Optimization I **Summer 2011**
Lecture 11

The Assignment Problem and Primal-Dual Algorithms

1 Assignment Problem

Suppose we want to solve the following problem: We are given

- a set of people I, and a set of jobs J, with $|I| = |J| = n$
- a cost $c_{ij} \geq 0$ for assigning job j to person i.

We would like to assign jobs to people, so that each job is assigned to one person and each person is assigned one job, and we minimize the cost of the assignment. This is called the **assignment problem**.

Example input:

	Jobs			
People	90	75	75	80
	35	85	55	65
	125	95	90	105
	45	110	95	115

The assignment problem is related to another problem, the maximum cardinality bipartite matching problem. In the **maximum cardinality bipartite matching problem**, you are given a bipartite graph $G = (V, E)$, and you want to find a matching, i.e., a subset of the edges F such that each node is incident on at most one edge in F, that maximizes $|F|$.

Suppose you have an algorithm for finding a maximum cardinality bipartite matching. If all of the costs in the assignment problem were either 0 or 1, then we could solve the assignment problem using the algorithm for the maximum cardinality bipartite matching problem: We construct a bipartite graph which has a node for every person i and every job j, and an edge from i to j if $c_{ij} = 0$. A matching in this graph of size k corresponds to a solution to the assignment problem of cost at most $n - k$ and vice versa.

Can we use this algorithm also for solving the assignment problem if the costs are arbitrary?

We'll assume that the algorithm you have for computing a maximum cardinality matching gives, in addition to the matching, a **vertex cover** of the same size as the maximum matching. A **vertex cover** of $G = (V, E)$ is a subset of the vertices W such that each edge in E is incident on at least one vertex in W.

You can think of the vertex cover as a **certificate of optimality** of the matching: Let W be a vertex cover, and suppose $|W| = k$. All edges have an endpoint in W and a matching can have at most one edge incident on each vertex. Hence a matching can have at most k edges.

So if the output of the algorithm is a vertex cover and a matching of the same size, then we know the matching must be optimal. By König's theorem, such a pair always exists in a bipartite graph:

Theorem 1. (König) *In a bipartite graph, the maximum size of a matching is equal to the minimum size of a vertex cover.*

(You may be reminded of weak and strong duality, and you would be absolutely right. You can demonstrate a primal and dual LP that correspond to the problem of finding a maximum matching and a minimum vertex cover in a bipartite graph.)

Observation 1. *Consider an assignment problem with cost matrix C.*

- *If we subtract the same amount from each entry in a row of C, we will not change the optimal solution.*
- *If we subtract the same amount from each entry in a column of C, we will not change the optimal solution.*

Observation 2. *Consider an assignment problem with cost matrix C. If $C \geq 0$, and there exists an assignment which only assigns i to j if $c_{ij} = 0$, then this assigment is optimal.*

These two observations give us an idea for an algorithm: we subtract the minimum element from each row and each column and obtain a new cost matrix $\bar{C}$. By the first observation, the optimal assignment for $\bar{C}$ is the same as the optimal assignment for C.

Now, we use the maximum cardinality matching algorithm to check if there is a perfect matching that uses only edges of cost 0. If such a perfect matching exists, then it corresponds to an assignment of cost 0 for $\bar{C}$, which is optimal by the second observation.

90	75	75	80
35	85	55	65
125	95	90	105
45	110	95	115

$\rightarrow$

15	0	0	5
0	50	20	30
35	5	0	15
0	65	50	70

$\rightarrow$

15	0	0	0
0	50	20	25
35	5	0	10
0	65	50	65

In the example, the matrix $\bar{C}$ does not have a perfect matching of edges of cost 0. What to do next? We would like to create more 0's, but we can no longer subtract from individual rows or columns, without creating negative entries.

Recall that the maximum cardinality bipartite matching algorithm will also produce a vertex cover of the same size as the matching. Hence, if the maximum cardinality matching has size $k < n$, we will obtain a vertex cover of size k. In the assignment problem, the nodes in the vertex cover correspond to rows and columns of the matrix. So a vertex cover in the bipartite matching instance corresponds to a subset of the rows and columns of the matrix, say $I' \subset I, J' \subset J$, such that if $\bar{c}_{ij} = 0$ then $i \in I'$ or $j \in J'$.

Suppose we subtract some value $\alpha > 0$ from every row that is *not* in I', and we add α to every column in J'. Then, the only entries that get decreased are the entries (i, j) where $i \notin I', j \notin J'$. So, we let $\alpha = \min_{(i,j):i \notin I', j \notin J'} \bar{c}_{ij}$.

We know that $\alpha > 0$, and that by subtracting α from every row that is not in I' and adding α to every column in J', we maintain that all entries are non-negative. In our example, the rows and columns in $I' \cup J'$ are indicated with a $*$.

	*		*	
*	15	0	0	0
	0	50	20	25
	35	5	0	10
	0	65	50	65

$\rightarrow$

20	0	5	0
0	45	20	20
35	0	0	5
0	60	50	60

We also know that we create an additional 0, but unfortunately, we may also remove some 0's. So how do we know we are making progress?

Lemma 2. *"The total amount subtracted from the entries in the matrix strictly increases."*

Proof. We subtract α from $n(n - |I'|)$ entries, and we add α to $n(|J'|)$ entries. Hence in total we subtracted $\alpha n(n - |I'| - |J'|)$. Now, note that $|I'| + |J'|$ is equal to the size of the vertex cover, which is strictly less than n. □

If the entries in the original matrix are integer, then they remain integer in the course of these operations. So we know our algorithm is finite if the matrix is integer (and hence also if it is rational). It is possible to analyze this algorithm a lot better (using knowledge of the maximum cardinality bipartite matching algorithm) and show that it runs in $O(n^3)$ time.

The algorithm we derived is called the Hungarian algorithm. It was first proposed by Kuhn in 1955, and it was shown to run in polynomial time by Munkes in 1957.

2 Connection to Duality

The following linear program gives a lower bound on the optimal value of the assignment problem:

$$\begin{aligned} \min \quad & \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j \in J} x_{ij} = 1 \text{ for all } i \in I \\ & \sum_{i \in I} x_{ij} = 1 \text{ for all } j \in J \\ & x_{ij} \geq 0 \end{aligned}$$

To see this, note that we can let $x_{ij} = 1$ if i is assigned to j and 0 otherwise. Clearly, this is a feasible solution to the LP, so the optimal value of the LP must be at most the optimal value of the assignment problem.

We consider the dual of the LP:

$$\begin{aligned} \max \quad & \sum_{i \in I} u_i + \sum_{j \in J} v_j \\ \text{s.t.} \quad & u_i + v_j \leq c_{ij} \text{ for all } i \in I, j \in J \end{aligned}$$

Now, we know that x is an optimal solution to the primal LP and u, v is an optimal solution to the dual LP if and only if (i) x is feasible for the primal LP, (ii) u, v is feasible for the dual LP, (iii) the primal and dual solutions obey complementary slackness.

Thinking about what these conditions mean for the assignment problem allows us to formulate the Hungarian algorithm in a much more general way:

1. **We maintain a feasible dual solution.** We let u_i be the amount subtracted from row i and v_j is the amount subtracted from column j. Feasibility means that we must ensure that $\bar{c}_{ij} = c_{ij} - u_i - v_j$ is non-negative for all (i, j).

2. **We try to find a primal solution x that satisfies complementary slackness with respect to the current dual solution.** Complementary slackness for the assignment problem means that we try to find an assignment that only uses edges with $\bar{c}_{ij} = 0$, i.e., we solve the maximum cardinality bipartite matching problem on the graph that contains a node for every $i \in I, j \in J$ and an edge (i, j) if $\bar{c}_{ij} = 0$. We either find a perfect matching, or we get a vertex cover of size $< n$.

3. **If we cannot find such a primal solution, we find a *direction of dual increase*.** The vertex cover in the bipartite matching instance corresponds to I', J', a subset of the rows and columns, such that $|I'| + |J'| < n$ and if $\bar{c}_{ij} = 0$ then $i \in I'$ or $j \in J'$.

   We let $\alpha = \min_{(i,j): i \notin I', j \notin J'} \bar{c}_{ij}$, and we update

   $$u_i \leftarrow u_i - \alpha \text{ for all } i \notin I', \quad v_j \leftarrow v_j + \alpha \text{ for all } j \in J'.$$

   The new dual is feasible, by our choice of α. Lemma 2 says that the dual objective strictly increases, and we used this to derive the fact that the algorithm is finite.

This is what is known as a primal-dual algorithm, and many combinatorial algorithms can be interpreted in this framework, for example, Dijkstra's algorithm and Ford-Fulkerson's maximum flow algorithm. Later in this course, we will see how to use these ideas also to develop *approximation algorithms* for NP-hard problems.