Lecture #18

CS 170

Spring 2021

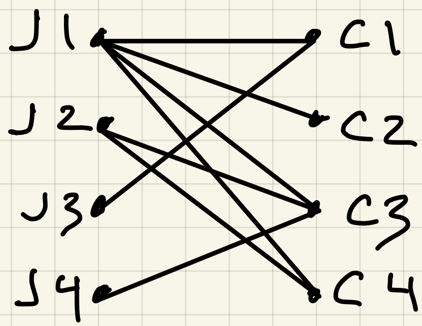
Reductions

- Reducing Problem A to Problem B
  = using a subroutine for solving Problem B to solve Problem A

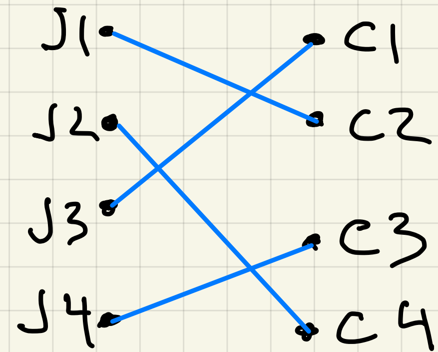
- Good news: "Fast" algorithm for B provides a fast algorithm for A
- Bad news: If we know A is "hard" then B must be hard too
- Assumptions: Converting input of A ⇒ input of B and answer for B ⇒ answer for A not expensive

Examples

- Good news:
  - Reduce Bipartite Matching (BM) to MaxFlow (MF)



How many jobs and computers can we pair up?

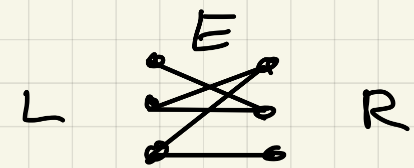


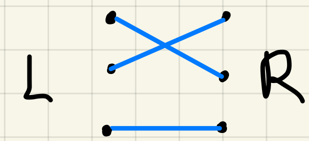
all 4!

  - Reduce any polynomial time problem to LP
  - Reduce matrix inversion to matrix multiply
- Bad news: Chap 8, NP-completeness

Bipartite Matching (BM)

Input: Bipartite Graph $G = (L, R, E)$, $E \subseteq L \times R$





Matching: $M \subseteq E$ where no pair of edges in M touch same vertex

Goal: Maximum matching: maximize $|M|$

(not same as "maximal matching" = matching to which no more edges can be added)

Ex: L = jobs, R = computers

L = people, R = partners

Connect BM and Max Flow (MF) (1/2)

BM:

undirected $G = (L, R, E)$

matching $M \subseteq E$ touches
any vertex at most once

goal: maximize $|M|$

MF

directed graph $G = (V, E)$
with source $s \in V$
and sink $t \in V$

edge "capacities" $c_e \geq 0$

goal: maximize "flow"
from s to t
subject to capacity limits,
conservation of flow

To solve BM using MF:

need to identify s and t

need to set capacities c_e

need to direct edges

need to connect $|M|$ with flow

Connect BM and Max Flow (MF) (2/2)

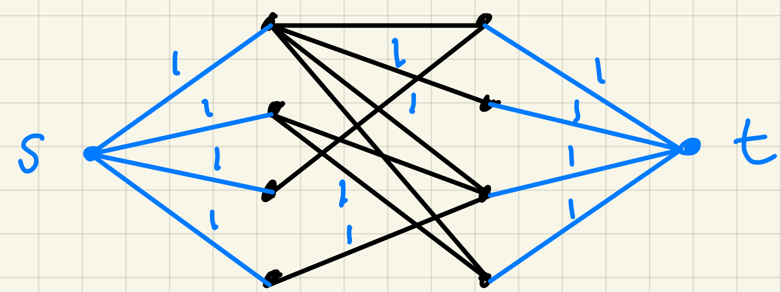
To solve BM using MF:

need to identify s and t

need to set capacities c_e all $c_e = 1$

need to direct edges all left to right

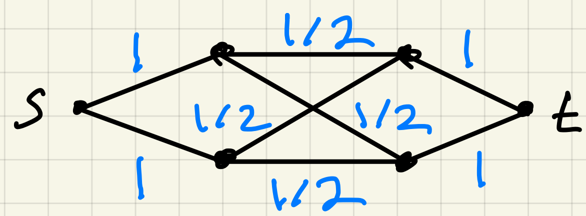
need to connect $|M|$ with flow



max flow = mincut = 4

$M = \{(L1, R2), (L2, R4), (L3, R1), (L4, R3)\}$ $|M| = 4$

What could go wrong?



Correct solution to MF
But not to BM

Recall MF algorithm (Ford-Fulkerson)

repeat
  find a path from s to t with capacity>0
  increase flow along path by maximum amount
until no path from s to t with capacity>0

$\Rightarrow$ if inital capacities all integer, flows along each edge will be integer

$\Rightarrow$ For BM, all capacities $=1 \Rightarrow$ all flows along edges either 0 or 1

Claim: There is a 1-1 correspondence between solutions to BM and integer solutions to MF

- Let M be a maximum matching. For each $(u,v) \in M$, let flow be 1 along $s \xrightarrow{1} u \xrightarrow{1} v \xrightarrow{1} t$
  totalflow = # edges $(u,v) = |M|$
- Let $v(E)$ be integer solution to MF where $v(e)$ = flow on edge e
  $\exists$ one edge (s,u) for each $u \in L \Rightarrow$ inflow to each $u \in \{0,1\} \Rightarrow$ out flow $\in \{0,1\}$
  $\exists$ one edge (v,t) for each $v \in R \Rightarrow$ out flow from each $v \in \{0,1\} \Rightarrow$ inflow $\in \{0,1\}$
  $\Rightarrow$ at most one flow (u,v) from any u, or to any v
  $\Rightarrow$ Matching, with $|M|$ = total flow

Defining Reductions

Def: Problem A reduces to Problem B ($A \to B$) if there are "efficient" algorithms Preprocess and Postprocess such that solution $A(X)$ is

$$X \to \boxed{\text{Preprocess}} \xrightarrow{Y} \boxed{B} \xrightarrow{B(Y)} \boxed{\text{Postprocess}} \to A(X)$$

Ex: $A = BM$ and $B = MF$

Preprocess: add (s,u), (v,t), directions, capacities

Postprocess: read edges (u,v) with $flow = 1$

$Cost = O(|V| + |E|)$

- Efficient algorithm for $B \Rightarrow$ efficient algorithm for A
- No efficient algorithm for $A \Rightarrow$ no efficient algorithm for B

Circuit Value Problem (CV)

- Def: A Boolean Circuit is a DAG with
  - input nodes $x_i = 0$ or 1
  - AND nodes $x_i, x_j \to \text{AND} \to x_k = x_i \wedge x_j$
  - OR nodes $x_i, x_j \to \text{OR} \to x_k = x_i \vee x_j$
  - NOT nodes $x_i \to \text{NOT} \to x_k = \overline{x}_i$
  - output nodes: subset of resulting x_k
- CV: Given a Boolean Circuit, is its output $=1$?
  - Topologically sort DAG, evaluate it
- Claim: any efficient algorithm $\to$ CV $\to$ LP

  so any efficient algorithm $\to$ LP

Any efficient algorithm $\rightarrow$ CV $\rightarrow$ LP

- Informal argument (CS 172 discusses Turing machines)
  - A computer with poly-sized memory can run algorithm in poly-time
  - Have 1 copy of circuit representing internal state of computer for each time step, with output of copy i = input for copy $i+1$
  - Size of entire circuit is polynomial in input
- CV $\rightarrow$ LP
  - Each Boolean variable $x_i \rightarrow 0 \le x_i \le 1$
  - $x_k = x_i \wedge x_j \longrightarrow x_k \le x_i,\ x_k \le x_j,\ x_k \ge x_i + x_j - 1$
  - $x_k = x_i \vee x_j \longrightarrow x_k \ge x_i,\ x_k \ge x_j,\ x_k \le x_i + x_j$
  - $x_k = \bar{x}_i \longrightarrow x_k = 1 - x_i$

Each input $= 0$ or $1 \Rightarrow$ each $x_k = 0$ or 1

only one feasible point $\Rightarrow$ any objective function works

Matrix Multiply (MM) $\longleftrightarrow$ Matrix Inversion (MI)

- Each one reduces to other
  - "Fast" algorithm for one $\Rightarrow$ works for other

- Easy direction: MM $\rightarrow$ MI want $A \cdot B$

Form $X = \begin{bmatrix} I & -A & 0 \\ 0 & I & -B \\ 0 & 0 & I \end{bmatrix}$, compute $X^{-1} = \begin{bmatrix} I & A & AB \\ 0 & I & B \\ 0 & 0 & I \end{bmatrix}$

If inverting $n \times n$ X costs $O(n^e)$

then multiplying $n \times n$ $A \cdot B$ costs

$$O(n^2) + O((3n)^e) + O(n^2) = O(n^e)$$

- Eg: $e = 3$ (usual alg), $e = \log_2 7 \approx 2.81$ (Strassen)
  $e \approx 2.373$ (world record from Oct 2020)

Matrix Multiply (MM) $\longleftrightarrow$ Matrix Inversion (MI)

- Trickier Direction: MI $\rightarrow$ MM

2×2 Gaussian Elimination: Cost: $I_n(n)$

$$\begin{bmatrix}A & B\\ C & D\end{bmatrix} = \begin{bmatrix}I & 0\\ \underbrace{C\cdot A^{-1}}_{Y} & I\end{bmatrix}\cdot\begin{bmatrix}A & B\\ 0 & \underbrace{D-\overbrace{C\cdot A^{-1}\cdot B}^{Y\cdot B}}_{S}\end{bmatrix}$$

$$= \underbrace{I_n\left(\tfrac{n}{2}\right)}_{A^{-1}} + \underbrace{n^e}_{Y} + \underbrace{n^e}_{Y\cdot B} + \underbrace{n^2}_{S}$$

$$\begin{bmatrix}A & B\\ C & D\end{bmatrix}^{-1} = \begin{bmatrix}A & B\\ 0 & S\end{bmatrix}^{-1}\cdot\begin{bmatrix}I & 0\\ Y & I\end{bmatrix}^{-1} = \begin{bmatrix}A^{-1} & \overbrace{-A^{-1}\cdot B\cdot S^{-1}}^{Z}\\ 0 & S^{-1}\end{bmatrix}\cdot\begin{bmatrix}I & 0\\ -Y & I\end{bmatrix}$$

$$= \begin{bmatrix}A^{-1}-Z\cdot Y & Z\\ -S^{-1}\cdot Y & S^{-1}\end{bmatrix}$$

$$+ \underbrace{I_n\left(\tfrac{n}{2}\right)}_{S^{-1}} + \underbrace{2n^e}_{Z} + \underbrace{2n^e}_{Z\cdot Y,\ S^{-1}\cdot Y} + n^2$$

$$I(n) = 2\,I_n\left(\tfrac{n}{2}\right) + O(n^e) = O(n^e)$$

by Master Theorem for recurrences