MATH 566: Lecture 15 (10/13/2026)

Today: * all pairs SP
* max flow

All Pairs Shortest Path Problem

Goal: Find SP from i to j $\forall i,j \in N$.

We could solve n SP instances with $s=i$ $\forall i \in N$.

Assumptions

* c_{ij} can be < 0
* no negative cycles

But could we somehow use Dijkstra's algo (it is more efficient)? Note that some c_{ij} could be < 0 to start with.

Yes! We can use **reduced costs**:

Let $\bar{c}_{ij}^{d} = c_{ij} + d(i) - d(j)$ for given $d(i), i \in N$.

Recall: SP optimality conditions: $d(j) \leq d(i) + c_{ij}$.

$\Rightarrow$ If SP optimality conditions are satisfied by $d(\cdot)$, then $\bar{c}_{ij}^{d} \geq 0$.

Def More generally, for a set of node potentials $\pi(i)$ (vector $\bar{\pi}$), the reduced costs are

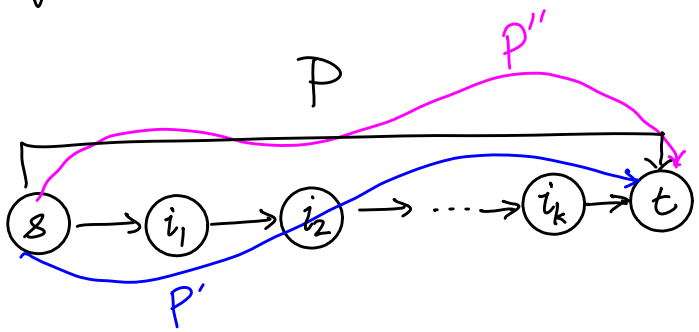
$$c_{ij}^{\bar{\pi}} = c_{ij} - \pi(i) + \pi(j).$$

We will use reduced costs again (in MCF, ...)

Here, we use $\bar{\pi} = -\bar{d}$. So, could we use $c_{ij}^{\bar{d}}$ instead of c_{ij}?

Property A shortest path w.r.t. c_{ij} is also a shortest path w.r.t $c_{ij}^{\bar{d}}$ for a given vector $\bar{d}$ (of $d(i)$'s).

Proof



The ordering of all s-t paths $(P, P', P'', \ldots)$ in terms of their total cost remains the same!

$$\sum_{(i,j)\in P} c_{ij}^{\bar{d}} = \sum_{(i,j)\in P} c_{ij} + \underbrace{d(s) - d(t)}_{\text{constant for a given } \bar{d}}$$

The cost of any s-t path is shifted by only a constant, hence the property holds. □

All-Pairs SP Algorithm

1. Solve SP with $s = 1$ (say). Let $d(j)$ be the SP distance from s to j, $j \in N$.
2. For $s = 2 \ldots n$, solve $SP^{\bar{d}}$ with $c_{ij}^{\bar{d}}$.

Complexity

Step 1. $O(mn)$ FIFO label correcting algorithm

Step 2. Each SP computation takes $O(m + n \log C)$.
(Using Dial's or radix heap implementation of Dijkstra's algorithm)

Overall: $O(mn + n(m + n \log C))$ → (n-1), to be precise

$$= O(mn + n^2 \log C).$$

For small values of C, this algorithm is essentially quadratic, and not cubic.

For the default SP (single source) case, we considered Dijkstra's algo first, and then optimality conditions. We follow the same trend for all pairs SP.

All Pairs Shortest Path Optimality Conditions

Let $d[i,j]$ be the length of some finite walk from i to j. So, $d[i,j]$ is an upper bound for the SP length from i to j.

We also let $d[i,i] = 0 \ \forall i \in N$, and $d[i,j] \leq c_{ij} \ \forall (i,j) \in A$.

↳ follows from the assumption that there are no negative cycles.

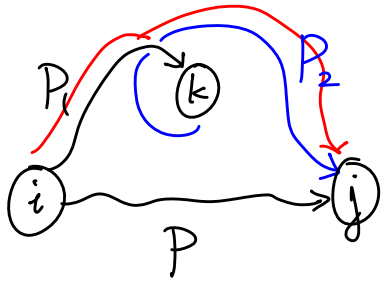
↳ else the arcs themselves are the shortest paths — somewhat trivial!

AMO Theorem 5.5 $d[i,j]$ represent shortest path lengths from i to j iff $d[i,j] \leq d[i,k]+d[k,j] \quad \forall i,j,k$.

Idea of proof

($\Rightarrow$) Let $d[i,j]$ represent SP distances from i to j, $\forall i,j \in N$.

Assume $d[i,j] > d[i,k] + d[k,j]$ for some triplet of nodes i,j,k.



$P_1 \cup P_2$ is a directed walk from i to j. This walk can be decomposed into a union of paths and cycles, from which we could obtain a path from i to j whose total length is $\leq d[i,k]+d[k,j]$ (as there are no negative cycles).

This path contradicts the optimality of $d[i,j]$.

The reverse direction ($\Leftarrow$) follows similar arguments to those used in the corresponding proof for the default SP case (single source). □

Floyd-Warshall algorithm for all pairs SP

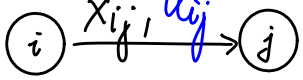
Repeatedly check for violations of all pairs SP optimality conditions, and UPDATE when violations are found.

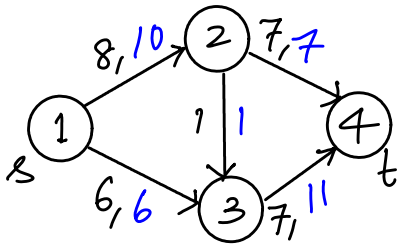
Complexity is $O(n^3)$. → check all triplets of nodes i,j,k. Note that this complexity term does not depend on m (# arcs).

Maximum Flows (AMO Chapter 6)

Goal: Send as much flow as possible (without exceeding arc capacities) from node s to node t in a capacitated network.

Here is an illustration:





value of flow here is

$$v = 8+6 = 7+7 = 14.$$

Consider the mathematical model (linear program):

v → value of flow

$$\max \quad v$$

$$\text{s.t.} \quad \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = \begin{cases} v & \text{if } i = s \\ 0 & \text{if } i \in N \setminus \{s,t\} \\ -v & \text{if } i = t \end{cases}$$

$$0 \le x_{ij} \le u_{ij} \quad \forall (i,j) \in A$$

Compare this model with the default min-cost flow model. Note that v is a variable here! All nodes except s and t are transshipment nodes. And we do not worry about costs c_{ij}'s.

Assumptions

1. $G_R = (N, A)$ is directed.

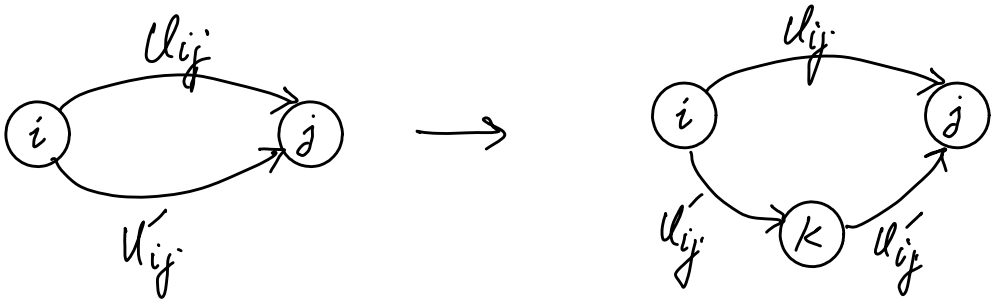
2. $u_{ij} \in \mathbb{Z}_{\geq 0}$ (nonnegative integers) → makes complexity analysis easier.

3. G_R does not contain a directed s-t path P with $u_{ij} = \infty \ \forall (i,j) \in P$. → else, we can keep pushing arbitrarily large amounts of flow from s to t

4. When $(i,j) \in A$, (j,i) is also present in A. Else, we add (j,i) with $u_{ji} = 0$.

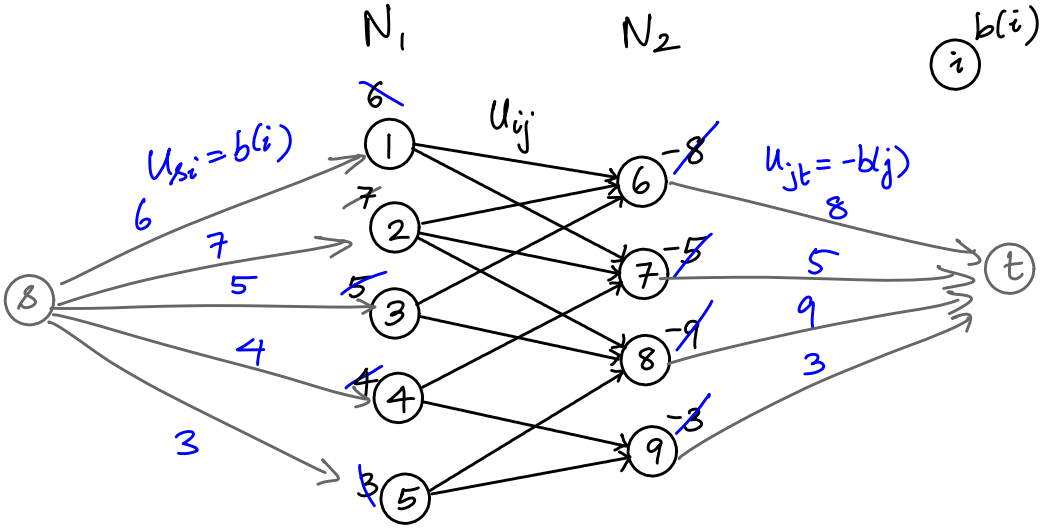
We will see soon that most algorithms for max flow try to push flow "up" an arc, and possibly pull back some of the flow. Hence we assume (i,j) & $(j,i) \in A$ in "preparation", so to speak.

5. There are no parallel (forward) arcs. If such arcs are present, use extra node(s) to "split" all but one of them.



Applications of Max Flow (Read AMO Chapter 6 Section 3!)

Feasibility problem: Find a feasible flow in $G=(N,A)$ with $b(i)$'s (with $\sum b(i)=0$) and u_{ij}'s. For instance, consider the transportation problem. The question is to find if there is a way to ship from supply nodes to demand nodes to satisfy all demands. This is the feasibility version, before the costs are considered. We formulate this problem as a max flow problem. Here is an example.



Add nodes s,t, arcs (s,i) with $u_{si}=b(i)$ $\forall i\in N_1$, (j,t) with $u_{jt}=-b(j)$, $j\in N_2$. We then solve the max flow problem. If the max flow saturates all arcs (s,i), $i\in N_1$ (equivalently, all arcs (j,t), $j\in N_2$), then there exists a feasible flow. The x_{ij} values for $(i,j)\in A$ in the input network give a feasible flow.