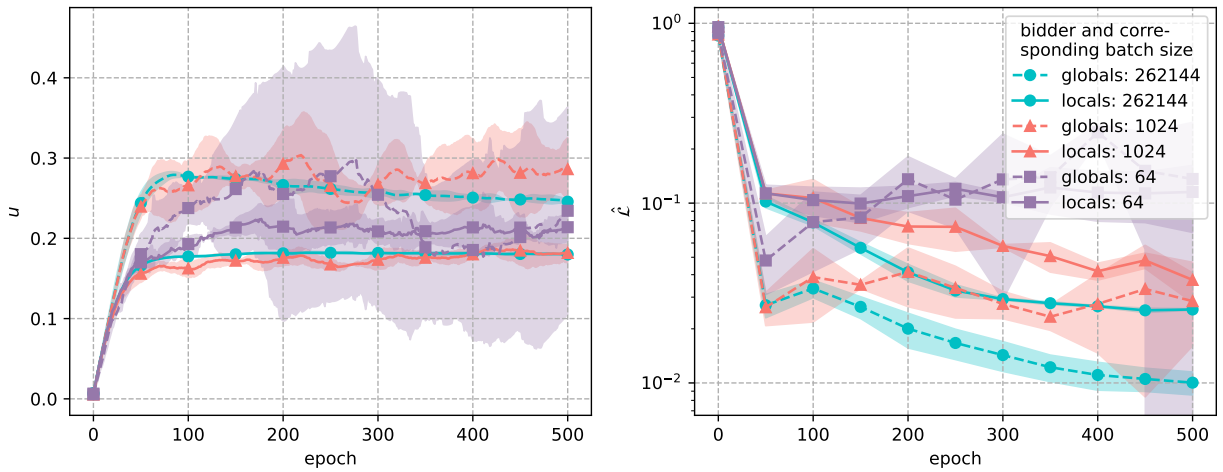

Figure 1 **Average utility of both bidder types (left) and relative utility loss (right) while learning via NPGA in LLLLGG first-price auction. Runs for three different population sizes, where each configuration was run three times (mean ± std), are depicted.**

1.2. Influence of population size

The population size refers to the number of sampled parameters that are considered during one estimation of the pseudogradient. Even though the needed utility estimates for each sample are independent of one another and could thus in principle be calculated in parallel, this is not implemented in favor of larger batch sizes that allow for better utility estimates. Therefore, it influences the running time linearly. Results for different population sizes can be seen in Figure 2 for the first 500 iterations.

As can be seen, the utility for larger population sizes changes quicker while the utility loss also decreases faster. The influence when increasing the population size is not as strong compared to changes in the batch size. This justifies using most of the GPU memory for sampling a large number of auction games for precise utility estimates.

Furthermore, the average time per iteration increases quickly as to be expected by the sequential computation of the fitness of individual population samples. The average time per iteration for population sizes of 16, 32, and 64 are 0.1264 (±0.0042), 0.2426 (±0.0162), and 0.4687 (±0.0283) seconds, respectively.

The observations extend qualitatively to other auction formats and payment rules except for an increased run time for core selecting payment rules.

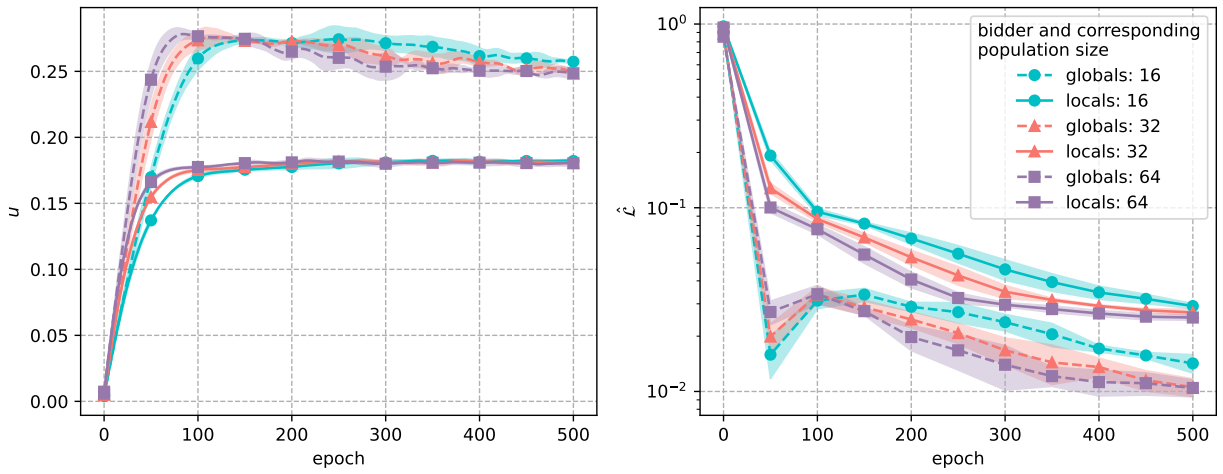

Figure 2 **Average utility of both bidder types (left) and relative utility loss (right) while learning via NPGA in LLLLGG first-price auction. Runs for three different batch sizes, where each configuration was run three times (mean $\pm$ std), are depicted.**

2. Approximation quality of utility loss

As we rely on the approximate utility loss $\hat{\mathcal{L}}$ from Equation 5.4, it is essential to have a reference of its estimation quality. Therefore, we have run the calculations for a variety of values for n_{batch} and n_{grid} in equilibrium for the weak bidder in the single-item auction with overlapping valuations. Of course, here the exact $\mathcal{L}$ is known to be zero. Figure 3 shows the results.

Let us emphasize two observations. First, counterintuitively, when increasing the grid size (comparing a finer grid of possible best responses), the utility loss increases. This is due to the bias introduced by choosing the best responses that maximize the utility of a particular sample, thus always selecting just that action that exploits the particular sample the most. One sees, however, that this effect decreases in severity with increasing grid sizes. Second, the right plot clearly shows the massive computational advantage of GPU computation as the calculations can be run in parallel, having nearly constant computational time for increasing batch sizes, as long as the tensor fits in memory.

In terms of scalability one has, on the one hand, the number of opposing agents that linearly affect the memory requirements, and on the other hand, the number of objects for sale that makes it exponentially harder to find a best response with the same level of accuracy.

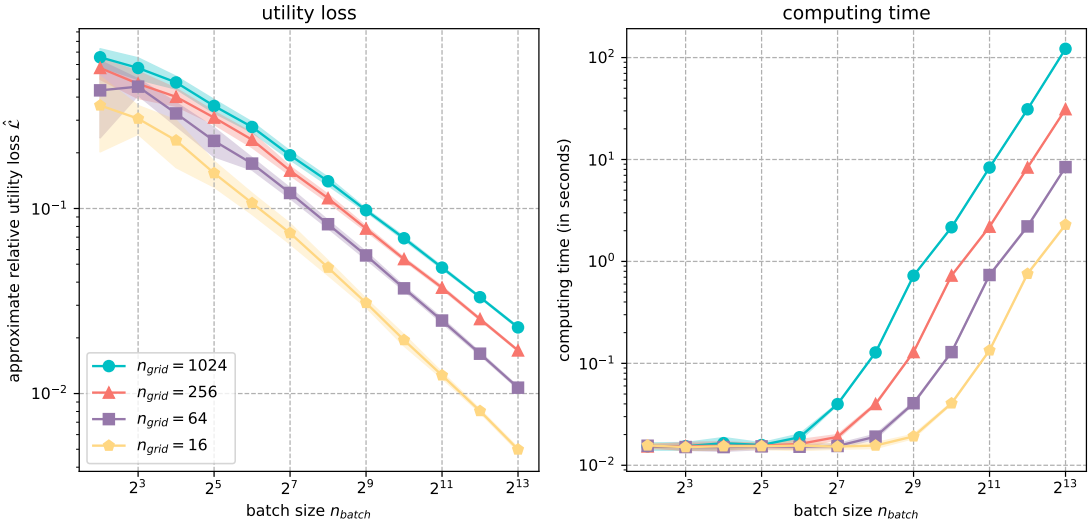

Figure 3 **Approximation quality of the utility loss (left) and computing time (right) for different batch sizes and different grid sizes averaged over ten runs each. The utility loss for the weak bidder in the single-item auction with overlapping valuations from Subsubsection 6.1.1 is depicted. Note that all axes are log-scaled.**

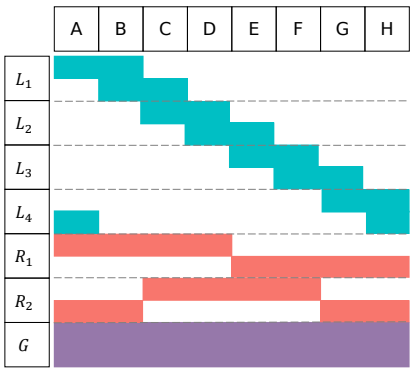

Figure 4 **Valuations in the LLLLRRG auction model. The columns depict items A through H and the rows correspond to the local, regional, and global bidders.**

3. Valuations in the LLLLRRG auction

As introduced in Subsection 6.5 of the main papaer, Figure 4 gives an overview of which agents are interested in which set of items (bundles) in the LLLLRRG auction.