MIN-COST (PERFECT) BIPAPTITE MATCHING

The problem:

- **Input:** Bipartite graph $G = (V \cup W, E)$
  Each edge $e \in E$ has a cost c_e
- **Assumption:** For convenient
  - $|V| = |W| = n$ (if $|V| < |W|$, add dummy nodes)
  - G has a perfect matching
  - $c_e \geq 0 \quad \forall e \in E$
- **Goal:** Find Perfect Matching which minimize $\sum_{e \in E} c_e$

Can we reduces to max-flow ?

Previously, we saw problem like Bipartite Matching can be reduced to Max-flow problem, can we do the same here ?

→ No, because there is a difference between capacity in Bipartite Matching and cost in this problem. One is about constraining, another is about optimality

Strategy to come up with algorithm

- Sufficient conditions for optimality ?
- How to iteratively to achieve those conditions ?

↳ Example: Max-flow algorithm

- Condition for optimality: No path between $s \rightsquigarrow t$ in G_f
- Iteratively achieve those conditions:
  We explored 2 paradigm to ensure that conditions
  - Paradigm #1: Find a path between $s \rightsquigarrow t$, work towards disconnecting s and t by "fill" it with flows.
    (Ford-Fulkerson, Edmonds-Karp)
  - Paradigm #2: Assume there is no path $s \rightsquigarrow t$, by relaxing the condition checking path $s \rightsquigarrow t$ with Preflow, work towards restoring Flow
    (Push-relabel)

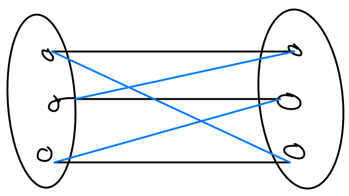
Process of coming up with an algorithm

- We'll follow the second Paradigm to come up with an algorithm to find Perfect Matching with Minimum Cost.
  - The high level idea is: Come up with some invariants that implies conditions for optimality, relax "feasibility". Then work towards "restoring" the "feasibility".

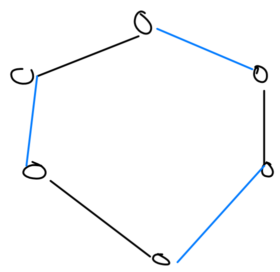
Step 1-a: Conditions for optimality

Given a Perfect matching, how do you know it is the best possible (min-cost)?

- M – alternating cycle:

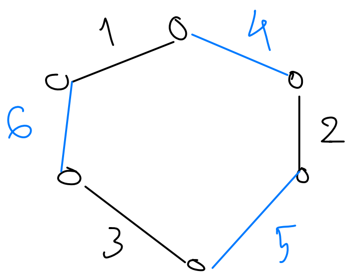


or



Definition: A cycle C of G is M-alternating if every other edge of C is in matching M.

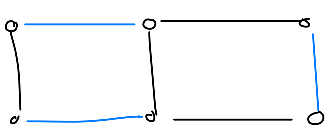
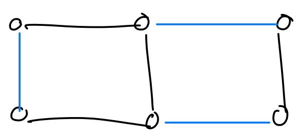
- Negative M-alternating cycle:

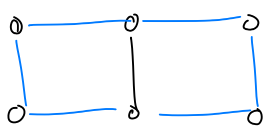


Definition: cycle C is negative if the total cost of edges in the matching exceed total cost not in matching

$$\sum_{e \in M_1} c_e > \sum_{e \in M_2} c_e$$

- Symmetric differencen between 2 matchings

M =  M' = 

$M \oplus M' =$ 

Think of it as XOR operation

Claim: $\exists$ a perfect matching is min-cost $\Leftrightarrow$ No negative cycle in G

Proof:

- Forward direction: $\exists$ perfect matching is min-cost $\Rightarrow$ No negative cycle

Let M^* be the perfect matching with min-cost

$$\Rightarrow \sum_{e \in M^*} c_e \leq \sum_{e \in M} c_e \qquad M: \text{any other perfect matching}$$

Assume the cycle C^* enclosing perfect matching M^*, by definition of negative cycle:

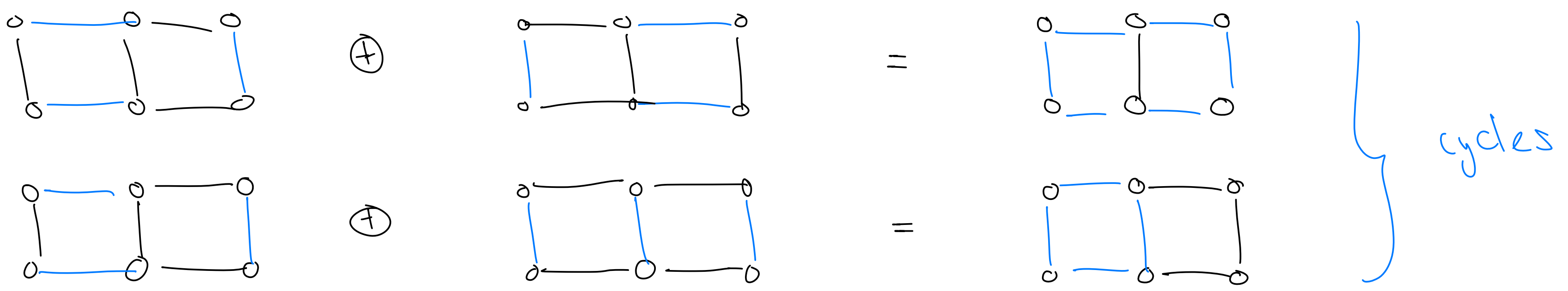
$$\sum_{e \in M^*} c_e > \sum_{e \in M} c_e$$

$\Rightarrow$ Contradict definition of M^*

- In other words, given an M-alternating cycle C, we can always transform it into an M'-alternating cycle C'. In this case, after transforming, since we assume C is negative, total cost of M' is lesser than M, contradicting definition of M.

- Backward direction: No negative cycle $\Rightarrow$ $\exists$ perfect matching with min-cost

Consider $M \oplus M'$:



Observations: given a cycle C and $M \oplus M'$, we can transform between $M \leftrightarrow M'$ by toggle the edges in the cycle

We have: non negative cycle $\Rightarrow$ cost in $\leq$ cost out

If we toggle a perfect matching and we toggle it in nonnegative cycle $\Rightarrow$ cost will non decrease

$\Rightarrow$ cost M' (transformed) $\geq$ cost M (original)

$\Rightarrow$ cost M is minimum

$\Rightarrow$ Done Conditions for Optimality

(Perfect matching Min-Cost if only if No negative M-alternating cycle.

Step 1.b: Invariants that implies Optimality

Remember we are trying to come up with an algorithm that follows Paradigm 2 - "Invariants hold at all time, relax feasibility, work towards restoring feasibility"

Invariants:

① All reduced costs ≥ 0

Reduced cost of edge (v, w) is

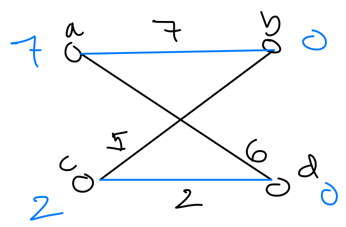
$$c^R_{v,w} = c_{v,w} - p(v) - p(w) \quad \text{where } \begin{cases} p(v): \text{price of } v \\ p(w): \text{price of } w \end{cases}$$

Note: can think of "price" here similar to "height" in Push-Relabel, which serve the purpose of $\boxed{\text{"relaxing feasibility"}}$

② Every edge e in matching M is "tight", meaning:

$$c^R_e = 0$$

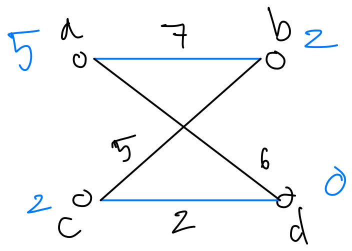
Example:



⇒ Invariant 1 not satisfied:

edge (a, d) has reduced cost < 0

⇒ Invariant 2 satisfied



⇒ Both invariants satisfied

⇒ This matching in Min-cost Perfect Matching

Claim: Invariants hold + M is perfect ⇒ M is optimal (no negative cycles)

.... Proof here

Step 2: the Algorithms

At steps 1.a and 1.b, we proven that:

" Invariants hold:
① reduced cost is non negative
$$c_R(e) = c(e) - p(v) - p(w) \geq 0$$
② $\forall$ edge $e \in M$
$$c_R(e) = 0$$

And M is perfect matching

Then, M is optimal (Min-cost) "

This proof gives us guidelines on how to design our algorithms

An simple intuitive algorithm could be:

" Starting from an empty matching M, do BFS / DFS to pick an edge s.t. the invariants are hold, keep doing it until M is perfect. Set M at termination is optimal "

Hungarian Algorithm

Initialization:

$M = \phi$

$p(v) = 0 \quad \forall v \in (V \cup W)$

Main loop:

While M is not perfect, consider edge e:

If exist a "good path" P:

$M := M \oplus P$ is a bigger matching

$\Rightarrow$ Invariants hold

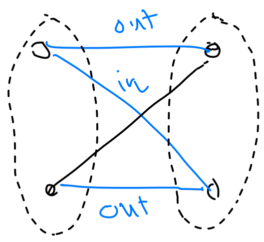
Else:

$\Delta :=$ as large as possible s.t. Invariant 1 holds

$p(v) := p(v) + \Delta$

$p(w) := p(w) - \Delta$

Good path P



- Start left, end right
- v, w are unmatched
- alternate edges, with odd length i.e "first edge is out, last edge is also out"
- all edges in P are tight, meaning $c_R(e) = 0$

Why $M \oplus P$ increase size of M?

- Recall P has odd length, and the first and last edges are "out" $\Rightarrow |P_{out}| = |P_{in}| + 1$
- $M \oplus P$ will toggle in $\leftrightarrow$ out in P.

$\Rightarrow M := M \oplus P$ will increase cardinality by 1

Why?

- $M \oplus P$ doesn't change the price $\Rightarrow$ Invariant 1 holds
- $M \oplus P$ with P "tight"

$\Rightarrow$ Invariant 2 holds