Combinatorial auction

Problem

A combinatorial auction is when the bidders specify a value for a subset of items not only for one item. It makes sense when the bidder needs all the subset and it doesn't make sense to bid for individual items.

This poses a problem for the auctioneer, how they decide who won the auction as two bids can have common items and you can't split them.

Use Cases

For example, a company auctions transportation services, for a client it only makes sense if they can win the whole trip, not only a faction. This model can be used in multiple fields, raw material procurment where a profit can be made if only the whole materials can be obtained for a given value.

LP Model

Our model uses a boolean variable to determine if a bid was accepted or not.
It tries to maximize the sum of money it can collect and has the restriction that for a given item you can only take a bid that contains that item.

$$I \text{ is the item set}$$
$$B \text{ is a set of bids, } B \subseteq I^*$$
$$v : B \rightarrow R \text{ is a function that maps a bid to the amount offered}$$

The variables are
$$x : B \rightarrow \{0, 1\}$$

We try to maximize the gained value

$$max\left(\sum_{b \in B} x(b) * v(b)\right)$$

And we have the following restrictions

$$\forall i \in I \quad \sum_{c \in \{b \mid \forall b \in B \text{ if } i \in b\}} x(c) <= 1$$

Dataset

We found a comprehensive dataset with examples ranging from 100 bids and 400 items to 2000 bids and 50000 items, with various variations and distributions.

The entire dataset consists of **630** samples adding up to around **166MB** of data.

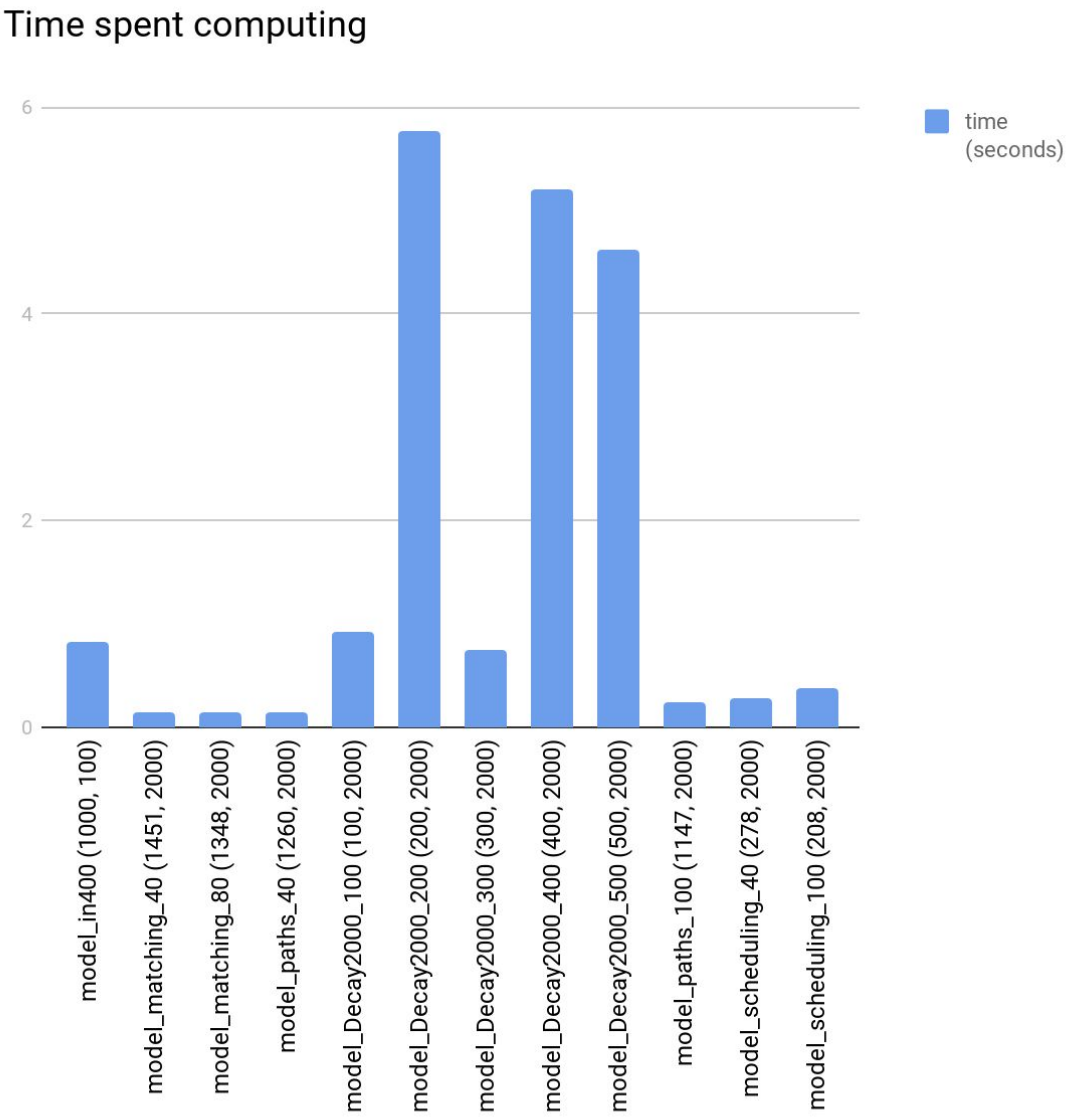
Unfortunately the datasets do not provide the optimal results.

Inițial experiments

We implemented this model in PULP that can use multiple solvers (it translated the model to the appropriate syntax depending on the solver)

The first solver we used was CPLEX, it worked fine for synthetic datasets but with the variable limitation, it can't process any of the samples we have.

The next one we used was GLPK (GNU Linear Programming Kit) which managed to find optimal solutions for a few samples.



Here we will run all the samples with a timeout limit in order to have a benchmark and solution for the whole dataset.

We also had a **dataset model_uniform2000_100_10 (100, 2000)** that took **7747** seconds to optimize that was not included in the graph.

Future plans

- We would like to also try Groby and compare it the free tool (this will help us in the future)
- Run the model for all the samples with a generous timeout, in order to compare our heuristic solution in the same scenarios