Optimization **Summer 2010**

Lecture 9 — May 10

Lecturer: Julián Mestre

9.1 The assignment problem

The input of the maximum assignment problem is a complete bipartite graph (U, V, E) with n vertices on each side of the bipartition and a cost function $c : E \to \mathcal{R}_+$. An *assignment* is a one-to-one mapping $\sigma : V \to U$ and its cost is defined as

$$\text{cost}(\sigma) = \sum_{j \in V} c(\sigma(j), j).$$

The goal is to produce an assignment with maximum cost. In this lecture we will show a combinatorial algorithm for this problem. Our algorithm is based on the following integer linear program formulation:

$$\begin{array}{lll} \text{maximize} & \displaystyle\sum_{\substack{i \in U \\ j \in V}} c(i,j)\, x_{ij} & \\ \text{subject to} & \displaystyle\sum_{j \in V} x_{ij} = 1 & \forall\, i \in U \\ & \displaystyle\sum_{i \in U} x_{ij} = 1 & \forall\, j \in V \\ & x_{ij} \in \{0,1\} & \forall\, i \in U, j \in V \end{array}$$

In this formulation setting $x_{ij} = 1$ corresponds to $\sigma(j) = i$. Now suppose we relax the integrality constraints; that is, we replace $x_{ij} \in \{0,1\}$ with $x_{ij} \geq 0$ (why not $0 \leq x_{ij} \leq 1$?). Taking the dual of the resulting LP we get:

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{i \in U} r_i + \sum_{j \in V} p_j & \\ \text{subject to} & r_i + p_j \geq c(i,j) & \forall\, i \in U, j \in V \\ & r_i, p_j \text{ free} & \forall\, i \in U, j \in V \end{array}$$

Clearly the value of any feasible dual solution is an upper bound on the value of a optimal fractional solution for the primal, which in turn is an upper bound on the value of an optimal integral solution. We will develop an algorithm that produces an assignment and a feasible dual solution with similar cost. This will allow us to argue that our assignment is nearly optimal.

9.2 Maximum assignment via auctions

Our algorithm will regard one side of the bipartition as a set of *bidders* (U), and the other side as a set of objects (V) that the bidders would like to get. For each bidder-object pair

$(i,j) \in E$ we interpret c_{ij} as i's valuation of j. Therefore, under this new interpretation, our goal is to come up with an assignment maximizing the overall happiness of the bidders.

The algorithm will maintain a *partial assignment* σ where some objects $j \in V$ may not be mapped; we denote this situation by $\sigma(j) = \perp$. We will also keep dual variables p_j for each $j \in V$, which we interpret as prices, and dual variables r_i for each $i \in U$.

Algorithm 1 AUCTION-MECHANISM(U, V, E, w, δ)

```
 1. For each object $j \in V$, set $p_j \leftarrow 0$ and $\sigma(j) = \perp$
 2. $Q \leftarrow$ a set containing all the bidders $U$
 3. while $Q \neq \emptyset$ do
 4.    $i \leftarrow$ some bidder from $Q$
 5.    $j \leftarrow$ an object maximizing $c(i,j) - p_j$
 6.    if $\sigma(j) \neq \perp$ then
 7.       add $\sigma(j)$ to $Q$
 8.    $\sigma(j) \leftarrow i$
 9.    $r_i \leftarrow c(i,j) - p_j$
10.    $p_j \leftarrow p_j + \delta$
11. return $\sigma$
```

Notice that the algorithm is parametrized by δ. We will do the analysis for a generic δ and then choose a suitable value to get the desired result.

Lemma 9.1. *The algorithm always terminates. In fact, the number of iterations is at most* $n\left(\frac{c_{max}}{\delta} + 1\right)$, *where* $c_{max} = \max\limits_{(i,j)\in E} c(i,j)$.

Lemma 9.2. *At any point during the execution of the algorithm, if bidder* $i \notin Q$ *then*

i) *i is assigned some object j (that is, $\sigma(j) = i$) such that $r_i + p_j = c(i,j) + \delta$, and*

ii) $r_i + p_{j'} \geq c(i,j')$ *for all* $j' \in V$.

In particular, at the end, the algorithm has a feasible dual solution $(\mathbf{r}, \mathbf{p})$ and a nearly optimal assignment σ because

$$\sum_{i\in U} r_i + \sum_{j \in V} p_j = \text{cost}(\sigma) + n\,\delta.$$

Theorem 9.3. *If all the edge cost are integral then running* AUCTION-MATCHING *with* $\delta = \frac{1}{n+1}$ *solves the maximum assignment problem in* $O\left(n^3 c_{max}\right)$ *time.*

Proof: From Lemma 9.1 and our choice of δ we know that the algorithm runs for at most $O(n^2 c_{max})$ iterations. Each iteration can be implemented in $O(n)$ time, so the claimed running time follows.

Since all edge costs are integral, any two assignment either have the same cost or the differ by at least 1 unit. From our choice of δ we know that the cost of any other assignment τ is no larger than the one output by the algorithm because the cost of τ cannot be larger than the cost of the dual solution. $\square$