Final results

Our first LP model was run over the whole dataset with a timeout of 1 min and 5 min so we can get a baseline performance and profit we can make to compare our heuristic implementations.

The first heuristic was a series of greedy algorithms that are deterministic and extremely fast (compared to the LP model) and give relatively good results.

Greedy Implementations

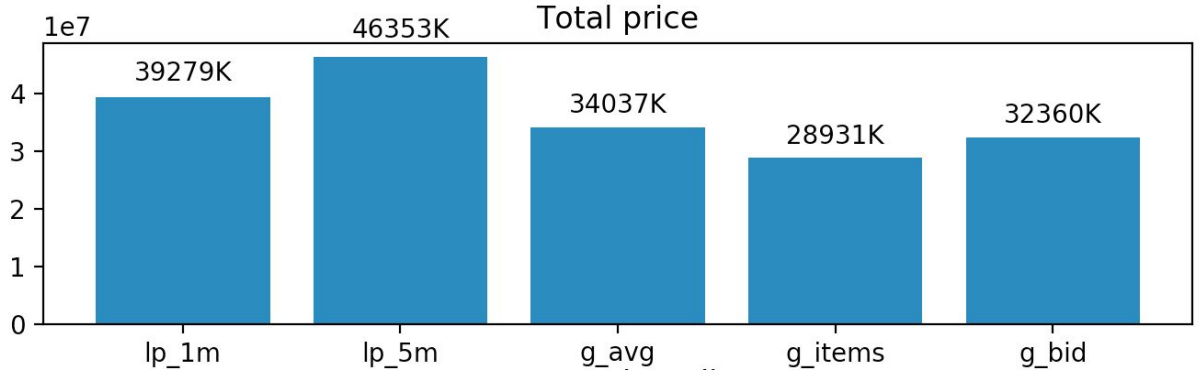
First, we looked at the number of items and chose the nonconflicting bids based on this metric. This one performed the most poorly because the number of items is not an indicator of profit also, taking bids with a lot of items results in a lot of conflicts.

The second approach was based on the bid price, this prioritized highed bids and allowed us to improve our total profit on the dataset, but was not looking at the items (even if a big price is high it may still undervalue the positions.

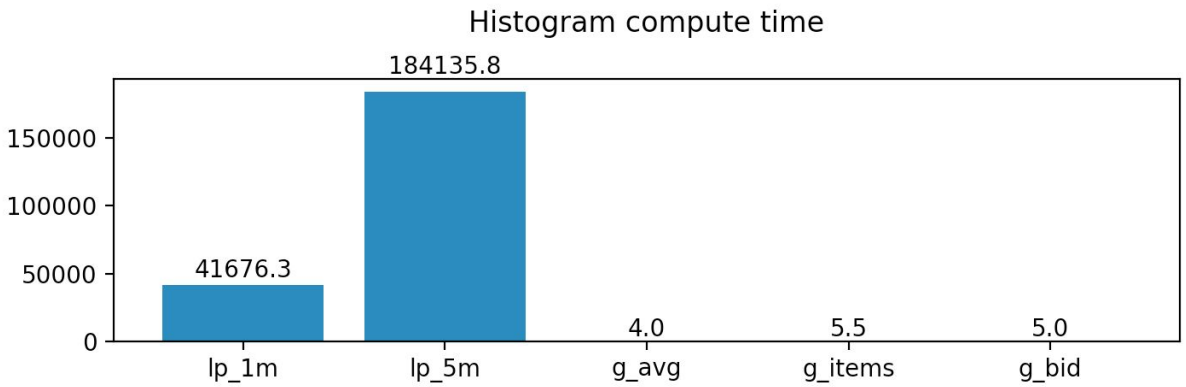
Our last approach and the most successful one where for each bid we computed the average price per item and then applied a greedy approach to choosing the best bids with no conflict.

In the graph below we can see the cumulated profit of the experiments.

- *lp_1m* is the LP model run with a timeout of 1 minute
- *lp_5m* is the LP model run with a timeout of 1 minute
- *g_avg* is the greedy approach taking the average items price into account
- *g_items* is the greedy approach taking the number of items price into account
- *g_bid* is the greedy approach taking the bid value into account



Overall the greedy solution performed poorly compared to the LP model but the speedup in processing time is considerable, in the graph below we can see the total time used to solve resolve all the datasets in seconds.
We can see that the greedy are almost instantly on the whole dataset and can offer a good tradeoff if speed is more important.



Ant Colony Optimization

The next heuristic model was based on Ant Colony Optimization [1] where each ant is initialized with an empty set of accepted bids and the algorithm is as follows

pheromone_power = 0.5
greedy_power = 0.5
pheromon_decay=0.9
Initialize pheromone for each bid equally
Initialize ants with an empty list
while timeout is not reached
    for each ant
        compute the probability to choose a bid
        for each conflicting bid to update the probability with 0 (avoid conflicts)
        choose with the probability computed
    update the pheromone trail based on ant fitness
    evaporate pheromon trail

One important step is probability computation, here we got inspiration from an existing paper An Ant Colony Approach for the Winner Determination Problem[2] that recommends using the following formula

$$probability\ for\ bid\ =\ pheromone^{\ pheromone\ powrt} + greedy\ evaluation^{\ greedy\ power}$$

where greedy evaluation is the average item price for the bid.

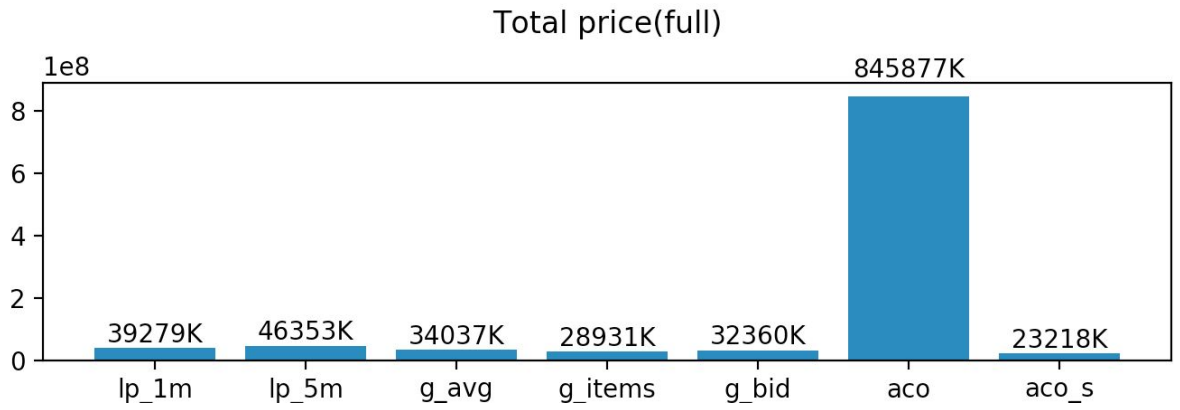
We can see that if we set *pheromone power* to 0 we gave a probability-based greedy implementation.

---

[1] "An Ant Colony Approach for the Winner Determination Problem." https://www.researchgate.net/publication/323612691_An_Ant_Colony_Approach_for_the_Winner_Determination_Problem. Accesat pe 17 ian.. 2020.
[2] "An Ant Colony Approach for the Winner Determination Problem." https://www.researchgate.net/publication/323612691_An_Ant_Colony_Approach_for_the_Winner_Determination_Problem. Accesat pe 17 ian.. 2020.

We choose the average item price as the metric because it performed better in the previous experiments.

One improvement that we tried was to increase the number of ants from 10t to 1000 hoping to encourage exploration with more ants.

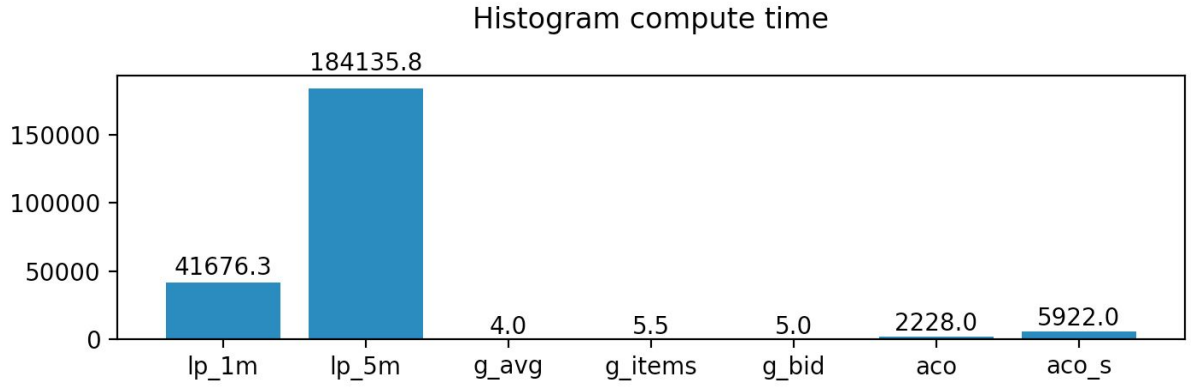


We can see that this approch offers huge improvements over all other implementations.

- *aco* is the run with 10 ants
- *aco_s* is the run with 100 ants

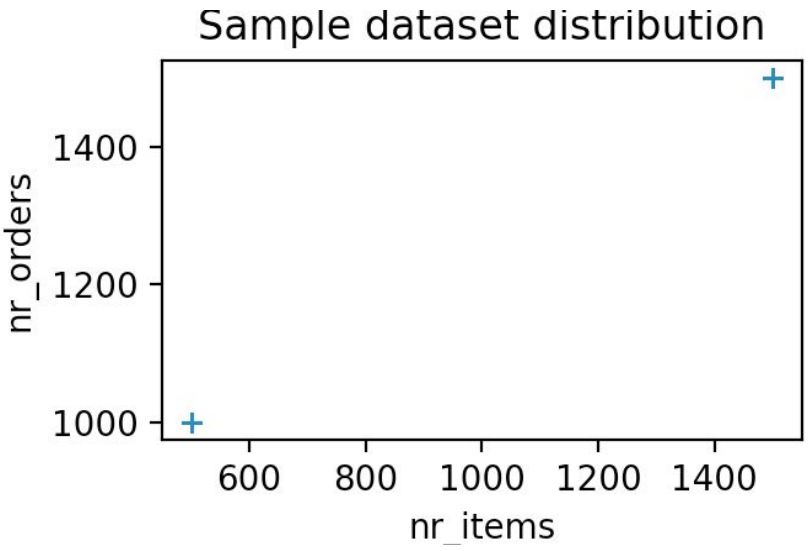
The number of ants combined with the short timeout actually had a negative effect, as the algorithm took looked to compute an epoch and this resulted in poor exploitation.

Looking at the total profit and time to compute that over the whole dataset.

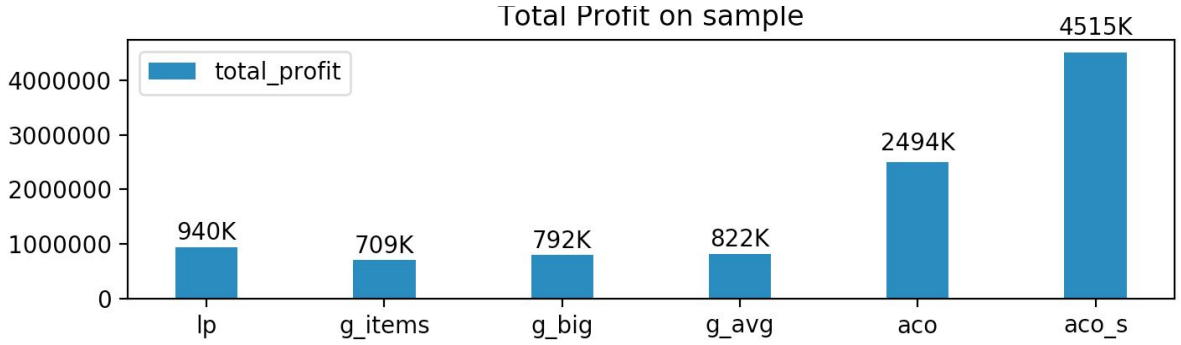


In order to properly understand the performance and bottlenecks of each algorithm we created a subset for rapid iterations.

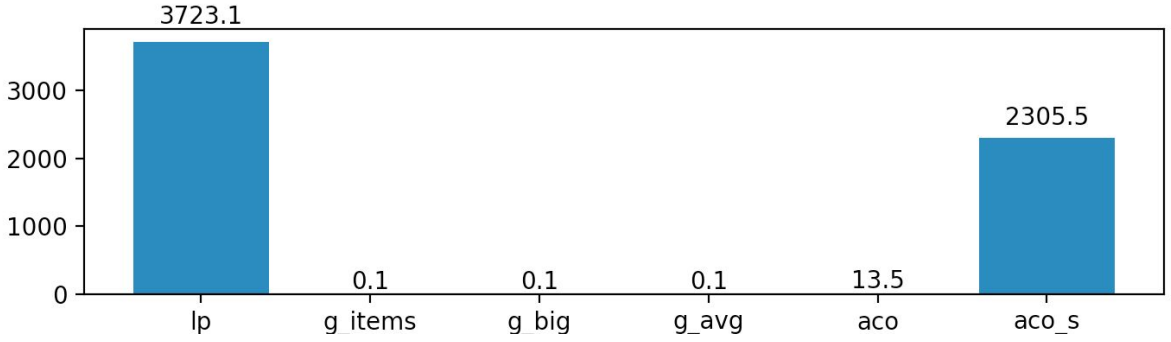
The distribution of datasets contained small datasets and large ones as we can see in the graph below.



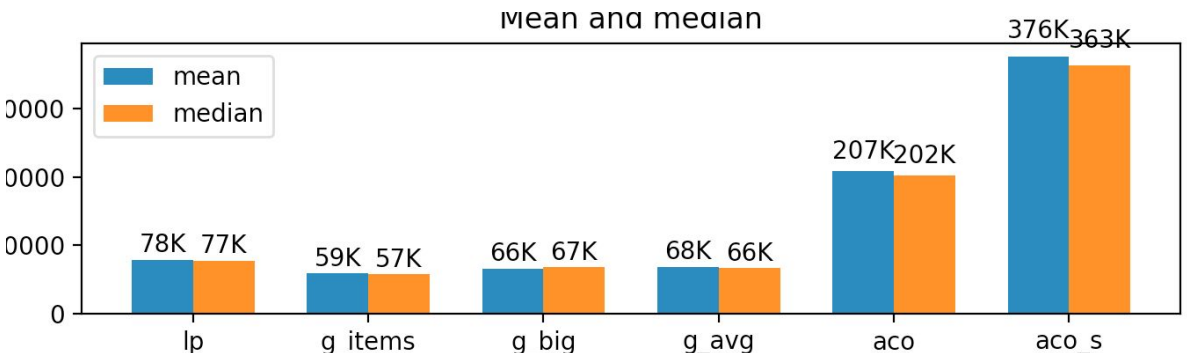
The total profit made on the sample also confirms that aco_s had the best performance.



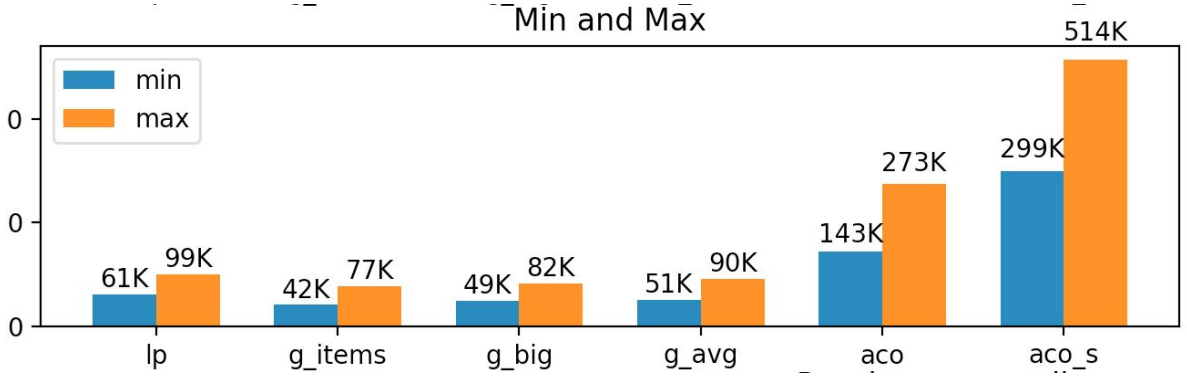
But it did not reach the timeout limit as LP model, this means we could have explored more and reach even better results. An improvement can be to dynamically add ants if we have time left to encourage exploration.



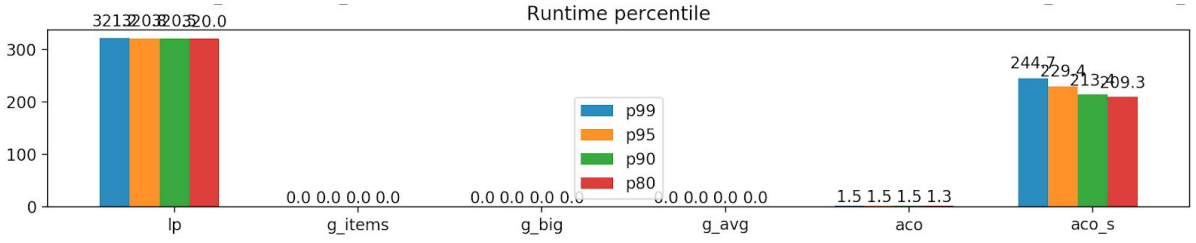
Looking at the mean and median profits we can see that better explorations allowed us to discover more profitable subsets of bids with.



Also looking at the Min profit we can see that aco_s managed to get better results then even the max profits. this means that none of the other algorithms solved a dataset optimally.



Looking at the runtime percentile we can see that aco_s did not use the whole available time like LP that was trying to optimize the solution. We know that lp did not reached an optimal solution from the previous graph but aco_s did not used the time given properly.



Conclusion

Ant Colony optimization is a good approch for this problem, giving better results then LP modeling if a partial answer is accepted. Given more time LP will finally reach the optimal solution.