# The Hungarian Method for the Optimum Bipartite Matching Problem, And Optimum Network Flows.

For convenience we assume that bipartite graph K is complete –
that is, for each girl node u and each boy node v there is an edge (u,v) in K.
We also assume that the set of girl nodes is
not smaller than the set of boy nodes.

For each edge j of K, let c(j) be the amount of love
which the boy of edge j has for the girl of edge j.
We want to find a matching M in K
which maximizes, over all possible matchings,
the total amount, $c(M) = \Sigma\{c(j): j \in M\}$ , of love consummated by M,
and of course we want a way to prove that M is optimum whenever it is.

That is, we want a 'good characterization'
of matchings, M, which maximize c(M).
That is, we want an EP theorem of the form:
For any matching M in K,
either there is a matching M' such that $c(M) < c(M')$
or there is some easy certification that no such M' exists.

**Egervary's Theorem**, 1931, is the theorem hoped for:
For bipartite graph K , and any weighting c = [c(j): j ε E(K)], of its edges, and any matching M in K,
either there is a matching M' in K such that c(M) < c(M')
or there is a weighting y = [y(i) ≥ 0: i ε V(K)] of the nodes of K
such that for each edge j = (u,v) of K, y(u) + y(v) ≥ c(j),
and c(M) ≡ ∑ [c(j): j∈M] = y(V) ≡ ∑[y(i): i∈V]. Not both.

**Egervary's Theorem**, stated in a more usual way:
For any complete bipartite graph, K = (V, E),
(with no more boy-nodes than girl-nodes),
and any weighting c = [c(j): j ε E], of its edges,
the maximum of c(M) ≡ ∑ [c(j): j∈M] where M is a matching in K equals
the minimum of y(V) ≡ ∑[y(i): i∈V] where y≥0,
and ∀ j=(u,v) ∈ E, y(u) + y(v) ≥ c(j).

Here is another equivalent form
which uses the "complimentary slackness principle":

**Egervary's Theorem.** For the same input as above,
there is a matching, M, and a node-weighting, y, such that
y≥0, and ∀ j=(u,v) ∈ E, y(u) + y(v) ≥ c(j);
for each y(i) > 0, there is an edge j in M which includes node i ;
for each edge j = (u,v) ∈ M, y(u) + y(v) = c(j).

The lp problem, max [cx: Ax ≤ b, and x≥0], is a 'relaxation'
where b is all 1's, where A is the 'incidence matrix' of graph K,
A(i,j) = 1 when i is a node of edge j, and =0 otherwise.
The dual lp problem is min[yb: yA ≥ c, and y≥0].

Complimentary slackness: y(b – Ax) + (yA – c)x ≥ 0,
And hence y(b – Ax) + (yA – c)x = 0 iff, for each term,
one of the factors is 0.

Here is the Hungarian Algorithm for proving Egervary's Theorem:

Assign any large enough non-neg y(i)'s to the nodes
so that $y(u) + y(v) \geq c(j)$.
Let E(=) denote the subset of edges such that $y(u) + y(v) = c(j)$.
We consider only matchings $M \subsetneq E(=)$.

The complimentary slackness conditions imply that
we have an optimum y and an optimum matching M,
when M is contained in E(=) and hits every node i such that $y(i) > 0$.

If there is a node r not met by the current M and where $y(r) > 0$,
we grow an alternating tree T from node r using only edges of E(=)
to either change M or change y.

In the case that we change y,
E(=) changes but continues to contain the edges of M and T.
We continue to grow the same T until
either M is replaced by another matching contained in E(=)
or until an additional y(i) becomes 0,
at which time we are optimum or we start a new T.

(1.3.0) Optimum Network Flows.

Given a digraph G = (V,E); a “source node”, r∈V;
a “sink node”, s∈V; and a cost c(j) for each edge j∈E;
and an integer, k≥0.

**A simple version of the “optimum network flow problem”** is to find, if the exist, k edge-disjoint directed paths from r to s, such the sum of the costs of the edges used is minimum.

The problem is not very different from the optimum love-making problem (bipartite matching) for which we learned the Hungarian method. If fact each of these problems can be translated into the other. However the form of network flows is especially beautiful and useful.

(1.3.1) Feasibility: **The max flow – min cut theorem:**
There exists k edge-disjoint directed paths from r to s in G
or there exists a cut of size (k-1) separating s from r.
(Obviously not both.)

In other words, the max number of edge-disjoint paths in G from r to s equals the minimum size of a cut separating s from r.

Let d(G,S), for a proper subset S of V, mean the set of edges j of G such that tail t(j)∉S and head h(j)∈S. In other words d(G,S) means the set of edges of G which ‘enter’ S. **A cut separating t from r** means the set d(G,S) for some r∉S, s∈S.

Algorithmic proof: Start with all the edges of G colored blue and directed as given. At the general step of the algorithm find, if possible, a directed path p from r to s. Color red the blue edges of p, color blue the red edges of p, and reverse the direction of all edges of p, to get the new current digraph G'.
Repeat until there is no p in the current G'.

Then the reason is that there is a cut d(G',S) separating s from r such that there are no blue or red edges in d(G',S).
Hence, for this S, d(G,S) is the red edges of d(G',(V-S)) reversed and restored to blue.

Let F ⊆ E be the red edges of the final G', recolored blue and properly directed. Clearly |d(G,S)| = |d(G,{s})|,
and clearly for each node i∈V different from r and s,
the number of edges of F directed into i
equals the number of edges of F directed out of i.

It is then easy to decompose F into |d(G,S)| directed paths from r to s. That's mostly it. □

To find a cheapest set F of edges: find each path p as a shortest path relative to “effective costs” in the current G’.

For each edge, j, calculate effective costs in the next G’ as the effective cost in the current G’ minus the min current path cost to the head of j minus the min current path cost to the tail of j.

See the Edmonds-Karp reprint for more careful details.