The Hungarian Method

Jake R. Gameroff, Jonathan Campana

March 15, 2024

This is the augmented transcript of a lecture on the Hungarian method given by Professor Luc Devroye on April 8, 2024 for Computer Science 252, Honours Algorithms and Data Structures.

The Assignment Problem

Our general objective is to develop an $O(n^3)$ algorithm which solves the $n \times n$ **assignment problem**.

We are provided as input a set $W = \{1, 2, \ldots, n\}$ of workers and another set $J = \{1, 2, \ldots, n\}$ of available jobs. We also have a cost function $C : W \times J \to \mathbb{R}_{\geq 0}$, where for $i, j \in \{1, 2, \ldots, n\}$, the value $C(i, j)$ is the cost of giving the job j to worker i.

Hence, we wish to minimize the cost of hiring each worker. Formally, we wish to find a bijection $f : W \to J$ such that $\sum_{w \in W} C(w, f(w))$ is minimal.

	1	2	3	4	5
1	3	8	1	9	6
2	2	11	4	15	3
3	7	2	8	8	10
4	4	10	12	7	8
5	5	6	6	11	9

Figure 1: Example of a minimal weight matching in a 5×5 matrix.

We may compactly represent this problem with an $n \times n$ matrix M, where $M[i, j] \geq 0$ represents the cost of giving job i to worker j. Formulated in this way, we seek a permutation $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ such that

$$\sum_{i=1}^{n} M[i, \sigma_i]$$

is minimal. The pairs (i, σ_i) then form a **minimal weight matching**.

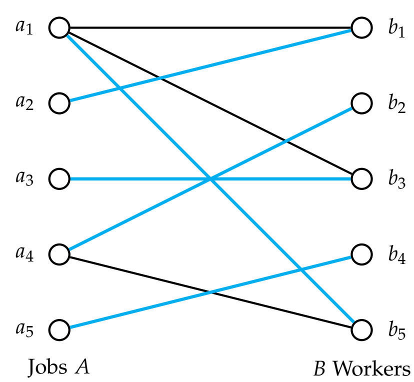

Figure 2: A perfect matching (cyan edges) in a graph.

Perfect Matchings

Definition 1. A **matching** in a graph G is a subset $M \subseteq E(G)$ of edges such that every vertex in G is incident to *at most* one edge in M. A matching M is called a **perfect matching** if every vertex in G is incident to *exactly* one edge in M.

A special case of the assignment problem regards finding a perfect matching in a bipartite graph G whose partite sets each have n nodes. In this case, M is an **adjacency matrix** representation of G, where $M[i, j] = 0$ if there is an edge between vertices i and j, and $M[i, j] = 1$ otherwise. A perfect matching in G corresponds to the solution of the assignment problem in M. Formally, G has a perfect matching if and only if

$$\min_{\sigma} \sum_{i=1}^{n} M[i, \sigma_i] = 0,$$

where the min is taken over all permutations $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$.

	b_1	b_2	b_3	b_4	b_5
a_1	0	1	0	1	0
a_2	0	1	1	1	1
a_3	1	1	0	1	1
a_4	1	0	1	1	0
a_5	1	1	1	0	1

Figure 3: A perfect matching corresponding to the graph in Figure 2. The a_i are the jobs, and the b_i are the workers

The Potential

We use a graph algorithm to solve the assignment problem in $O(n^3)$ time. Consider a graph complete bipartite G with vertex set $W \cup J$ (workers and jobs as vertices) and where two vertices are adjacent if and only if one is a worker and the other a job. This graph is bipartite, then, with bipartition (W, J); and it has $2n$ nodes and n^2 edges.

We assign to each worker or job i a **potential** $p_i \geq 0$ with the requirement that

$$\forall i \in W, \forall j \in J : p_i + p_j \leq M[i,j].$$

Initially we set $p_i = 0$ for every $i \in W \cup J$.

Consider a matching $E^* \subseteq W \times J$.[1] We say that E^* is a **full matching** if $|E^*| = n$.[2] Note that by construction

$$\sum_{i \in W \cup J} p_i \leq \min_{E^* : |E^*| = n} \sum_{(i,j) \in E^*} M[i,j], \qquad (*)$$

where the left hand side is called the **global potential** and the right hand side is the minimal weight of any matching in G.

[1] Note that $W \times J$ represents $E(G)$ where the pair (i, j) represents the edge between worker i and job j.

[2] When G has n edges, a matching is full if and only if it is perfect.

We say that an edge (i, j) in G is **tight** if $p_i + p_j = M[i, j]$. The following algorithm finds a full matching E^* such that every edge in G is tight; this must be optimal as we obtain equality in $(*)$.

$E^* = \emptyset$
for $i = 1$ to n do
  let $A_i = \{1, 2, \ldots, i\}$ [3]
  update E^* so that it is a subset of $A_i \times B$, and $|E^*| = i$

[3] before this step E^* is a vertex-disjoint subset of $A_{i-1} \times B$ with only tight edges; $|E^*| = i - 1$.

Note that if the i-th update of E^* takes time $O(n^2)$, then the overall time is $O(n^3)$. We now turn to the algorithm for updating E^*.

$A_i = \{1, 2, \ldots, i\}$
$Z = \{i\}$
while *true*
  $\Delta = \min_{k \in Z \cap A_i,\ \ell \in B \setminus Z} (M[k, \ell] - p_k - p_\ell)$
  $(k^*, \ell^*) = \operatorname{argmin}(M[k, \ell] - p_k - p_\ell)$
  $\forall \ell \in Z \cap B : p_\ell = p_\ell - \Delta$
  $\forall k \in Z \cap A_i : p_k = p_k + \Delta$
  if ℓ^* is matched (i.e. $\exists m^* : (m^*, \ell^*) \in E^*$)
    $Z = Z \cup \{m^*\} \cup \{\ell^*\}$ (green example)
  else (purple example)
    $\exists$ path in Z following only edges of E^* or newly added edges of E^* from ℓ^* to i. On that path, flip all edges to obtain a new E^*, now with $|E^*| = i$ and halt.

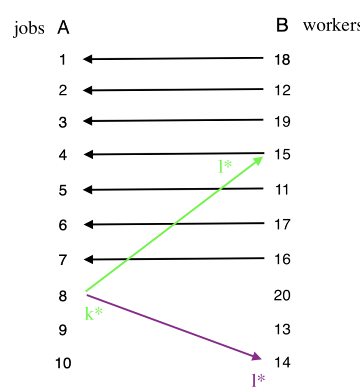

Figure 4: Updating E^*. The edges of E^* are those pointing from B to A.

Augmenting Path

If one starts at $i = 6$ in the example of figure 5, following the arrows, Z is the set of nodes that can be reached from i.

From the terminal $l^* = 16$, follow the arrows backwards to get back to i:

$$16 \longrightarrow 2 \longrightarrow 12 \longrightarrow 4 \longrightarrow 21 \longrightarrow 6$$ } called the augmenting path

Reverse the edges get a new matching E^*. Note that $|E^*| = i$ after this operation.

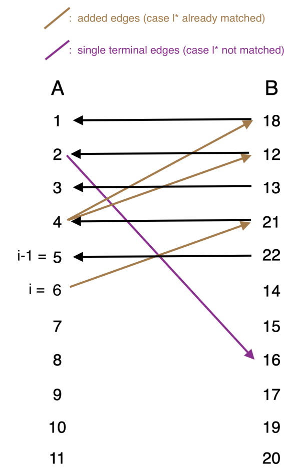

Figure 5: Augmenting path

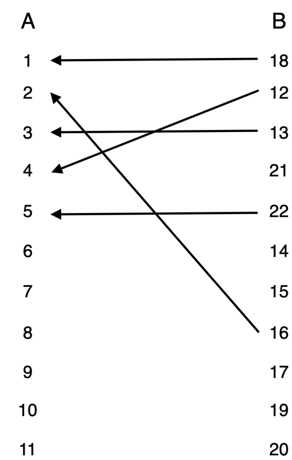

Figure 6: Reversed edges

Checking Things

Note that updating the potentials is okay: after processing (k^*, ℓ^*), only the vertices in Z are affected (Note also that $k^* \in Z$ and $\ell^* \notin Z$). We also have the following clarifying remarks:

(1) If $k, \ell \notin Z$ or $k, \ell \in Z$: $p_k + p_\ell$ remain the same so that $p_k + p_\ell \leq M[k, \ell]$;

(2) If $k \notin Z$ and $\ell \in Z$: $p_k + p_\ell$ decreases by Δ so that $p_k + p_\ell \leq M[k, \ell]$;

(3) $k \in Z$ and $\ell \notin Z$: $p_k + p_\ell$ increases by Δ, but by the choice of which, we still have $p_k + p_\ell \leq M[k, \ell]$;

(4) $(k, \ell) \in E^*$: we are in case (1), and thus, $p_k + p_\ell = M[k, \ell]$ before and after the update of the potentials; and

(5) After the update, if ℓ^* is not part of the vertices of E^*, then $p_{k^*} + p_{\ell^*}$ increases by Δ (case (3)), so after the update, $p_{k^*} + p_{\ell^*} = M[k^*, \ell^*]$, so that the edge (k^*, ℓ^*) becomes tight.

Therefore, the potential condition holds, and all edges added to form the augmenting path are tight, and tight edges remain tight!

Time Complexity Analysis

The step to go from A_{i-1} to A_i starts by building a set Z, which can grow to at most size i on the A-side. Each step requires at most $O(n)$ work to update the potentials. Hence, in $O(n \cdot i) = O(n^2)$ time, we obtain a matching $E^* \subseteq A_i \times B$ from a matching $E^* \subseteq A_{i-1} \times B$.

The Hungarian method for the assignment problem goes back to Kuhn (1955), who proposed an $O(n^4)$ algorithm. The $O(n^3)$ version presented here is due to Tomizawa (1971) and Edmonds and Karp (1972). Kuhn named the method after Hungarian mathematicians Kőnig and Egervary. It should be noted that a similar algorithm was already known by Jacobi before 1890.

References

Jack Edmonds and Richard M Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

Nobuaki Tomizawa. On some techniques useful for solution of transportation network problems. *Networks*, 1(2):173–194, 1971.