CS 163 & CS 265: Graph Algorithms

Week 8: Matching

Lecture 22: The assignment problem

David Eppstein
University of California, Irvine

Fall Quarter, 2022

Overview

Complete bipartite graphs

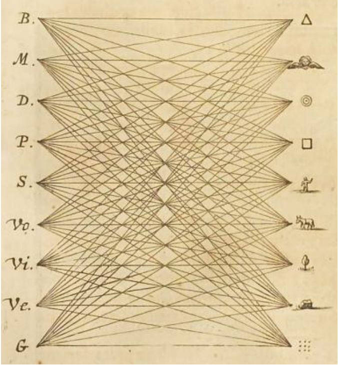
Bipartite: Vertices are divided into two independent sets

All edges go from one independent set to the other

Complete: All other edges that could be included are included

$K_{a,b}$: complete bipartite with a vertices on one side, b on other

Balanced complete bipartite graph: $K_{n/2,n/2}$, same number on each side

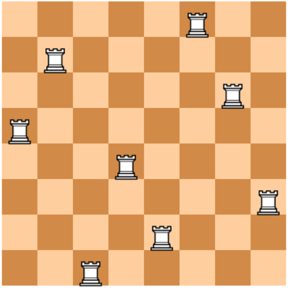


A balanced complete bipartite graph $K_{9,9}$ from *Ars Magna Sciendi*, Athanasius Kircher, 1669

Matching in complete bipartite graphs

Perfect matching: A matching that matches every vertex

In balanced complete bipartite graphs, every matching can be completed to a perfect matching



Eight rooks puzzle:
place 8 rooks on chessboard,
no two attacking each other

Much easier than 8 queens

Equivalent to matching in a graph with rows and columns as vertices and squares as edges (not vertices!)

We can also describe these matchings as permutations
e.g. the rook placement above is the permutation 57142863
where the ith digit is the row of the rook in column i

The assignment problem

Find a minimum-weight maximum matching
in a weighted bipartite graph

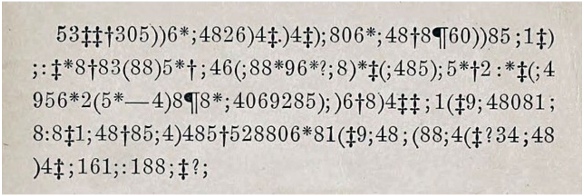
In many applications, graph is a complete bipartite graph, and
maximum matchings are perfect matchings

If $\nexists$ perfect matching, can add dummy vertices and edges of large weight to make it exist, without changing the minimum-weight matching or increasing n and m much $\Rightarrow$

Find a minimum-weight perfect matching
in a weighted bipartite graph

Example from machine learning

Suppose you are trying to recover an unknown permutation (for instance, decode a substitution cipher or cryptogram)


53‡‡†305))6*;4826)4‡.)4‡);806*;48†8¶60))85;1‡)
;:‡*8†83(88)5*†;46(;88*96*?;8)*‡(;485);5*†2:*‡(;4
956*2(5*—4)8¶8*;4069285);)6†8)4‡‡;1(‡9;48081;
8:8‡1;48†85;4)485†528806*81(‡9;48;(88;4(‡?34;48
)4‡;161;:188;‡?;

From *The Gold Bug*, Edgar Allen Poe, 1843

You have (somehow) computed likelihoods $P_{i,j}$ that input symbol i maps to output symbol j

Overall likelihood for any particular permutation π is $\prod_i P_{i,\pi(i)}$

The maximum likelihood estimator (most likely permutation) is the solution to the assignment problem for weight$(i,j) = -\log P_{i,j}$

Weighted matching in TSP approximation

Recall the Christofides–Serdyukov algorithm for approximating the traveling salesperson problem:

- Find a minimum spanning tree T of the input graph G
- Build a complete graph K (all edges) on the odd vertices of T
- Weight each edge in K by shortest path distance in G
- Find a minimum weight perfect matching M
- Return an Euler tour of $T \cup M$

This is not an assignment problem because K is not bipartite

Similar but more complicated algorithms can solve it in same time bounds as the assignment problem

The Hungarian algorithm

Some history

(From "Jenő Egerváry: from the origins of the Hungarian algorithm to satellite communication", Silvano Martello, 2009)

- The "Hungarian algorithm" for the assignment problem was discovered by Carl Gustav Jacob Jacobi (famous German mathematician) in 1840, in connection with solving systems of differential equations, but not published until 1865
- Matching was studied in 1916 by Dénes Kőnig, and weighted matching in 1931 by Jenő Egerváry, both Hungarian
- The assignment problem was formulated in 1950 by Robert L. Thorndike as an application of matching job openings to applicants, and named in 1952 by Votaw and Orden
- Harold Kuhn rediscovered Jacobi's algorithm in 1955 and named it the Hungarian algorithm after Kőnig and Egerváry
- The connection between this algorithm and the work of Jacobi went unnoticed until Ollivier and Sadik wrote about it in 2007

Hungarian algorithm in its simplest form

Start from an empty matching

Repeat $n/2$ times: find a minimum-weight alternating path

The weight of an alternating path is how much it increases the weight of a matching: the sum of the weights of its unmatched edges, minus the sum of the weights of its matched edges

Two problems:

- Negative contribution of matched edges suggests using Bellman–Ford to find each path, unnecessarily slow
- Written this way, it's not obvious why it finds the best matching

Assignment problem with vertex heights

Adjusted weight of an edge: its original weight, minus the heights of both endpoints

- Affects all perfect matchings equally
- Unlike shortest-path reweighting, we treat both endpoints the same as each other (because input graph is undirected)

Invariants: adjusted weights ≥ 0 and matched edges $= 0$

- Easy to achieve initially: just make all heights very negative
- Eliminates the subtraction in weight of alternating paths
- Allows shortest alternating path to be found by Dijkstra's algorithm, just like we found unweighted short alternating paths by a variant of BFS
- Final matching has total weight zero, minimum possible, so it is optimal among all perfect matchings

Hungarian algorithm with vertex heights

Initialize heights to make adjusted weights ≥ 0

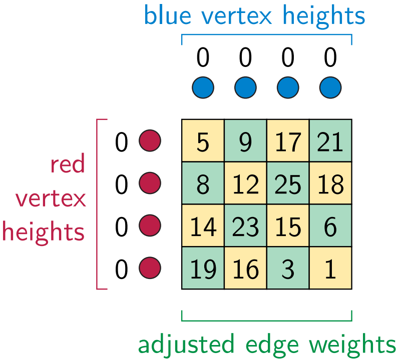
Repeat $n/2$ times:

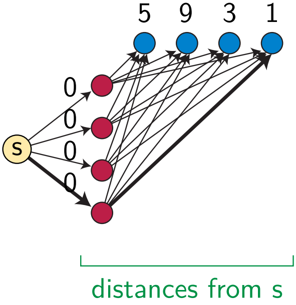
- Add an artificial start vertex s, with edges of weight zero to all unmatched red vertices, direct all unmatched edges red-to-blue and all matched edges blue-to-red
- Use Dijkstra's algorithm to find adjusted distances from s to all other vertices, including the shortest alternating path (shortest path from s to an unmatched blue vertex)
- Adjust heights: subtract distance at red vertices, add distance at blue (zeros all shortest-path edges leaving others ≥ 0)
- Use the shortest alternating path (which now has all adjusted edge weights zero) to increase size of matching

Time: $n/2$ runs of Dijkstra, $O(nm + n^2 \log n)$ total

Example

Initial weights, and first path



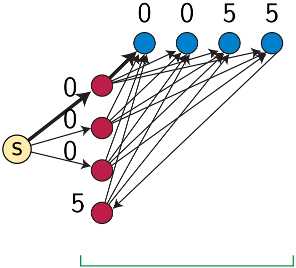


New weights after one matched edge

blue vertex heights

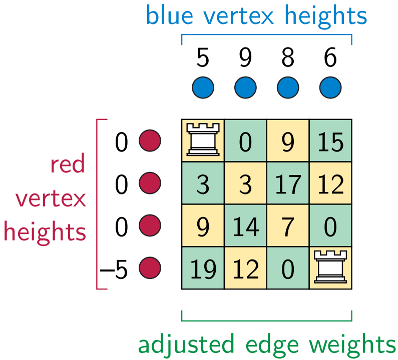
red vertex heights	5	9	3	1
0	0	0	14	20
0	3	3	22	17
0	9	14	12	5
0	14	7	0	♖

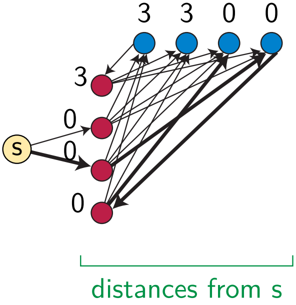
adjusted edge weights



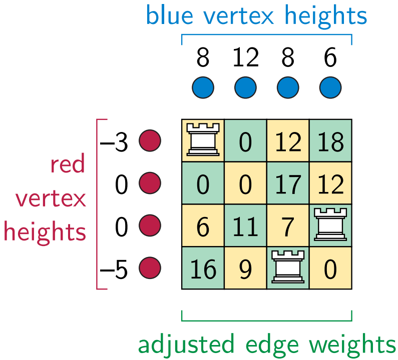
distances from s

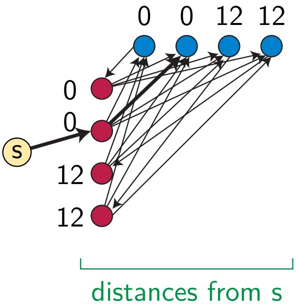
New weights after 2nd matched edge





New weights after 3rd matched edge





Final matching

Morals of the story

Bipartite graphs and matching algorithms
have both been studied for a long time

Assignment problem can be used to pick out the most likely
permutation given an array of likelihoods of individual pairings

Can be solved by repeatedly finding alternating paths using
Dijkstra, adjusting vertex heights to keep edges non-negative

Same reweighting gives an easy proof that the result is optimal

References I

Jenő Egerváry. Matrixok kombinatorikus tulajdonságairól. *Matematikai és Fizikai Lapok*, 38:16–28, 1931.

C. G. J. Jacobi and C. W. Borchardt. De investigando ordine systematis aequationum differentialium vulgarium cujuscunque. *Journal für die reine und angewandte Mathematik*, 64:297–320, 1865. doi:10.1515/crll.1865.64.297.

Athanasius Kircher. *Ars Magna Sciendi Sive Combinatoria*. 1669.

Dénes Kőnig. Gráfok és alkalmazásuk a determinánsok és a halmazok elméletére. *Matematikai és Természettudományi Értesítő*, 34:104–119, 1916.

Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955. doi:10.1002/nav.3800020109.

References II

Silvano Martello. Jenő Egerváry: from the origins of the Hungarian algorithm to satellite communication. *Central European Journal of Operations Research*, 18(1):47–58, 2010. doi:10.1007/s10100-009-0125-z.

François Ollivier and Brahim Sadik. La borne de Jacobi pour une diffiété définie par un système quasi régulier. *Comptes Rendus Mathématique, Académie des Sciences, Paris*, 345(3):139–144, 2007. doi:10.1016/j.crma.2007.06.010.

Robert L. Thorndike. The problem of classification of personnel. *Psychometrika*, 15(3):215–235, 1950. doi:10.1007/bf02289039.

D. F. Votaw and A. Orden. The personnel assignment problem. In *Symposium on Linear Inequalities and Programming (SCOOP 10)*, pages 155–163. US Air Force, 1952.