Auction Algorithm for Bipartite Matching in Data Streaming model

Student: Yu(Ivan) Qin, Supervisor: Dr. Christian Konrad, Project Type: Research

University of Bristol, Department of Computer Science

Abstract

A recent paper [a] introduced a new multi-pass streaming algorithm, called New Auction Algorithm, with strong theoretical grantees in space and time. Compared with the previous state-of-art algorithm, this new algorithm reduced the number of iteration from $O(\frac{\log\log(1/\epsilon)}{\epsilon^2})$ to $O(1/\epsilon^2)$; also it reduced the space complexity from $O(n/\epsilon)$ to $O(n)$.
Our project gives the improvement of this algorithm in engineering, as well as the extension in theoretical analysis. At the same time, we will use visualization to prove its special feature in data streaming. Moreover, we will illustrate the performance of the algorithm on the real-world dataset and the worst dataset.

[a] https://epubs.siam.org/doi/pdf/10.1137/1.9781611976496.18

1. Introduction

Today, many algorithms have been helping us solve the problem of data explosion. Bipartite matching is one of the central questions in the Graph algorithm. With the Auction algorithm, we can introduce a solution to find the maximum matching. To solve the problem of massive data explosion, the algorithm should be suitable for the data streaming model, in other words, it should be possible to use less than $O(n \cdot ploy\log(n))$ bits of space.

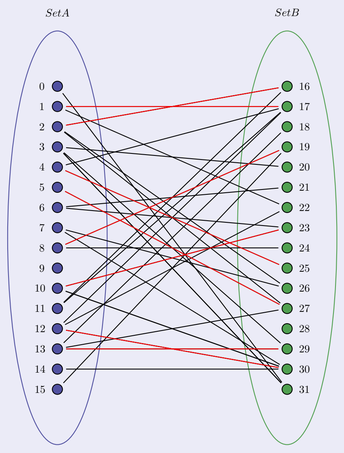


Figure: An example of bipartite matching

2. Motivations and Goals

In academia, bipartite graph maximum matching problems can migrate to other more difficult problems, such as minimum vertex cover, maximum independent set, etc.
In industrial, bipartite matching is also actively studied by many cloud computing companies, such as Facebook, Google, Uber, etc. The amount of their data in matching problems sometimes exceeds 90,000 operations per second.
Therefore, we are urgent to research the following property:

- The properties of this algorithm, such as Time and Space complexity, number of iterations.
- The performance on the real-world and worst-case data set.
- The further theoretical analysis of Auction Algorithm.
- The Comparison with the state-of-art algorithm.

3. Algorithm

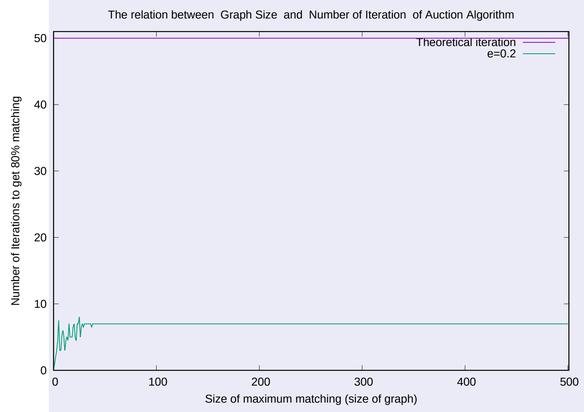
The input bipartite graph is interpreted as a collection of bidders on one side and items on the other, and the algorithms run an auction to find a welfare-maximizing assignment of items to bidders, which translates to a maximum matching of the input graph.
More specifically, in each iteration, we have the following 3 steps:

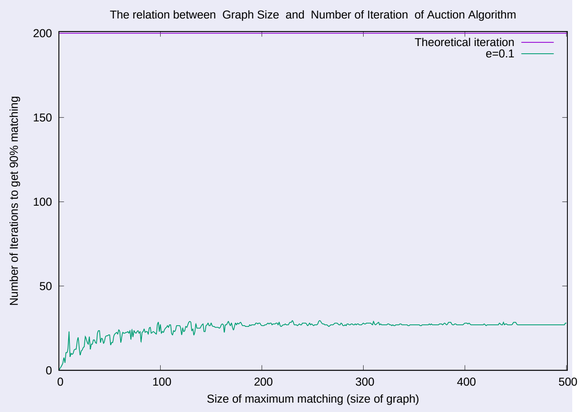
1. For each unallocated bidder, find the smallest price of its neighbourhood item and create a graph for them.
2. Find any maximum matching, say with the greedy algorithm.
3. Set allocation and update the price of each matching, if the item was connected, we reset the previous allocation to empty.

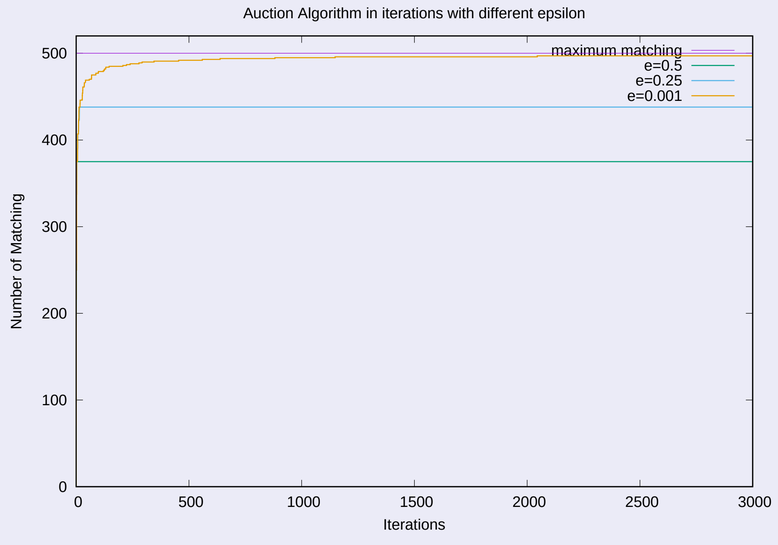
Here are video examples for random graph [b1] and semi-complete graph [b2].

4. Experiment and Results

- Implementation and Improvement
  The code is programmed in c++ with `boost graph library`; we also create a script to prepossessing the data with `SNAP Dataset`; Finally, we used `Gnuplot` and `Tikzpicture` to generate the visualization tools.
  The algorithm can still be optimized from an engineering point of view. We have done the following optimizations:
  1. Modify the data structure to HashMap, so that price and allocation are stored in the node of the graph, which is more convenient to find in $O(1)$.
  2. In the first step of iteration of the algorithm, if the graph is empty, we can exit the loop early.
- Dependency on epsilon







- Performance on Worst-case and Real-world dataset

	Nodes	Number of Iteration	Auction (second)	Auction with 90% result (second)	Build-in (second)
Semi-complete (Worst-case)	16	10	0.000052	0.000023	0.000018
Semi-complete	64	209	0.000709	0.000203	0.000116
Semi-complete	256	2965	0.032741	0.004156	0.003191
Semi-complete	1024	44644	0.443631	0.017393	0.019132
Facebook(Real-world)	4039	146	0.038141	0.005573	0.015142
Google_plus(Real-world)	107614	920	11.8238	1.73829	8.49265

5. Conclusion

- This algorithm is a deterministic data streaming algorithm for bipartite matching that for any $\epsilon > 0$ (not necessarily a constant) gives a $(1-\epsilon)$-approximation in $O(1/\epsilon^2)$ passes and $O(n)$ space.
- The dependency of epsilon is very low, and it is much smaller than the theoretical announced $O(2/\epsilon^2)$.
- Algorithm performs very fast at the beginning of the big data set, but very bad at the latter iteration. This algorithm is currently the best when we encounter answers that only require a partial (90%, *etc.*) matching.
  - For instance, 80% of the maximum matching, no matter the size of the graph, we only need around 7 iterations.
  - To achieve 90% of the maximum matching, we need around 30 iterations.
- More theoretical analysis and code will be available here [c].

University of BRISTOL