6 The Optimal Assignment Problem

6.1 Definition

The *complete bipartite graph* $K_{m,n}$ is the bipartite graph with bipartition $\{X, Y\}$ where $|X| = m$, $|Y| = n$ and each vertex of X is adjacent to every vertex of Y.

6.2 Problem

Let N be a network obtained from $K_{m,m}$ by giving each edge e an integer weight $w(e)$. Find a perfect matching of maximum weight in N.

As in most optimization problems, an important step in finding an algorithm for solving this problem is to give a criterion for recognising an optimal solution. We shall accomplish this by giving a 'max-min formula' using the following concept.

6.3 Definition

A *feasible vertex labelling* for N is a function $\ell : V(N) \to \mathbb{Z}$ such that $\ell(x) + \ell(y) \geq w(xy)$ for all $x \in X$ and $y \in Y$. We define the *size* of ℓ, by $size(\ell) = \sum_{v \in V(N)} \ell(v)$.

6.4 Example

Let N be the weighted $K_{5,5}$ with bipartition $X = \{x_1, x_2, x_3, x_4, x_5\}$ and $Y = \{y_1, y_2, y_3, y_4, y_5\}$, and weights given by the following matrix.

	y_1	y_2	y_3	y_4	y_5
x_1	3	5	5	4	1
x_2	2	2	0	2	2
x_3	2	4	4	1	0
x_4	0	1	1	0	0
x_5	3	2	1	3	3

Thus, for example, the weight of the edge x_1y_1 is $w(x_1y_1) = 3$, and the weight of the matching $M = \{x_1y_1, x_2y_2, x_3y_3, x_4y_4, x_5y_5\}$ is given by $w(M) = 3 + 2 + 4 + 0 + 3 = 12$. We may define a feasible vertex labelling ℓ of N by putting $\ell(x_i)$ equal to the maximum weight of an edge incident to x_i, and $\ell(y_i)$ equal to zero for all $1 \leq i \leq 5$. This gives $\ell(x_1) = 5$, $\ell(x_2) = 2$, $\ell(x_3) = 4$, $\ell(x_4) = 1$, $\ell(x_5) = 3$ and $\ell(y_i) = 0$ for all $1 \leq i \leq 5$. Thus $size(\ell) = 5 + 2 + 4 + 1 + 3 = 15$.

6.5 Lemma

Let ℓ be a feasible vertex labelling for N and M be a perfect matching in N. Then $w(M) \leq size(\ell)$.

Proof Let $M = \{x_1y_1, x_2y_2, \ldots, x_my_m\}$. Then

$$w(M) = \sum_{i=1}^{m} w(x_iy_i) \leq \sum_{i=1}^{m} [(\ell(x_i) + \ell(y_i)] = \sum_{v \in V(N)} \ell(v) = size(\ell)$$

since ℓ is a feasible vertex labelling.

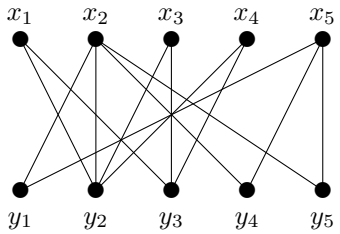
Lemma 6.5 implies that the maximum weight of a perfect matching in N is less than or equal to the minimum size of a feasible vertex labelling of N. We shall see that equality always occurs.

6.6 Definition

Let ℓ be a feasible vertex labelling of N. Then the *equality subgraph* $G(\ell)$ for ℓ in N is the spanning subgraph of N containing all edges xy for which $\ell(x) + \ell(y) = w(xy)$.

6.7 Example

The equality subgraph $G(\ell)$ for the feasible vertex labelling given in Example 6.4 is shown below.



Note that this bipartite graph is the same as the one considered in the previous chapter as Example 5.2.6

6.8 Lemma

Let ℓ be a feasible vertex labelling for N and M be a perfect matching in in the equality subgraph $G(\ell)$. Then $w(M) = size(\ell)$ and hence M is a maximum weight perfect matching in N and ℓ is a minimum size feasible vertex labelling of N.

Proof Let $M = \{x_1y_1, x_2y_2, \ldots, x_my_m\}$. Since $G(\ell)$ is the equality subgraph of ℓ in N, we have $\ell(x_i) + \ell(y_i) = w(x_iy_i)$ for all $1 \leq i \leq m$. Thus

$$w(M) = \sum_{i=1}^{m} w(x_iy_i) = \sum_{i=1}^{m} ((\ell(x_i) + \ell(y_i)) = \sum_{v \in V(N)} \ell(v) = size(\ell).$$

The facts that M is a maximum weight perfect matching in N and ℓ is a minimum size feasible vertex labelling of N now follows from Lemma 6.5.

6.9 Theorem (Egerváry, 1931)

Let N be a weighted complete bipartite graph. Then the maximum weight of a perfect matching in N is equal to the minimum size of a feasible vertex labelling of N.

Proof Let ℓ be a minimum size feasible vertex labelling of N and $G = G(\ell)$ be the equality subgraph for ℓ in N. By Lemma 6.8 it suffices to show that G has a perfect matching. We proceed by contradiction.

Suppose that G does not have a perfect matching. Then by Hall's Theorem, there exists a set $S \subseteq X$ such that $|\Gamma_G(S)| < |S|$. Let

$$\alpha = \min\{\ell(x) + \ell(y) - w(xy) \ : \ x \in S, y \in Y - \Gamma_G(S)\}.$$

Note that $\alpha > 0$ since there are no edges in the the equality subgraph from S to $Y - \Gamma_G(S)$ and hence we have $\ell(x) + \ell(y) > w(xy)$ for all $x \in S$ and $y \in Y - \Gamma_G(S)$. We may now define a feasible vertex labelling ℓ' of N as follows. For each $v \in V(N)$ let

$$\ell'(v) = \begin{cases} \ell(v) - \alpha & \text{for } v \in S \\ \ell(v) + \alpha & \text{for } v \in \Gamma_G(S) \\ \ell(v) & \text{otherwise.} \end{cases}$$

We shall show that ℓ' is a feasible vertex labelling of N. Suppose not. Then we have $\ell'(x)+\ell'(y) < w(xy)$ for some $x \in X$ and $y \in Y$. Since ℓ is a feasible vertex labelling of N, we must have $x \in S$ and $y \in Y - \Gamma_G(S)$. But then the definition of α implies that $\ell(x) + \ell(y) - w(xy) \geq \alpha$ and hence $\ell'(x) + \ell'(y) - w(xy) \geq 0$. Thus ℓ' is a feasible vertex labelling of N.

Since $\alpha > 0$ and $|S| > |\Gamma_G(S)|$, we have $size(\ell') = size(\ell) - \alpha(|S| - |\Gamma_G(S)|) < size(\ell)$. This contradicts the fact that ℓ is a minimum size feasible vertex labelling of N. Thus G has a perfect matching.

6.10 The Hungarian method

Kuhn gave the following algorithm for solving the optimal assignment problem in 1954. He called it the Hungarian method since it was inspired by Egerváry's proof of Theorem 6.9. Suppose N is a network obtained from $K_{m,m}$ by giving each edge e an integer weight $w(e)$. The algorithm iteratively constructs a sequence of feasible vertex labelling $\ell_1, \ell_2, \ldots$ for N such that $size(\ell_{i+1}) < size(\ell_i)$, and a sequence of matchings M_i such that M_i is a maximum matching in the equality subgraph $G(\ell_i)$, for all $i \geq 1$. It stops when it finds a feasible vertex labelling ℓ_i for which M_i is a perfect matching in $G(\ell_i)$.

Initial Step Construct a feasible vertex labelling ℓ_1 for N by putting $\ell_1(x) = \max\{w(xy) : y \in Y\}$ for each $x \in X$, and $\ell_1(y) = 0$ for all $y \in Y$. Construct a maximum matching M_1 in $G(\ell_1)$ using Algorithm 5.2.5.

Iterative Step Suppose we have constructed a feasible vertex labelling ℓ_i of N, and a maximum matching M_i in $G = G(\ell_i)$, for some $i \geq 1$.

- If $|M_i| < m$, then construct a new feasible vertex labelling ℓ_{i+1} for N as follows:

  (1) Let F be a maximal M_i-alternating forest in G rooted at the set of M_i-unsaturated vertices in X. Put $S = V(F) \cap X$. Then $\Gamma_G(S) = V(F) \cap Y$.

  (2) Compute $\alpha = \min\{\ell_i(x) + \ell_i(y) - w(xy) : x \in S, y \in Y - \Gamma_G(S)\}$.

  (3) For each $v \in V(N)$ let

$$\ell_{i+1}(v) = \begin{cases} \ell_i(v) - \alpha & \text{for } v \in S \\ \ell_i(v) + \alpha & \text{for } v \in \Gamma_G(S) \\ \ell_i(v) & \text{otherwise.} \end{cases}$$

  Construct a maximum matching M_{i+1} in $G(\ell_{i+1})$ using Algorithm 5.2.5, starting with the matching M_i. Now iterate.

- If $|M_i| = m$ then STOP. Output M_i and ℓ_i.

6.11 Notes

(a) The fact that each labelling ℓ_i constructed by Algorithm 6.10 is a feasible vertex labelling of N follows by induction on i, using a similar argument as in the proof of Theorem 6.9.

(b) The fact that $size(\ell_{i+1}) < size(\ell_i)$ in each iteration of Algorithm 6.10 also follows by a similar argument as in the proof of Theorem 6.9.

(c) The fact that M_i is contained in $G(\ell_{i+1})$ follows from the definition of ℓ_{i+1}.
(d) The algorithm must terminate since each iteration decreases the size of the feasible vertex labelling, and this size is bounded below by the weight of any perfect matching of N.
(e) When the algorithm terminates it outputs a feasible vertex labelling l_i and a perfect matching M_i in the equality subgraph $G(\ell_i)$. Lemma 6.8 implies that $w(M_i) = size(\ell_i)$, and hence M_i is a maximum weight matching in N and ℓ_i is a feasible vertex labelling of minimum size.

6.12 Example

Let N be the weighted $K_{5,5}$ given in Example 6.4. with weights given by the following matrix.

First iteration

We first construct the feasible vertex labelling ℓ_1 below.

	y_1	y_2	y_3	y_4	y_5	
x_1	3	5	5	4	1	5
x_2	2	2	0	2	2	2
x_3	2	4	4	1	0	4
x_4	0	1	1	0	0	1
x_5	3	2	1	3	3	3
	0	0	0	0	0	ℓ_1

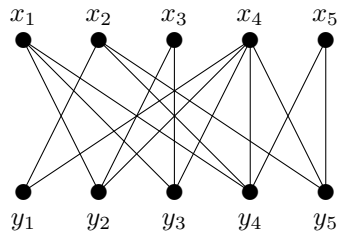
The equality subgraph $G(\ell_1)$ for ℓ_1 is shown in Example 6.7. Applying Algorithm 5.2.5 to $G = G(\ell_1)$ we construct the maximum matching $M_1 = \{x_2y_1, x_3y_2, x_4y_3, x_5y_4\}$ in $G(\ell_1)$, see Example 5.2.6.

Second iteration

The maximal M_1-alternating forest F in G rooted at the set of M_1-unsaturated vertices in X, $\{x_1\}$, is also given in Example 5.2.6. We have $S = V(F) \cap X = \{x_1, x_3, x_4\}$ and $\Gamma_{G(\ell_1)}(S) = V(F) \cap Y = \{y_2, y_3\}$. Hence $\alpha = 1$, and we construct a new feasible vertex labelling ℓ_2 for N given below.

	y_1	y_2	y_3	y_4	y_5		
x_1	3	5	5	4	1	5	4
x_2	2	2	0	2	2	2	2
x_3	2	4	4	1	0	4	3
x_4	0	1	1	0	0	1	0
x_5	3	2	1	3	3	3	3
	0	0	0	0	0	ℓ_1	
	0	1	1	0	0		ℓ_2

The equality subgraph $G(\ell_2)$ as shown below.



We apply Algorithm 5.2.5 to $G(\ell_2)$ starting with the matching M_1 and construct the perfect matching $M_2 = \{x_4y_2, x_1y_4, x_2y_1, x_3y_3, x_5y_5\}$ in $G(\ell_2)$. We have

$w(M_2) = 14 = size(\ell_2)$. Thus M_2 is a maximum weight perfect matching in N, and ℓ_2 is a minimum size feasible vertex labelling for N.

We close this chapter by showing that the Hungarian Method is a strongly polynomial algorithm. We first need a result which tells us that the number of times the algorithm grows an alternating forest cannot be too large.

6.13 Lemma

Let N is a network obtained from $K_{m,m}$ by giving each edge e an integer weight. Suppose we use the Hungarian method to construct a maximal weight perfect matching in N. Then the number of times the method grows an alternating forest is at most $2m^2$.

Proof Suppose that at some point in the algorithm we have a matching M in an equality subgraph $G(\ell)$ and we have grown an M-alternating forest F in $G(\ell)$ rooted at the set of M-unsaturated vertices in X. The are two possible alternatives: either F contains an M-augmenting path and we use it to construct a matching M' with $|M'| > |M|$; or we deduce that M is a maximum matching in $G(\ell)$ and we use the set $S = V(F) \cap X$ to construct a new feasible vertex labelling ℓ'. If the second alternative occurs, we grow an M-alternating forest F' in $G(\ell')$ rooted at the set of M-unsaturated vertices in X. It can be shown that F will be properly contained in F'.

It follows that each time we grow an alternating forest, we either construct a bigger matching, or the matching stays the same and we construct a new feasible vertex labelling which gives rise to a bigger alternating forest in the new equality subgraph. Since the maximum number of vertices in an alternating forest is $2m$, the size of the matching must increase after we grow at most $2m$ alternating forests. Since the algorithm terminates when it finds a matching with m edges, the number of edges in a matching can increase at most m times. Thus the total number of alternating forests grown during the algorithm is at most $2m^2$.

6.14 Theorem

Suppose N is a network obtained from $K_{m,m}$ by giving each edge e an integer weight. Then the Hungarian method finds a maximum weight perfect matching in N in time $O(m^4)$, under the assumption that all elementary arithmetic operations take constant time.

Proof The algorithm proceeds by growing alternating forests. Growing an M-alternating forest in an equality subgraph G by breadth first search takes $O(|E(G(\ell))| = O(m^2)$ time. Once we have grown the forest, we either update the matching, which takes $O(m)$ time, or update the feasible vertex labelling, which again takes $O(m)$ time. Thus the total time spent on each alternating forest is $O(m^2 + m) = O(m^2)$. Since the algorithm grows at most $2m^2$ alternating forests by Lemma 6.13, the total time taken by the algorithm is $O(2m^4) = O(m^4)$.

6.15 Remark

The *complete graph* K_n is the simple graph with n vertices in which all pairs of vertices are adjacent. In 1965, J. Edmonds gave an algorithm for constructing a maximum weight perfect matching in a network obtained by assigning an integer weight to the edges of a complete graph on an even number of vertices. This algorithm uses his algorithm for finding a maximum matching in a graph as a subroutine, see Remark 5.1.13.