MATH 566: Lecture 13 (10/01/2024)

Today: * label correcting algo
 — poly time implementation
* practice midterm

Complexity of the modified label correcting algorithm

* $d(j)$ is updated at most $2nC$ times. $\rightarrow -nC \le d(j) \le nC$
* After update of $d(j)$, j is added to LIST, and arcs in $A(j)$ are scanned afterward.
* Total # arc scans $= \sum_{j \in N}' 2nC \underbrace{|A(j)|}_{\text{\# outarcs of } j}$.

$\Rightarrow$ Modified label correcting algorithm runs in $O(mnC)$ time. $\rightarrow$ pseudo polynomial time algorithm

We have come from an exponential time algo ($O(2^n)$) to a pseudopolynomial time algo (pseudopolynomial due to the dependence on C, rather than $\log C$). But we now present an implementation that runs in polynomial time — $O(mn)$!

FIFO Implementation of the label correcting algorithm

* "Pass": Scan all arcs in A, update $d(j)$ if $d(j) > d(i) + c_{ij}$.
* Do n passes, or stop if no $d(j)$'s change in a pass (whichever comes first).
* Maintain LIST as a FIFO queue.

In fact, we don't have to examine all arcs in A in each pass! We can take a node i from top/front of the LIST, and examine only its outarcs, i.e., the arcs in $A\{i\}$. If we update $d(j)$ for any arc $(i,j) \in A\{i\}$ and $j \notin$ LIST, then we add j to the back of the LIST. See AMO Fig 5.5 (in pg 141) for pseudocode.

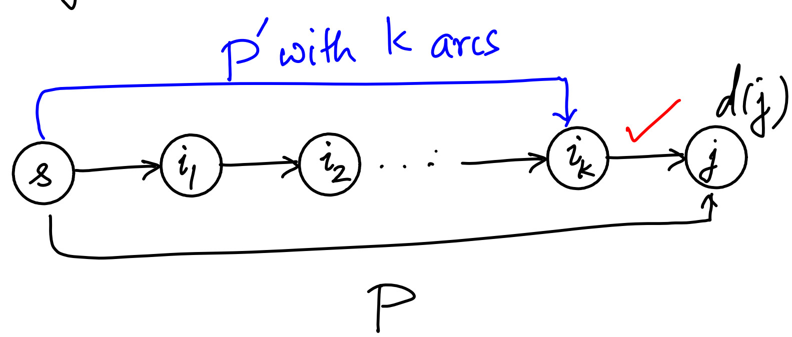
AMO Theorem 5.3 The FIFO label correcting algorithm solves SP in $O(mn)$ time, or else shows that there is a negative cycle.

Proof Each pass : $O(m)$ time (at most m updates).

Need to show the algo will run correctly in n passes.

Claim At the end of the k^{th} pass, the algorithm computes SP distances for all nodes whose SP from s has k or fewer arcs.

We use induction to prove this claim. Assume it holds for k passes. Consider node j.



P' is an SP from s to i_k with k arcs. By induction assumption, after k passes, we will have

$$d(i_k) = \sum_{(i',j') \in P'} c_{i'j'}.$$

In the $(k+1)$-st pass, we update $d(j)$ to $d(i_k) + c_{i_k,j}$. Hence $d(j)$ will be the SP distance from s to j with $k+1$ arcs.

$\Rightarrow$ The claim holds.

Note that if $d(j)$ is not updated in the $(k+1)$-st pass, node j will (trivially) be not a candidate to consider here.

Since any SP from s to j has at most $(n-1)$ arcs, the $d(j)$'s are optimal after $(n-1)$ passes. But if there is an update in the n^{th} pass, there exists a negative cycle. Else, the SP problem is solved.

In the modified label correcting algorithm, if we maintain LIST as FIFO queue, we get the $O(mn)$ running time. □

If we maintain LIST as a LIFO queue, it might work better in practice, especially on large instances. But we need the FIFO handling to prove the $O(mn)$ running time.

Detecting Negative Cycles

1. Stop if $d(j) \leq -nC$.
2. Run the FIFO label correcting algorithm, and stop if a node is scanned at least n times.
3. Keep track of the # arcs in an SP from s to j, $j \in N$, and stop if any SP has more than $(n-1)$ arcs.

Review for Midterm

For Problems 1–3, just eyeball solutions! No proof of correctness is required, nor are any evidence of use of any algorithms.

6. FALSE. $O(\cdot)$ does not hold, e.g., $f(n)=n^2$, $g(n)=n$.

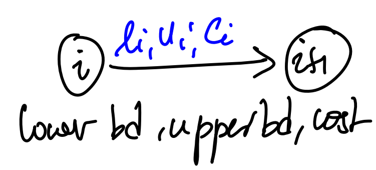
8. $y_i \in [0, v_i] \implies 0 \le y_i \le v_i$

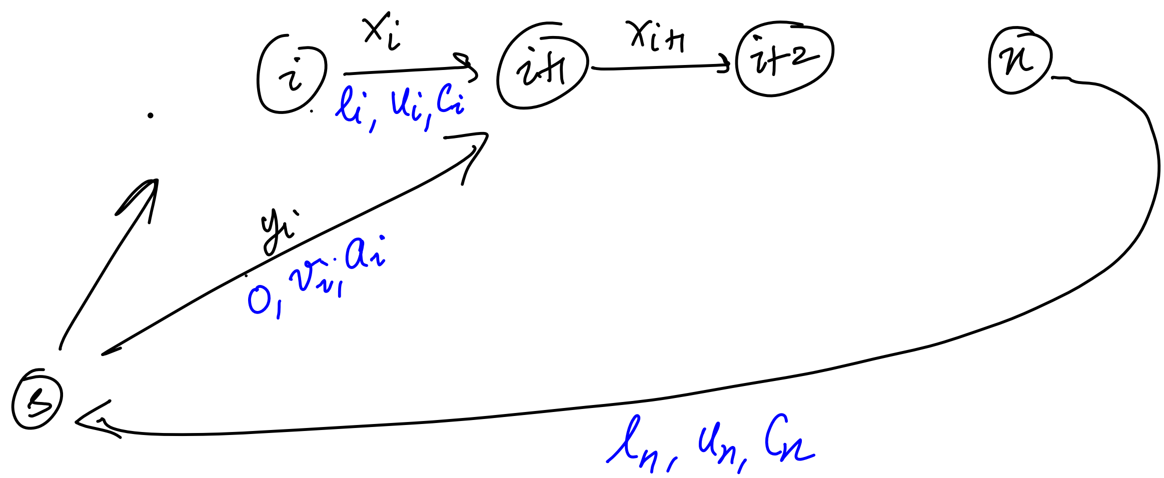
lower bound   upper bound

$x_i \in [l_i, u_i] \implies l_i \le x_i \le u_i$

All #s (v_i, l_i, u_i)'s suggest lb/ub for arcs. Then there are costs c_{i+1}, a_i. But no supply/demand!

$\implies$ Circulation model ✓!

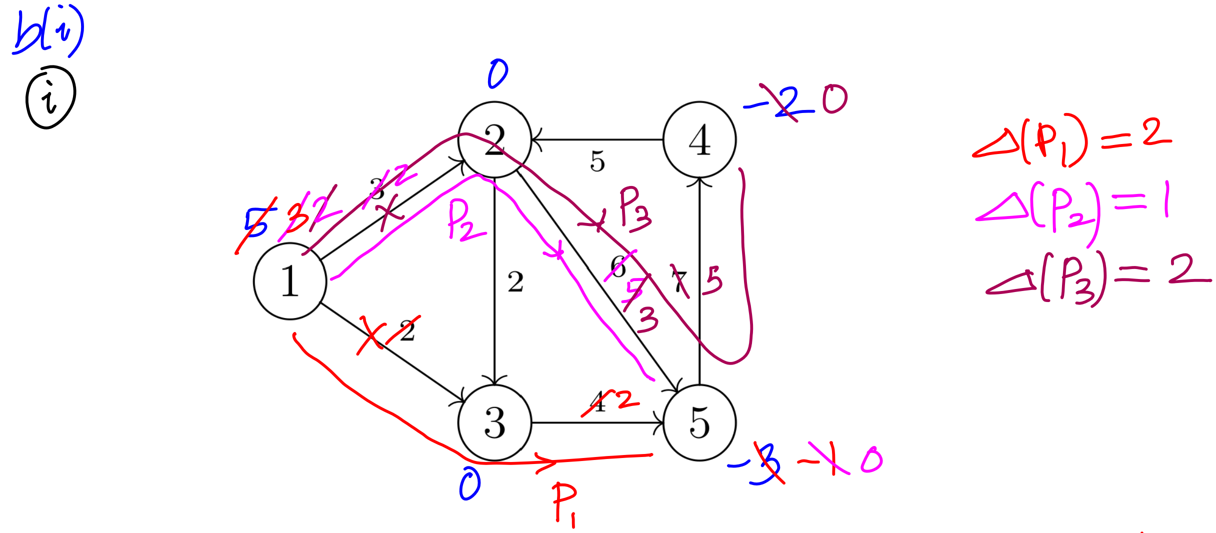




See Solutions for details!

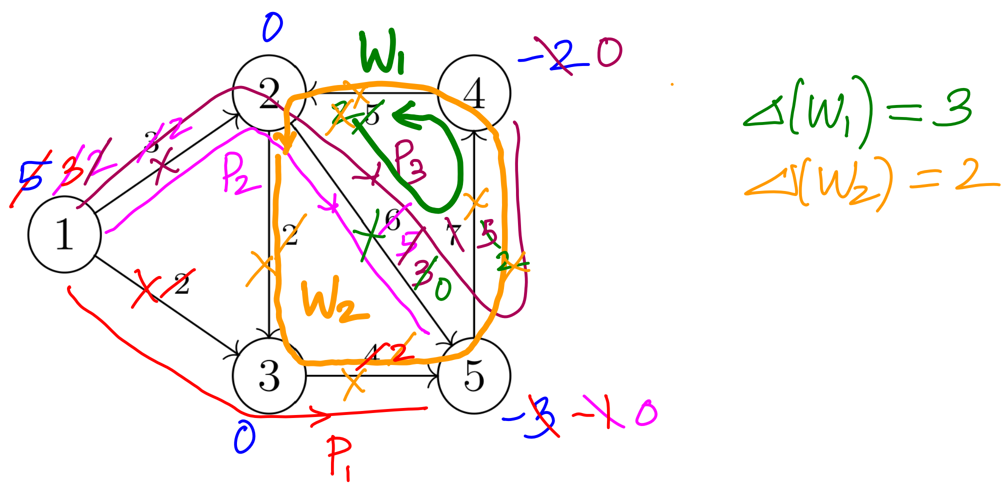
2. Flow decomposition problem

Depending on the order in which you choose paths and cycles, you can get different decompositions.



Identify the Supply and Demand subsets of nodes
Here, $\mathcal{S} = \{1\}$ and $\mathcal{D} = \{4, 5\}$.

If we identify paths first, one option is to get P_1, P_2, P_3 as shown.



We could then identify cycles W_1 and W_2 as shown.
Naturally, alternative choices exist!