EE 382V: Social Computing **Fall 2018**

Lecture 6: August 25

Lecturer: Vijay Garg *Scribe: Ari Bruck*

6.1 Demange, Gale, and Sotomayor aka `DGS/Auction` Algorithm

After the discussion of the KuhnMunkres algorithm, the professor introduced a new algorithm. The algorithm will provide the best possible assignment of goods and bidders such that the prices are maximized.

Input : `Bipartite graph with non-negative integer weights on the edges`,
`B` : set of Bidders,
`G` : set of Goods,
$\mathtt{W}_{i,j}$: weight between bidder i and good j
Output : `Maximum weight matching`

Q : Queue of Bidders
P_j : Price of Good j
$Owner_j$: Current winning bidder of Good j
n_m : Size of the matching (In the example in class $n_g \leq n_b$ [more bidders than goods] so $n_m = n_g$)
δ : The incremental price increase in auction price as a result of a matching

Algorithm 1 DGS/Auction

for all goods j **do**
  $P_j \leftarrow 0$ ▷ Each good's price starts at 0
  $Owner_j \leftarrow$ `NULL` ▷ bidders do not own any good
$Q \leftarrow B$ ▷ all bidders are in the queue
$n_m \leftarrow \min(n_g, n_b)$
$\delta \leftarrow \frac{1}{n_m+1}$

while $Q \neq \emptyset$ **do**
  $i \leftarrow Q.dequeue()$
  find j that maximizes $W_{i,j} - P_j$ ▷ Find the good that has best "effective" payoff
  if $W_{i,j} - P_j \geq 0$ **then** ▷ If the good adds to the overall weight matching
    $Q.enqueue(Owner_j)$ ▷ Replace current owner
    $Owner_j \leftarrow i$ ▷ with new one
    $P_j \leftarrow P_j + \delta$ ▷ Increase the auction price for that good
return $(j, Owner_j) \forall j$. ▷ maximum weight matching has been found, return all goods and their owners

Correctness: The proof of correctness is based on showing that the algorithm gets into an equilibrium, a situation where all bidders "are happy".

Definition: Bidder i is δ-happy with respect to P if $\exists$ good j s.t. **either**:

$Owner_j = i$

AND

$\forall$ goods $j' : \delta + W_{i,j} - P_j \geq W_{i,j'} - P_{j'}$

OR

$Owner_j \neq i$

AND

$\forall$ goods $j : W_{i,j} \leq P_j$

Loop Invariant: $\forall$ bidders $\not\subset Q$ are δ-happy

Initially: `TRUE`
Q is initialized to all bidders

At Runtime: `TRUE`
For the bidder i dequeued in an iteration, the loop exactly chooses the j that makes him happy, if such j exists, and the δ-error is due to the final increase in P_j.
Therefore this iteration cannot hurt the invariant for any other i': any increase in P_j for j that is not owned by i' does not hurt the inequality while an increase for the j that was owned by i' immediately enqueues i'.

The running time analysis above implies that the algorithm terminates, at which point Q must be empty and thus all bidders must be δ-happy.

■

6.1.1 δ-Happy Bidders

Claim: If all bidders are δ-happy then for every matching M'

$$n\delta + \sum_{i=owner_j} W_{i,j} \geq \sum_{(i,j)\in M'} W_{i,j}$$

Proof: Fix a bidder i and assume i receives item j by this algorithm such that:

$\delta + W_{i,j} - P_j \geq W_{i,j'} - P_{j'}$ where j' is item received in M'

$$\text{Sum over all } i = \sum_{i=owner_j} (\delta + W_{i,j} - P_j) \geq \sum_{i,j'} (W_{i,j'} - P_{j'})$$
$$= n\delta + \sum_{i=owner_j} (W_{i,j} - P_j) \geq \sum_{i,j'} (W_{i,j'} - P_{j'})$$

Since both the algorithm and M' give matchings, each j appears at most once on the left hand side and at most once on the right hand side.
Moreover, if some j does not appear on the left hand side then it was never picked by the algorithm and thus $P_j = 0$. Thus when we subtract $\sum_j P_j$ from both sides of the inequality, the LHS becomes the LHS

of the inequality in the lemma and the RHS becomes at most the RHS of the inequality in the lemma such that

$$= n\delta + \sum_{i=owner_j} W_{i,j} \geq \sum_{i,j'} W_{i,j'}$$

■

Some things to note: In the algorithm, either some bidder gets out of Q or the price of good j goes up by δ. Since all weights (prices) $\in \mathbb{Z}$ (rational prices can be scaled to integers), then $n\delta < 1$.

No P_j can ever increase once its value is above $C = max_{i,j} W_{i,j}$. It follows that the total number of iterations of the main loop is at most $Cn/\delta = O(Cn^2)$ where n is the total number of vertices (goods+bidders). Each loop can be trivially implemented in $O(n)$ time, giving total running time of $O(Cn^3)$, which for the unweighted case, $C = 1$, matches the running time of the basic alternating paths algorithm on dense graphs.

References

[] Noam Nisan, Auction Algorithm for Bipartite Matching, *Turing's Invisible Hand: Computation, Economics, and Game Theory* (July 2009), https://agtb.wordpress.com/2009/07/13/auction-algorithm-for-bipartite-matching/