Quan Nguyen
quan@wustl.edu

Research note
April 17, 2020

1 On the real data

We rerun the computation of the efficient and the leximin assignments on the real data set, now without the `PSH` column. To ensure that the number of agents matches the total capacity, we also filter out the agents that are assigned to `PSH` in real life. Figure 1 visualizes the distribution of the original probabilities and the normalized probabilities in this new filtered data set:

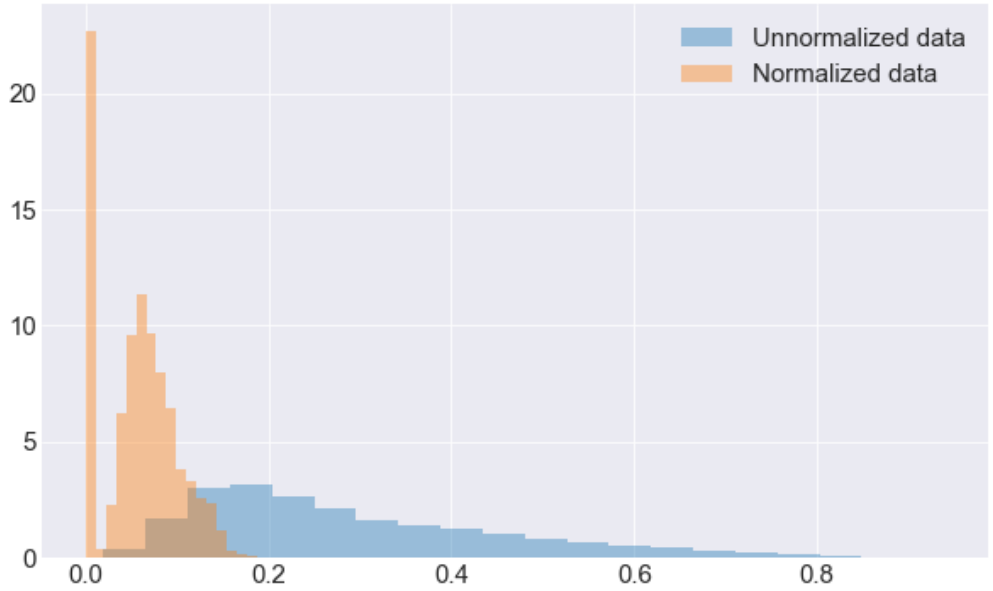

Figure 1: Distribution of probabilities in the real data set

Running the efficient and leximin assignments and comparing them with the original/real assignment, we obtain the following results:

	Efficient	**Leximin**	**Original**
Unnormalized probability sum	3682.021	3692.081	3998.077
Normalized probability sum	383.637	393.697	699.692
Bottleneck of normalized probability	0.080	0.075	0.221
PoF wrt unnormalized probabilities	1.00273		
PoF wrt normalized probabilities	1.02622		

Table 1: Efficient vs. Leximin vs. Original assignment

Figure 2 visualizes the distribution of normalized probabilities assigned to agents under the leximin, efficient, and original assignments:

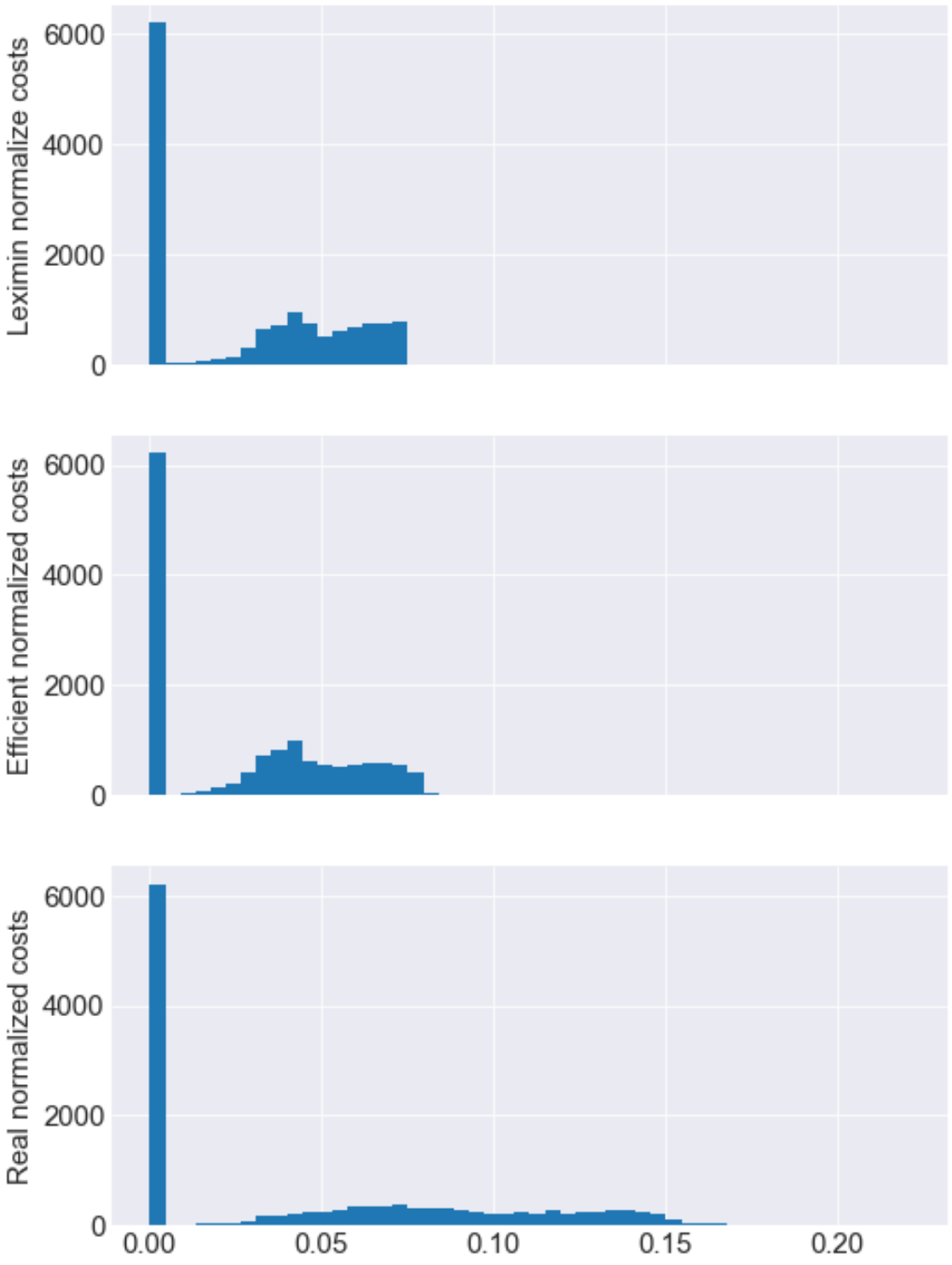

Figure 2: Assigned normalized probabilities

2 When rows are sorted

In this section, we consider the behavior of PoF when the rows in our cost matrices are sorted under different situations.

2.1 The greedy leximin assignment

Under the assumption of sorted rows, we can first prove that the least effective intervention will be filled with agents with the lowest costs for that intervention. (Otherwise, there would exist a pair of agents who, when their assignments are switched, will have their larger cost lowered.) From here we can prove the following upper bound for the PoF quantity:

$$\mathrm{PoF}(L) \leq 1 + \frac{n - c_m - c_1}{c_m \ \underline{b}}$$

where n is the number of agents, c_1 and c_m are respectively the capacities of the most and least effective interventions, and $\underline{b}$ is the lowest cost in the least effective intervention.

In the special case where the interventions all have equal capacities, the bound will become:

$$\mathrm{PoF}(L) \leq 1 + \frac{m - 2}{\underline{b}}$$

where m is the number of interventions.

We can also prove that the leximin assignment corresponds to the greedy algorithm that sequentially assigns agents with the lowest costs with respect to individual interventions (going from the least effective to the most effective intervention) until they are filled.

It is not obvious how tie-breaking should be done. In the simulations subsequently described, if a cost matrix was generated to contain elements that would lead to a tie during the execution of the greedy algorithm, that specific experiment would be skipped and repeated.

2.2 Simulations

2.2.1 Sorted vs. not sorted

We first compare the distribution of PoF when the matrices are randomly generated from specific distributions against when they are additionally row-sorted. Each experiment was with a 30-by-5 cost matrix that was generated by drawing from one of nine different distributions and its version where the rows are sorted. 500 experiments were used for each distribution.

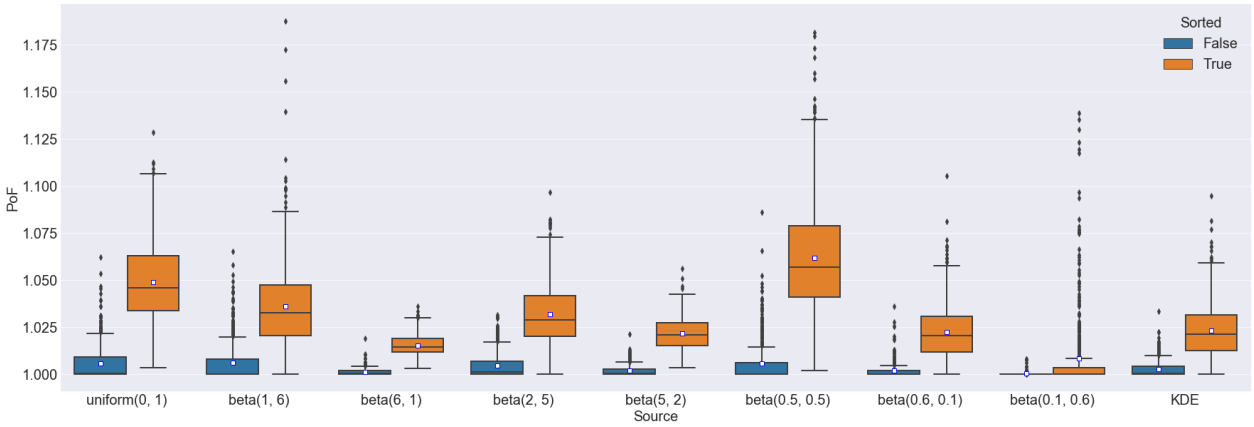

Figure 3: Distribution of PoF - sorted vs. not sorted

It seems that overall, row-sorted matrices give lower-valued PoF than randomly generated ones. For each of the experiments above, the ratio of PoF resulting from its original matrix and the row-sorted matrix was also recorded. The distribution of this ratio is visualized above:

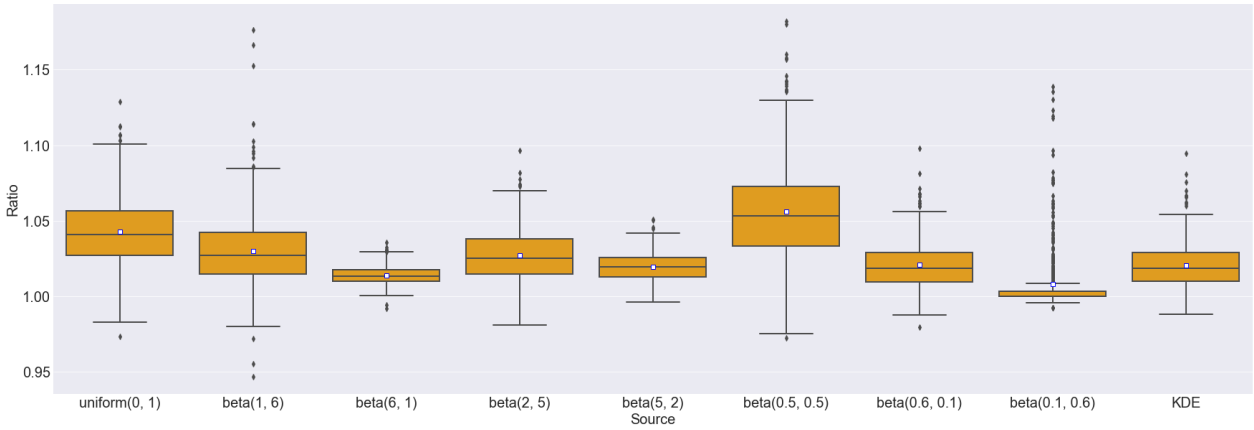

Figure 4: Distribution of PoF ratio - sorted vs. not sorted

We see that it is not always the case that the ratio is greater than 1, or in other words, going from a row-sorted matrix to the original randomly generated version does not always result in a lower PoF.

As an additional note, the following visualizes the distribution of costs and their normalized values with respect to the different source distributions above:

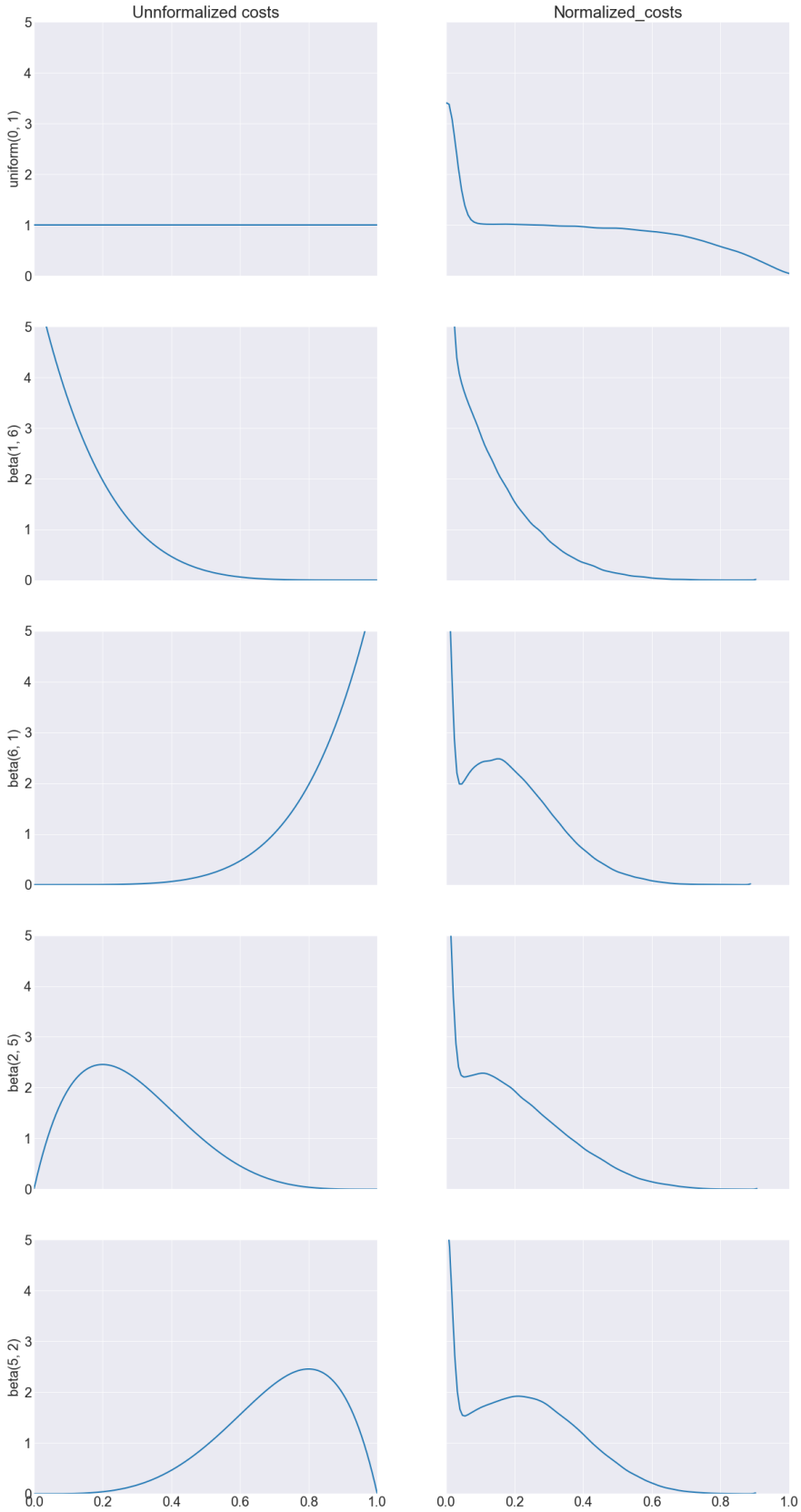

Figure 5: Distribution of normalized costs

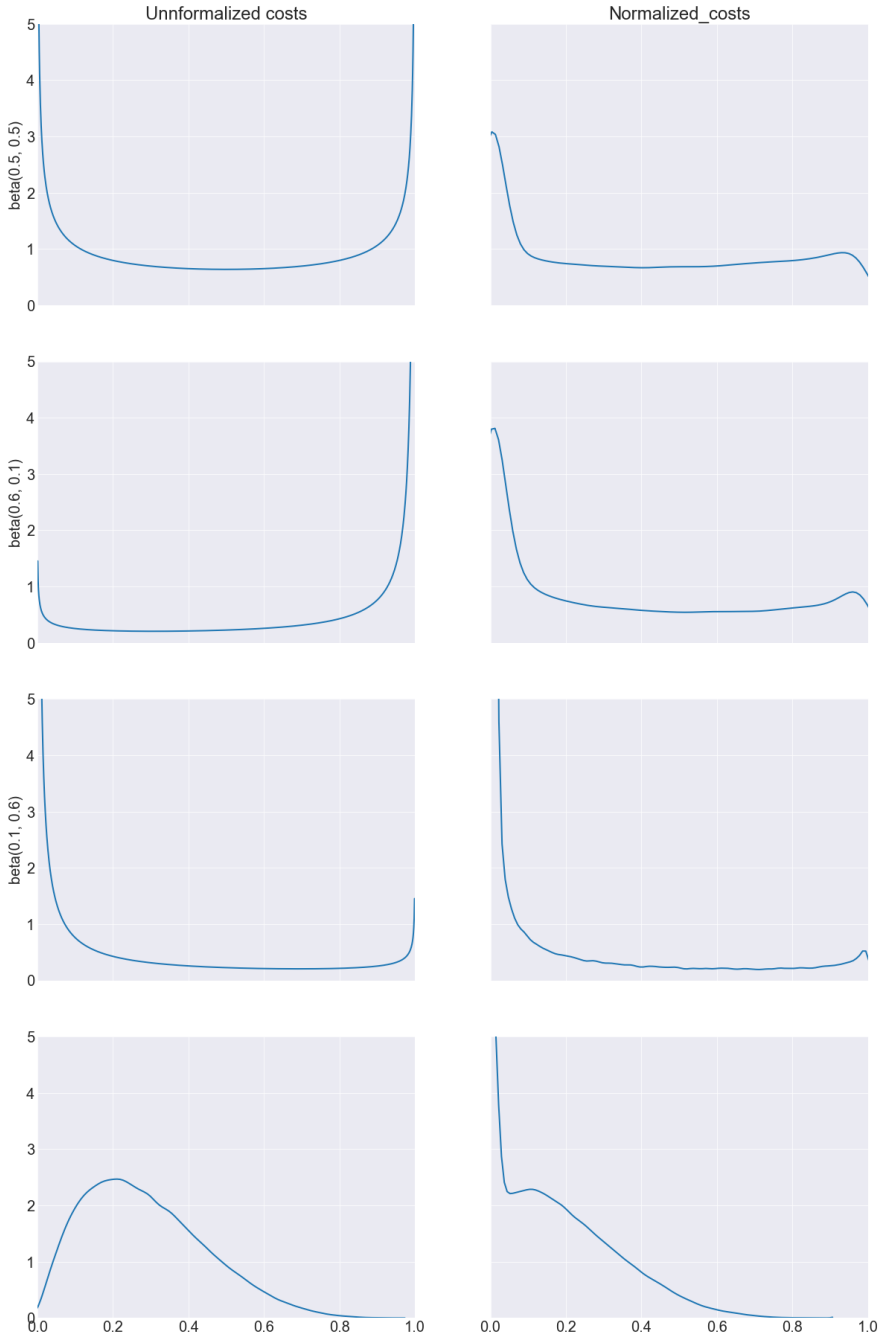

Figure 6: Distribution of normalized costs

We now consider the average PoF across different combinations of α and β as parameters for the cost distribution. (For each combination of α and β, 1000 simulations were used. PoFs were generated from row-sorted matrices.)

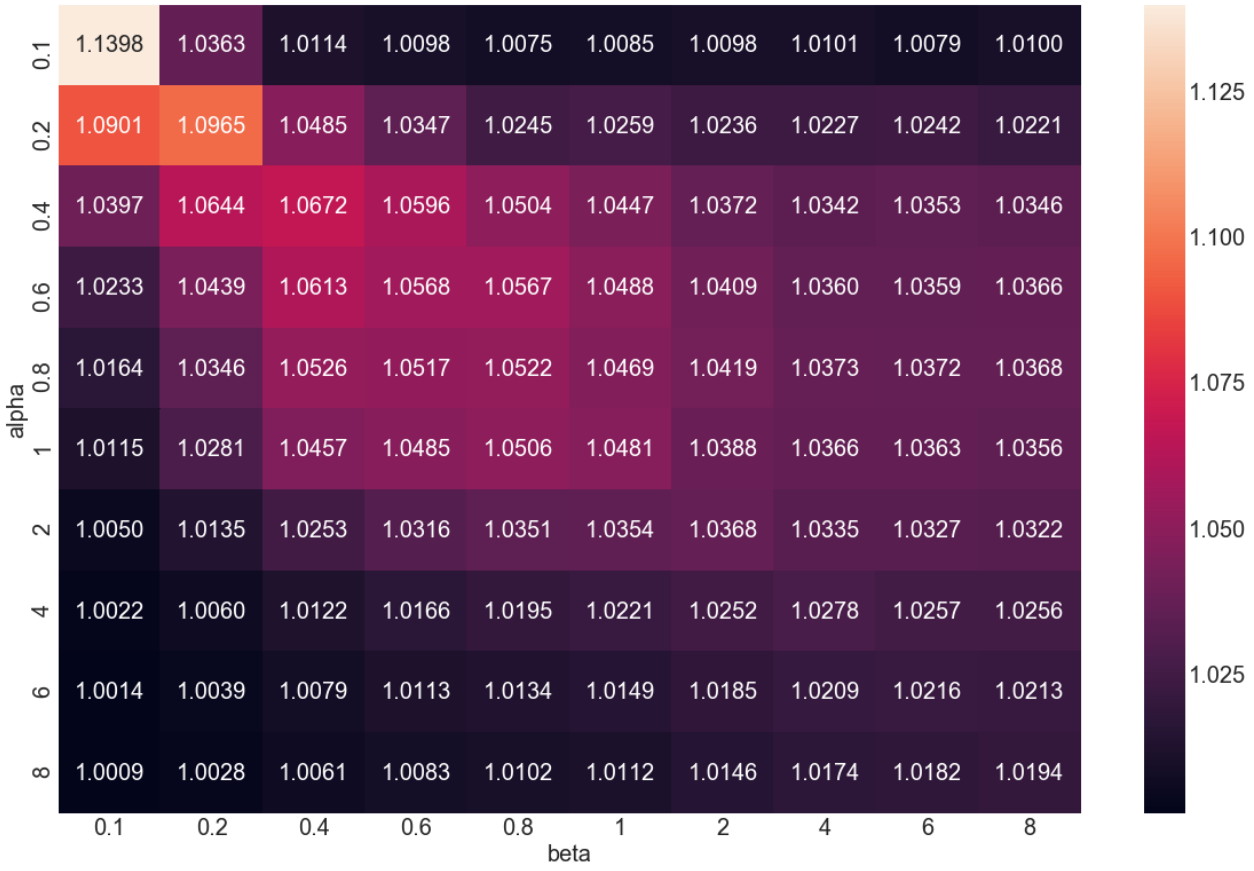

Figure 7: Average PoF with different α's and β's

2.2.2 Number of interventions

Here we consider the effect of the number of interventions available (the number of columns in the cost matrix) on the PoF distribution. The individual rows in our cost matrix are still sorted. The following results were from 1000 simulations where elements in the cost matrix were drawn from the uniform distribution. The number of agents were fixed at 40 and all interventions had equal capacities.

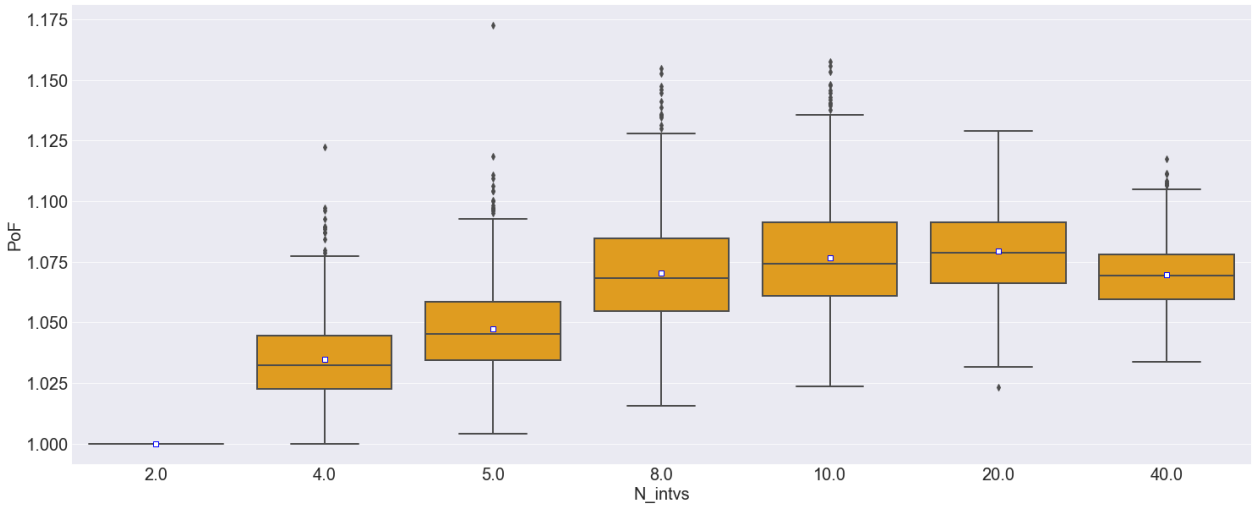

Figure 8: Uniform costs with varying number of interventions

The same process is set up where the costs are drawn from the KDE on the real data, which results in the following:

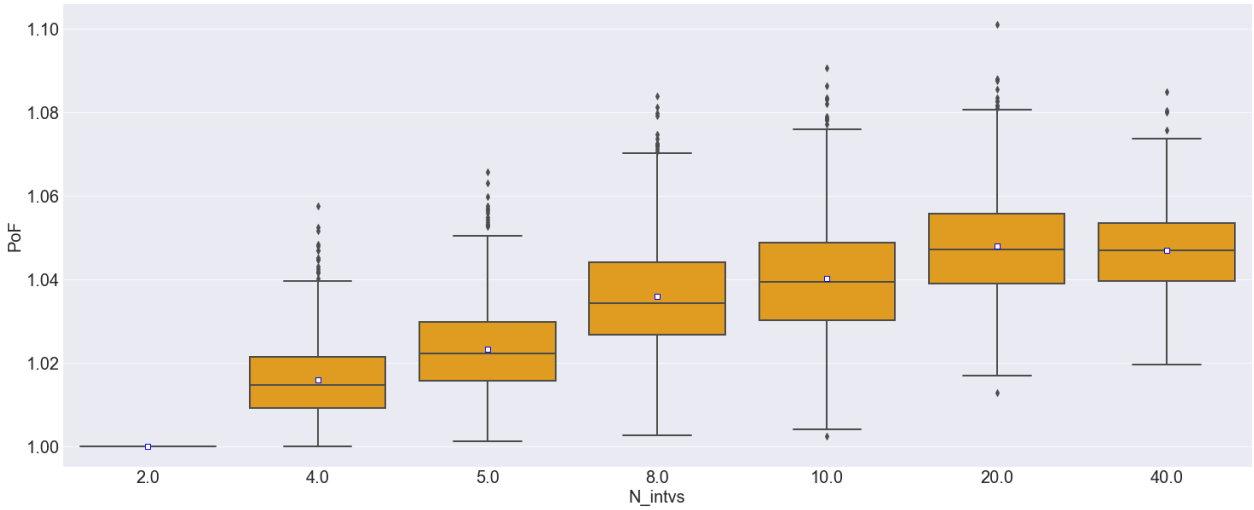

Figure 9: KDE costs with varying number of interventions

2.2.3 Percentage of the most common order

Relaxing the assumption of sorted rows, we consider the PoF distribution with respect to the number of rows with the most common row order.

For this simulation, 100-by-5 matrices were generated, each with a specific threshold; this threshold specifies the number of rows in the matrix that will be sorted post-generation. Then the actual number of rows in the most common order of the matrix is recorded along with the resulting PoF for each experiment. We group the experiments whose numbers of rows in the most common order fall into the same ranges together and consider the PoF distribution of each group. (Each group is guaranteed to have at least 500 experiment members.)

The result below is from the experiments where costs were drawn from $U[0, 1]$. For each group range, the left endpoint is inclusive and the right endpoint is exclusive:

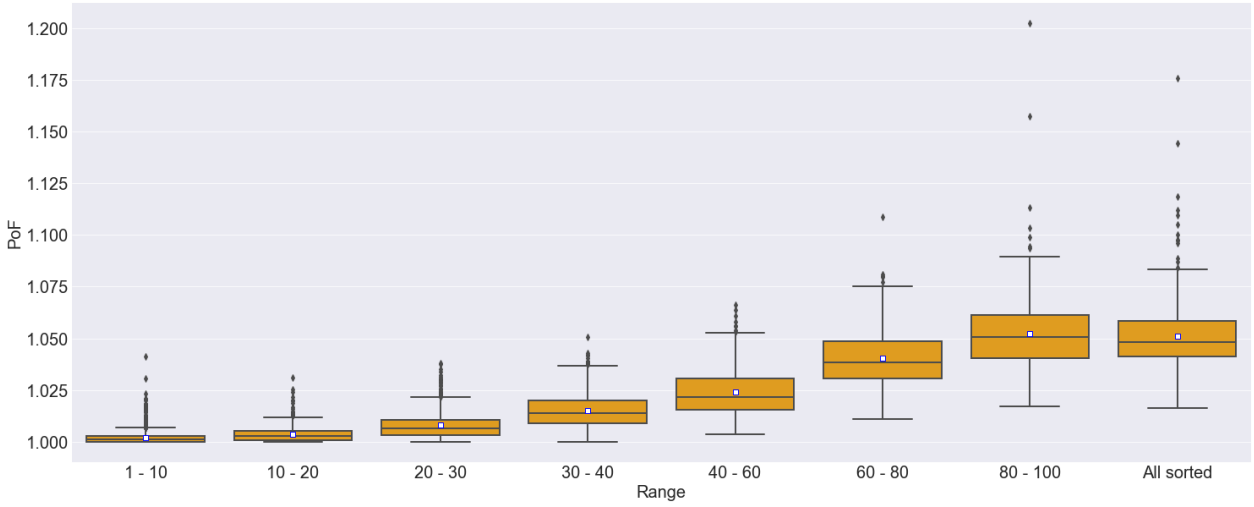

Figure 10: Uniform costs with varying row order heterogeneity

The same process is set up where the costs are drawn from the KDE on the real data, which results in the following:

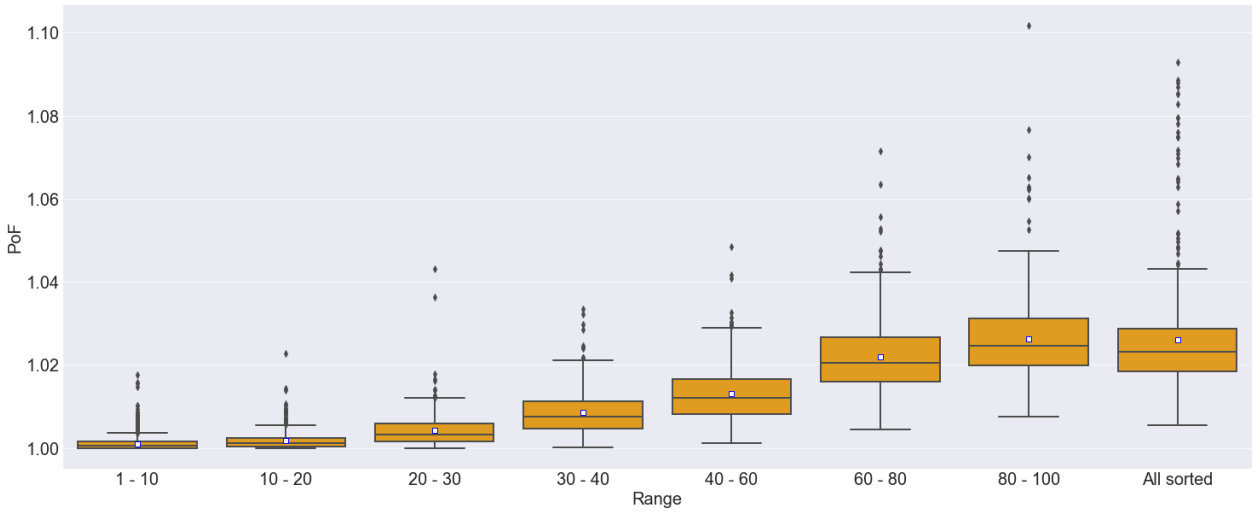

Figure 11: KDE costs with varying row order heterogeneity