Tuesday, March 1, 2022 11:07 AM

Lecture: Flow Networks
Unit: 7.4
Instructor: Carlos C.L

CPCFI

Flow network

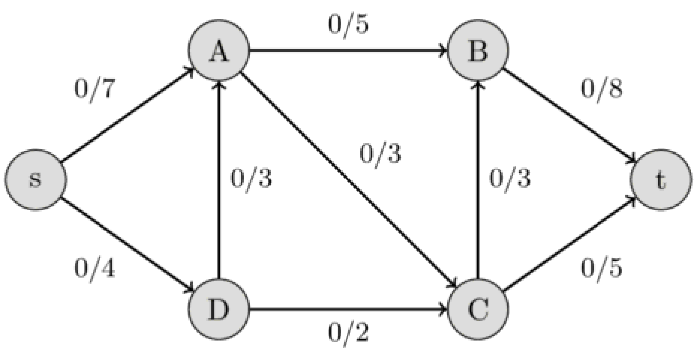
A directed graph G(V, E) combined with a capacity function c: E -> {R > 0} and two vertices, a source(s) and a sink(t).

Flow

A function f: E -> {R > 0}. It must satisfy the following conditions:

$$f(e) \le c(e)$$

$$\Sigma\, f((u, v)) = \Sigma\, f((v, u)) \text{ for any node } v$$



We can prove the following holds given this conditions:

$\Sigma\, f((s, u) = \Sigma\, f((u, t)) =$ flow of the network

Ford-Fulkerson (published1956)

Our problem is to find the flow function f, that maximizes the flow of the network.

Residual Graph

The residual graph is the same graph but with reverse edges and a different capacity function, the residual capacity.

Residual capacity

How much we can increase the flow for a given edge:

$$rc(e) = c(e) - f(e)$$

Reverse edges

For any edge (u, v) in the original graph, we define a reverse edge (v, u) in the residual graph

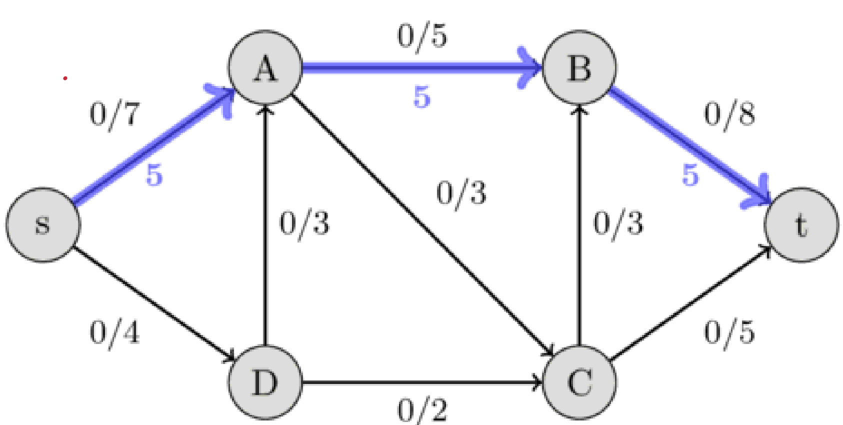
$$f((v, u)) = - f((u, v))$$
$$c((v, u)) = 0$$
$$rc((v, u)) = f((u, v))$$

Augmented path

It's a path on the residual graph with all edges having positive residual capacity.

Bottleneck capacity

The lowest residual capacity on an augmented path.

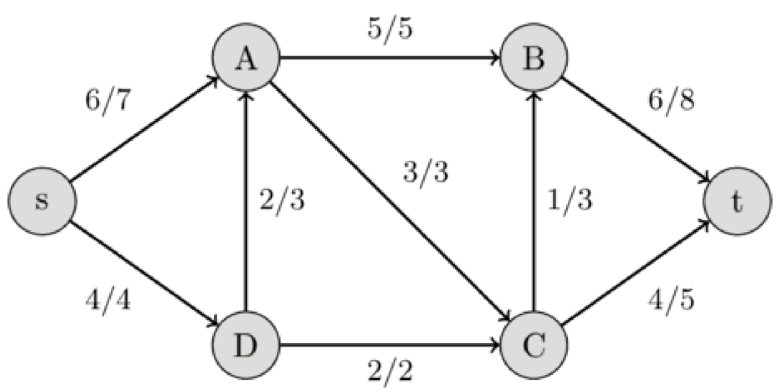


Algorithm

1. **Find augmented path from s to t (use BFS or DFS)**
2. **Find bottleneck capacity = bc**
3. **Augment each edge on the path**

   f((u, v)) += bc
   f((v, u)) -= bc
   rc((u, v)) -= bc
   rc((v, u)) += bc
4. **Repeat until no augmented paths can be achieved**

Solution



Edmonds Karp

If we use BFS for step one then we are solving with Edmonds Karp.

Complexity:

Time: O(EF), E = #Aristas, F = Max flow

Time Edmons Karp: O(EV^2)

Space Edmons Karp: O(V+E)

Problems:

[CSES Download Speed]()

Dinic's Algorithm

Solves max flow problem.
[Dinic's Algorithm | Network Flow | Graph Theory](#)

Complexity:

Time complexity: O(V^2E)

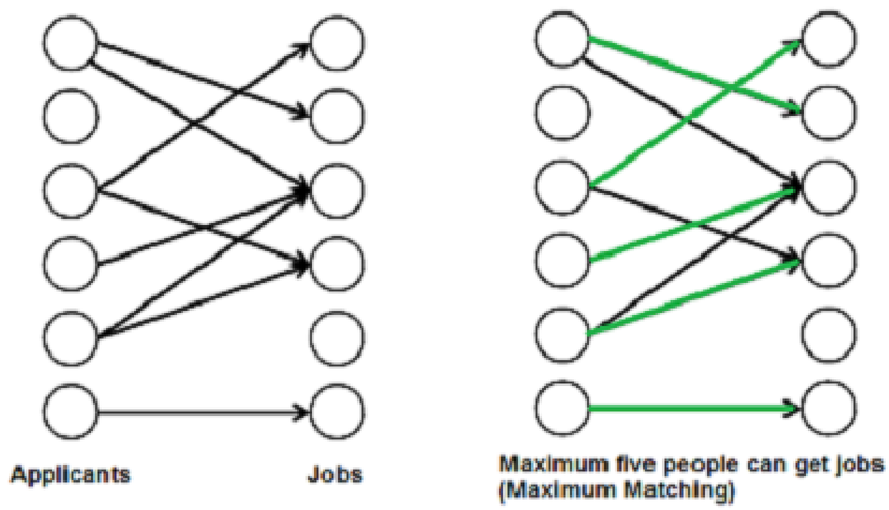
For unit networks:

A unit network is a network where all edges have unit capacity.

Time complexity: O(EV^1/2)

Unweighted Bipartite Matching

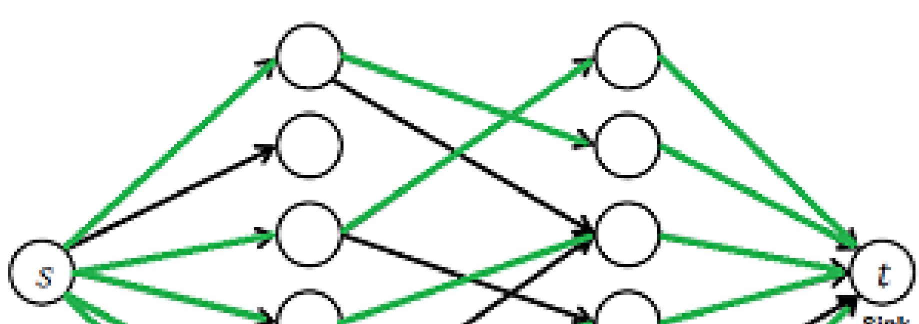
We have a bipartite graph representing applicants and jobs. There is an edge (applicant, job) if the applicant has the requirements to do the job.



We want to maximize the number of applicants that get a job.

Idea

1. Transform problem to max flow problem
2. Make an imaginary source and target, connect source to applicant and jobs to applicants
3. Find max flow in this graph

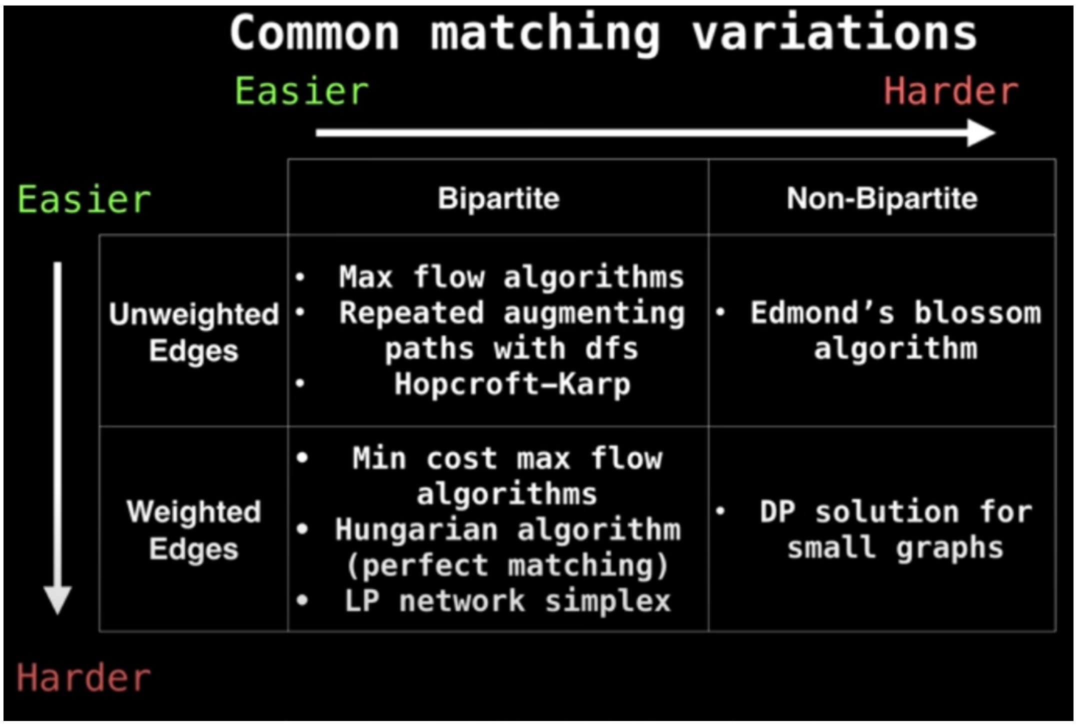


The maximum flow from source to sink is five units. Therefore, maximum five people can get jobs.

Problems:

[CSES School Dance]

Matching Problems



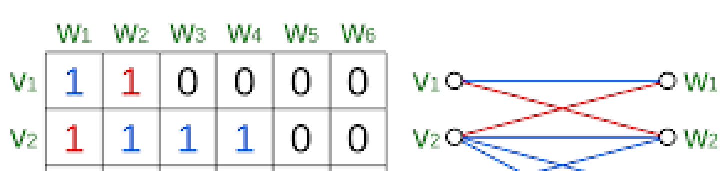
Resources:

[Unweighted Bipartite Matching | Network Flow | Graph Theory]

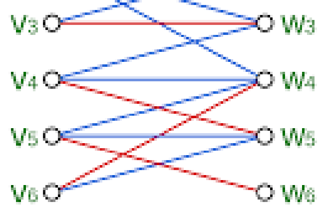
Assignment problem

First Example:

There are N jobs and N workers. Some jobs can only done by some workers, and we know the cost it takes for the worker to do the job. ***Make a 1:1 assignment (each worker gets assigned only one job and each job gets assigned only one worker) so that the total cost is minimized.***

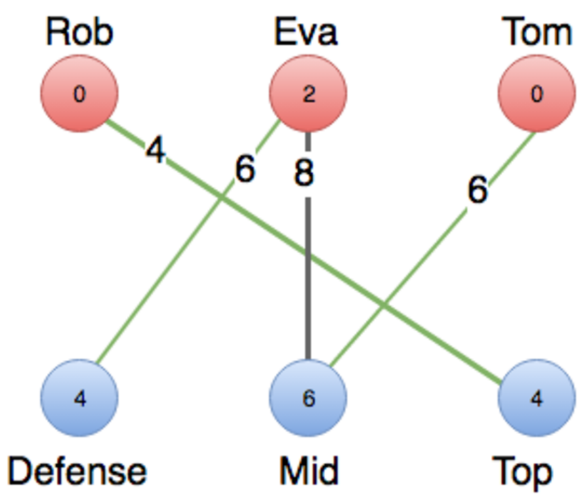


V3	0	1	1	0	0	0
V4	0	0	1	1	1	0
V5	0	0	0	1	1	1
V6	0	0	0	1	1	0



Second Example:

A football coach wants to assign a position to each of its players, he knows the skill of each player for the positions. Make a 1:1 assign (each player has only one position and each position has only one player) so that the total skill is maximized.



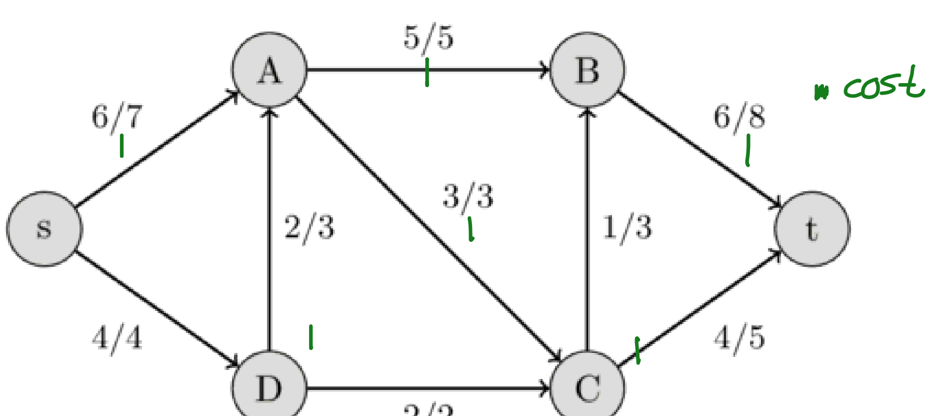
Min Cost Max Flow Problem

Suppose we have a network flow, G(V, E, capacity) and a cost function, cost: E -> {R > 0}.

cost(e) = cost per unit flow for edge e

Total cost(G) = Σ cost(e)*flow(e)

We want to find the max flow in this network and the lowest possible total cost to achieve this flow.



Algorithm:

1. Find augmented path from s to t with the lowest cost possible cost (use Bellman Ford)
2. Find bottleneck capacity = bc
3. Augment each edge on the path

   f((u, v)) += bc
   f((v, u)) -= bc
   cp((u, v)) -= bc
   cp((v, u)) += bc
4. Repeat until no augmented paths can be achieved

Solve Initial problem using Min cost Max Flow:

- What nodes should we create and how should we connect them to the graph?

Complexity:

Time: O(V^5)

Hungarian Algorithm

It can solve the assignment problem on O(V^3)

Problems:

CSES Police Chase

Resources:

Brilliant – Matching Problem