Domain	Elicitation Method	# of Queries / Bidder	Elicitation Efficiency
GSVM	No Elicitation	0	22.0% (0.9%)
	Random Query	50	68.8% (0.7%)
	ML-based	≤ 50	98.5% (0.1%)
	Full Elicitation	2^{18}	100.0% (0.0%)
LSVM	No Elicitation	0	20.3% (0.6%)
	Random Query	50	62.5% (0.8%)
	ML-based	≤ 50	93.5% (0.4%)
	Full Elicitation	2^{18}	100.0% (0.0%)
MRVM	No Elicitation	0	32.7% (0.6%)
	Random Query	100	51.5% (0.4%)
	ML-based	≤ 100	93.3% (0.1%)
	Full Elicitation	2^{98}	100.0% (0.0%)

Table 1: Elicitation efficiency and average number of queries asked under different elicitation methods assuming truthful reports. Standard errors are reported in parentheses. All results are averaged over 100 auction instances.

queries to achieve high efficiency. In our experiments, we found that, for the same number of queries, quadratic kernels provided about a 2% gain in efficiency in MRVM, and a 10% gain in GSVM and LSVM. Accordingly, we present results for quadratic kernels; results for linear kernels are qualitatively similar at a correspondingly lower efficiency.

The integer programs (IPs) used to find the allocations a^t were solved with CPLEX_Studio (version 1261). We set a time limit of 1h for solving each IP and, when the time limit was reached, adopted the best solution found so far. We conducted our experiments on machines with Intel Xeon E5-2650 v4 2.20GHz processors with 40 logical cores. The $n+1$ elicitation runs in PVM were run in parallel. Since $n = 10$ in our largest models, we had up to 11 elicitations running in parallel, and thus allocated 3 logical cores to CPLEX for each elicitation run, such that we never requested more than the available number of cores on a single machine.

6.2 Evaluation of ML-based Elicitation

In this section, we evaluate our ML-based elicitation algorithm (i.e., a single run of Algorithm 1). We compare it against three baselines. First, we consider *No Elicitation*, which does not query any values and allocates each item to a bidder selected uniformly at random. We include this "degenerate" elicitation method as a baseline to obtain a lower bound on efficiency that can trivially be achieved without any elicitation. Next, we consider *Random Query*, which asks each bidder her values for a pre-specified number of bundles, whereby the bundles are selected uniformly at random in the bundle space without replacement. The allocation is then selected by solving the winner determination problem based on those reported values. This baseline represents the performance of uninformed sampling from the bundle space without any ML-based elicitation. Lastly, we include *Full Elicitation*, which leverages the special capability of SATS to directly encode value functions in a winner determination MIP so that a full value profile can be implicitly evaluated without requiring an otherwise infeasible enumeration of, e.g., 2^{98} goods. This represents an upper bound on efficiency because full elicitation is generally infeasible in practice.

Domain	Evaluation Uncertainty (μ)					
	0%		1%		5%	
GSVM	98.5%	(0.1%)	98.0%	(0.2%)	96.3%	(0.2%)
LSVM	93.5%	(0.4%)	93.1%	(0.5%)	91.2%	(0.6%)
MRVM	93.3%	(0.1%)	92.3%	(0.3%)	85.8%	(0.4%)

Table 2: Elicitation efficiency of ML-based elicitation with upper and lower bounds on values. Standard errors are reported in parentheses. All results are averaged over 100 auction instances.

Table 1 presents the results from running these baselines as well as our ML-based elicitation algorithm. Despite only querying a small number of values from each bidder, our ML-based elicitation algorithm achieves more than 98% efficiency in the GSVM domain, and more than 93% efficiency in the more complex LSVM domain and in the highly realistic MRVM domain. The *No Elicitation* results show that all domains exhibit complex value structures such that high efficiency cannot be achieved by simply assigning items at random. The *Random Query* baseline results show that the ML algorithm used in our elicitation has a huge effect on efficiency. In MRVM, this effect translates into an average efficiency increase of more than 40% points.

Elicitation with Upper and Lower Bounds. It is often easier for bidders to specify bounds on bundle values, rather than determine such values precisely. We have therefore developed a modification of Algorithm 1 that works when bidders report upper and lower bounds instead of exact values.

To adapt Algorithm 1 to handle bounds, the key step to modify is Step 4 of the algorithm, i.e., how to apply ML algorithm $\mathcal{A}$ on reports consisting of bounds. This works particularly well when SVRs are used as the ML algorithm. Specifically, we can then leverage the particular structure of the ε-insensitive loss function used in standard SVR, presented in Equation (11). Note that this loss function does not penalize any linear model w for which $w \cdot \varphi(x_{ik}) \in [\hat{v}_{ik} - \varepsilon, \hat{v}_{ik} + \varepsilon]$. This means that we can employ the following trick: we consider any report consisting of an upper bound $\hat{v}^u_{ik}$ and a lower bound $\hat{v}^\ell_{ik}$ as if the bidder had reported value $\hat{v}_{ik} = (\hat{v}^u_{ik} + \hat{v}^\ell_{ik})/2$. As long as we also measure the interpolation error in this sample via an insensitivity loss with $\varepsilon_{ik} = (\hat{v}^u_{ik} - \hat{v}^\ell_{ik})/2$, we guarantee that the semantics of the bidding bounds are maintained. Thus, this trick allows us to compute the inferred social welfare function in Step 4 of the algorithm with bounds as inputs. The remainder of Algorithm 1 remains the same (except that we operate on bounds instead of exact values), and the algorithm terminates in round T when bidders have specified at least one upper and lower bound for all bundles they are allocated in a^T.

Note that with this modification, the output of Algorithm 1 now contains bundles and upper/lower bounds (instead of exact values). This is not yet enough information to determine an optimal allocation. Consequently, one needs an additional *refinement process* that asks the bidders to refine their