15-850: Advanced Algorithms **CMU, Spring 2017**

Lecture #9: Maximum Bipartite Matchings

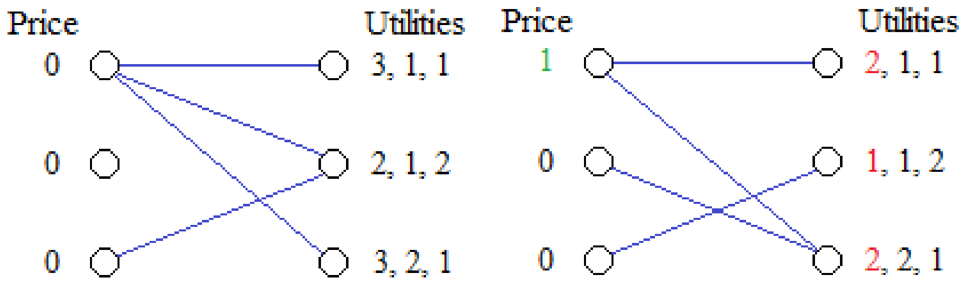
February 13, 2017
Lecturer: Anupam Gupta Scribe: Pranav Devrakonda

We present two different approaches for this problem of maximum bipartite matchings.

1 Buyers and sellers

The setting is n buyers and n items, and buyer b values item i at v_{bi}. The goal is to match each buyer to a distinct item and maximize the sum of the values obtained.

Now suppose that each item i is sold at a price p_i. Then, we can define the utility of item i to buyer b as $v_{bi} - p_i$. Intuitively, utility measures how favorable it is for buyer b to buy item i, since it factors in both the value and the price of the item. We will say that buyer b prefers item i if i gives the highest utility to buyer b, among all items. A buyer can have more than one preferred item, since there can be a tie. Then, we can build a preference graph whose edges are the edges (b, i) such that buyer b prefers item i. Here are two examples of preference graphs, where the second graph results from an increase in price of item 1:



Theorem 9.1 (LP Duality). *If an LP is feasible, then the optimum of the primal LP is equal to the optimum of the dual LP.*

Theorem 9.2. *For all price assignments $p_1, \ldots, p_n$, if the preference graph contains a perfect matching M, then M is a max weight perfect matching.*

Proof. We will formulate the matching problem as a linear program. We will allow "fractional" matching by assigning each edge (b, i) a value x_{bi} in the interval $[0, 1]$:

Primal linear program:

$$\begin{array}{lll} \text{maximize} & \displaystyle\sum_{(b,i)} v_{bi} x_{bi} & \\ \text{subject to} & \displaystyle\sum_{b=1}^{n} x_{bi} = 1 & \forall i \\ & \displaystyle\sum_{i=1}^{n} x_{bi} = 1 & \forall b \\ & x_{bi} \geq 0 & \forall (b,i) \end{array}$$

Dual linear program:

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{i=1}^{n} p_i + \sum_{b=1}^{n} u_b & \\ \text{subject to} & p_i + u_b \geq v_{ib} & \forall (b,i) \end{array}$$

Note that the dual formulation closely resembles prices and utilities: p_i are the prices, and u_b are almost the utilities, with $u_b \geq v_{ib} - p_i$.

Now suppose that the preference graph has a perfect matching M. Let u_b be the utility of the item matched to buyer b, so that $u_b = v_{ib} - p_i$ for each matched (b, i). We know that $v_{ib} - p_i \leq u_b$ is satisfied, because all of the matches are in the utility-maximizing preference graph. Therefore, the matching is a feasible solution to the dual program, with value

$$\sum_i p_i + \sum_b u_b = \sum_i p_i + \sum_{(b,i)\in M} (v_{ib} - p_i) = \sum_{(b,i)\in M} v_{ib}.$$

But note that M is also feasible in the primal program with the same value $\sum_{(b,i)\in M} v_{ib}$! And by linear programming duality, if a value can be attained in both the primal and dual programs, then that value is optimal in both. Thus, M is an optimal matching in the primal program.

$\square$

2 The Hungarian algorithm

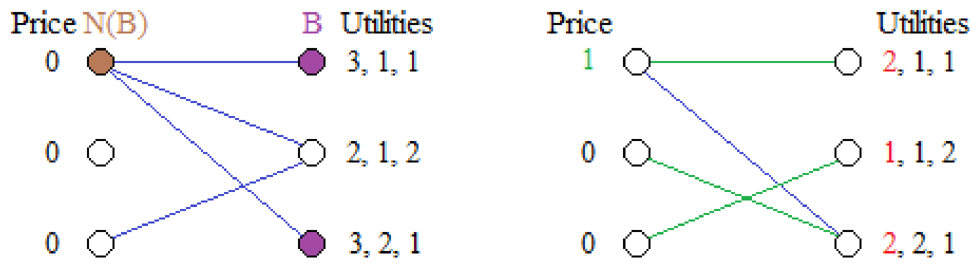
Now we present the Hungarian algorithm, which is based on the buyers-and-sellers setting from the previous section. The algorithm was originally solved by Carl Gustav Jacobi, but it was first published by Harold Kuhn [1], who based his ideas on the works of Jenö Egervary and Dénes König. Munkres showed that the algorithm was in fact implementable in $O(n^3)$ [2]. The algorithm goes as follows:

Initially, all items are sold at price 0.

For each iteration, build the current preference graph. If the graph contains a perfect matching, then return it. The previous theorem ensures that the matching is optimal.

Otherwise, by Hall's theorem, there must be a set B of buyers such that if $N(B)$ is the set of items preferred by at least one buyer in B, then $|N(B)| < |B|$. ($N(B)$ is the *neighborhood* of B in the preference graph.) Intuitively, we have many buyers trying to buy few items, so logically, the sellers of those items should raise their prices! Therefore, the algorithm increases the price of every item in $N(B)$ by 1. (Let's assume that all values in this problem are integral.)

That is the end of the algorithm. Here is the algorithm run on the graph from before:



It is not even clear that the algorithm is guaranteed to terminate, so we will prove termination by a semi-invariant argument. The quantity to keep track of is the value of $\sum_i p_i + \sum_b u_b$, where p_i are the prices and u_b is the maximum utility of buyer b. Note that this value is lower-bounded by the optimum of the dual program. Then, it suffices to prove the following:

Theorem 9.3. *Every time we increase the prices in $N(B)$ by 1, the value of $\sum_i p_i + \sum_b u_b$ decreases by at least 1.*

Proof. The value of $\sum_i p_i$ increases by $|N(B)|$, because we increase the price of each item in $N(B)$ by 1. For each buyer $b \in B$, the value u_b must decrease by 1, since all preferred items had their prices increased by 1, and all other items had utilities at least 1 lower than the original u_b. (Here, we need the fact that all values are integral.) Therefore, the value of $\sum_i p_i + \sum_b u_b$ changes by $|N(B)| - |B| < 0$.

□

Remark 9.4.

- The above is an extension of Vickery auctions with $n-1$ dummy items and one real one with prices set to the second highest price.
- If you don't like the fact that utilities may be negative, then you can just choose S to be the smallest "conflicted" set and raise the prices for $N(S)$. THen, we can ensure that each buyer still has at least preferred item which has nonngegative utility.
- This gives VCG prices!

3 The augmenting path algorithm

Here, we will switch from maximum matchings to minimum matchings, because the latter is easier to explain in this section. Of course, the two problems are equivalent, since we can always negate edges (and, if necessary, shift them so that they become positive again).

A second algorithm to compute minimum matchings uses an augmenting path subroutine. The subroutine, which takes in a matching M and returns one of size $|M|+1$, is presented below. Then, we can start with the empty matching and call this subroutine until we get a maximum matching.

Let the original bipartite graph be G. Construct the **directed** graph G_M as follows: For each edge $e \in M$, insert that edge directed from right to left, with weight $-w_e$. For each edge $e \in G\backslash M$, insert that edge directed from left to right, with weight w_e. Then, compute the shortest path P that starts from the left and ends on the right, and return $M \triangle P$. It is easy to see that $M \triangle P$ is a matching of size $|M|+1$, and has total weight equal to the sum of the weights of M and P.

Call a matching M an <u>extreme</u> matching if M has minimum weight among all matchings of size $|M|$. The main idea is to show that the above subroutine preserves extremity, so that the final matching must be extreme and therefore optimal.

Theorem 9.5. *If M is an extreme matching, then so is $M \triangle P$.*

Proof. Suppose that M is extreme. We will show that there exists an augmenting path P such that $M \triangle P$ is extreme. Then, since the algorithm finds the shortest augmenting path, it will find a path that is no longer than P, so the returned matching must also be extreme.

Consider an extreme matching M' of size $|M|+1$. Then, the edges in $M \triangle M'$ are composed of disjoint paths and cycles. Since $M \triangle M'$ has more edges in M' than edges in M, there is some path $P \subset M \triangle M'$ with one more edge in M' than in M. This path necessarily starts and ends on opposite sides, so we can direct it to start from the left and end on the right. We know that $|M' \cap P| = |M \cap P| + 1$, which means that $M\backslash P$ and $M'\backslash P$ must have equal size. The total weight of $M\backslash P$ and $M'\backslash P$ must be the same, since otherwise, we can swap the two matchings and improve one of M and M'. Therefore, $M \triangle P = (M' \cap P) \cup (M\backslash P)$ has the same weight as M' and is extreme. □

Note that the formulation of G_M is exactly the graph constructed if we represent the minimum matching problem as a min-cost flow. Indeed, the previous theorem can be generalized to a very similar statement for the augmenting path algorithm for min-cost flows.

References

[1] Harold W Kuhn. The hungarian method for the assignment problem. In *50 Years of Integer Programming 1958-2008*, pages 29–47. Springer, 2010. 2

[2] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2002. 2