Network Flows

3. Shortest Path Problems

3.5 Label-Correcting Algorithm

Fall 2010

Instructor: Dr. Masoud Yaghini
Optimality Conditions
Optimality Conditions

- **Shortest Path Optimality Conditions**
  - let \( d(j) \) denote the length of some directed path from the source node to node \( j \), for every node \( j \in N \),
  - the numbers \( d(j) \) represent shortest path distances if and only if they satisfy the following shortest path optimality conditions:

\[
d(j) \leq d(i) + c_{ij} \quad \text{for all } (i, j) \in A.
\]
Optimality Conditions

\[ d(j) \leq d(i) + c_{ij} \quad \text{for all } (i, j) \in A. \]

- These inequalities state that for every arc \((i, j)\) in the network, the length of the shortest path to node \(j\) is no greater than the length of the shortest path to node \(i\) plus the length of the arc \((i, j)\).

- For, if not, some arc \((i, j) \in A\) must satisfy the condition \(d(j) > d(i) + c_{ij}\)

- in this case, we could improve the length of the shortest path to node \(j\) by passing through node \(i\), thereby contradicting the optimality of distance labels \(d(j)\).
Optimality Conditions

- Let \( s = i_1 - i_2 - \ldots - i_k = j \) be any directed path \( P \) from the source to node \( j \).

\[
\begin{align*}
    d(j) &= d(i_k) \leq d(i_{k-1}) + c_{i_{k-1}i_k}, \\
    d(i_{k-1}) &\leq d(i_{k-2}) + c_{i_{k-2}i_{k-1}}, \\
    &\vdots \\
    d(i_2) &\leq d(i_1) + c_{i_1i_2} = c_{i_1i_2}.
\end{align*}
\]

- Adding these inequalities, we find that

\[
d(j) = d(i_k) \leq c_{i_{k-1}i_k} + c_{i_{k-2}i_{k-1}} + c_{i_{k-3}i_{k-2}} + \ldots + c_{i_1i_2} = \sum_{(i,j)\in P} c_{ij}.
\]
Optimality Conditions

- Thus $d(j)$ is a **lower bound** on the length of any directed path from the source to node $j$.
- Since $d(j)$ is the length of some directed path from the source to node $j$, it also is an **upper bound** on the shortest path length.
- Therefore, $d(j)$ is the shortest path length, and we have established the following result.
Generic Label-Correcting Algorithms
Generic Label-Correcting Algorithms

- **The generic label-correcting algorithm**
  - negative costs permitted
  - no negative cycle
  - maintains a set of distance labels $d(.)$ at every stage.

- The label $d(j)$
  - either $\infty$, indicating that we have yet to discover a directed path from the source to node $j$,
  - or it is the length of some directed path from the source to node $j$.

- The predecessor index
  - $pred(j)$, which records the node prior to node $j$ in the current directed path of length $d(j)$. 
At termination, the predecessor indices allow us to trace the shortest path from the source node back to node $j$. 
Generic Label-Correcting Algorithms

\begin{algorithm}
\textbf{algorithm} label-correcting;
\begin{algorithmic}
\State $d(s) := 0$ and $\text{pred}(s) := 0$;
\State $d(j) := \infty$ for each $j \in N - \{s\}$;
\While {some arc $(i, j)$ satisfies $d(j) > d(i) + c_{ij}$}
\begin{algorithmic}
\State $d(j) := d(i) + c_{ij}$;
\State $\text{pred}(j) := i$;
\End;
\End;
\end{algorithmic}
\end{algorithmic}
\end{algorithm}

- <Animation>
Modified Label-Correcting Algorithms
The generic label-correcting algorithm does not specify any method for selecting an arc violating the optimality condition.

One obvious approach is to scan the arc list sequentially and identify any arc violating this condition.

This procedure is very time consuming because it requires $O(m)$ time per iteration.
Modified Label-Correcting Algorithms

- **Modified Label-Correcting Algorithms**
  - an improved approach that reduces the workload to an average of $O(m/n)$ time per iteration.

- Suppose that we maintain a list, LIST, of all arcs that might violate their optimality conditions.

- If LIST is empty, clearly we have an optimal solution.
  - Otherwise, we examine this list to select an arc, say $(i, j)$, violating its optimality condition.

- We remove arc $(i, j)$ from LIST, and if this arc violates its optimality condition we use it to update the distance label of node $j$. 
Modified Label-Correcting Algorithms

- Any decrease in the distance label of node $j$ decreases the reduced lengths of all arcs emanating from node $j$ and some of these arcs might violate the optimality condition.
- Also notice that decreasing $d(j)$ maintains the optimality condition for all incoming arcs at node $j$.
- Therefore, if $d(j)$ decreases, we must add arcs in $A(j)$ to the set LIST.
- Next, observe that whenever we add arcs to LIST, we add all arcs emanating from a single node (whose distance label decreases).
Modified Label-Correcting Algorithms

algorithm modified label-correcting;
begin
    $d(s) := 0$ and $\text{pred}(s) := 0$;
    $d(j) := \infty$ for each node $j \in N - \{s\}$;
    $\text{LIST} := \{s\}$;
    while $\text{LIST} \neq \emptyset$ do
        begin
            remove an element $i$ from $\text{LIST}$;
            for each arc $(i, j) \in A(i)$ do
                if $d(j) > d(i) + c_{ij}$ then
                    begin
                        $d(j) := d(i) + c_{ij}$;
                        $\text{pred}(j) := i$;
                        if $j \notin \text{LIST}$ then add node $j$ to $\text{LIST}$;
                    end;
        end;
    end;
end;
Modified Label-Correcting Algorithms

- This suggests that instead of maintaining a list of all arcs that might violate their optimality conditions, we may maintain a list of nodes with the property that if an arc \((i, j)\) violates the optimality condition, \(\text{LIST}\) must contain node \(i\).

- Maintaining a node list rather than the arc list requires less work and leads to faster algorithms in practice.

- This is the essential idea behind the modified label-correcting algorithm

\(<\text{Animation}>\)
The End