Graphs and Network Flows
IE411

Lecture 11

Dr. Ted Ralphs
References for Today’s Lecture

- Required reading
  - Sections 21.3

- References
  - AMO Chapter 5
  - CLRS Chapter 25
Label-Correcting Algorithms

• Generic
  – $O(n^2C)$ iterations (recall $d(j)$ bounded by $nC$ and $-nC$)
  – No specified method for selecting an arc violating optimality conditions

• Modified
  – By repeatedly scanning arcs in a fixed order, we can get a strongly polynomial time algorithm.
  – **Practical improvement**: Maintain a list of arcs that *might* violate optimality conditions
    ✴ If we decrease $d(j)$, what do we know about reduced lengths of incoming arcs? outgoing arcs?
    ✴ Which arcs could violate optimality conditions after a label is modified?
Special Implementations of Modified Label-Correcting

• FIFO Label-Correcting
  – $O(mn)$ is best strongly polynomial-time implementation
  – Maintain a queue and examine nodes in FIFO order

• Dequeue Implementation
  – a `dequeue` allows elements to be added or deleted from both front and back
  – always select nodes from front; add previously seen nodes to front, all others to back
  – $O(nmC)$ but performs well in practice for sparse networks
FIFO Label-Correcting Algorithm

Input: A network $G = (N, A)$ and a vector of arc lengths $c \in \mathbb{Z}^A$

Output: $d(i)$ is the length of a shortest path from node $s$ to node $i$ and $\text{pred}(i)$ is the immediate predecessor of $i$ in an associated shortest paths tree.

$d(s) \leftarrow 0$ and $\text{pred}(s) \leftarrow 0$

$d(j) \leftarrow \infty$ for each $j \in N \setminus \{s\}$

$Q \leftarrow \{s\}$

while $Q \neq \emptyset$ do

  Remove the first element $i$ from $Q$

  for $(i, j) \in A(i)$ do

    if $d(j) > d(i) + c_{ij}$ then

      $d(j) \leftarrow d(i) + c_{ij}$

      $\text{pred}(j) \leftarrow i$

      if $j \notin Q$ then

        add $j$ to the end of $Q$

      end if

    end if

  end for

end while
All-Pairs Shortest Path Problem

• Determine the shortest path distance between every pair of nodes in the network.
  – Assume underlying network is *strongly connected*
  – Assume network does not contain a negative cost cycle

• Algorithms
  – Repeated Shortest Path
  – All-Pairs Label-Correcting
Repeated Shortest Path Algorithm (Non-Negative Arc Lengths)

• For each node $i \in N$, solve a single-source shortest path problem with node $i$ as the source using any appropriate algorithm.

• Complexity: Let $S(n, m, C)$ denote the time required to solve a shortest path problem with non-negative arc lengths. Then, the complexity is $O(n \cdot S(n, m, C))$. 

Repeated Shortest Path Algorithm (Negative Arc Lengths)

- Transform the network into one with non-negative arc lengths.
- For each node $i \in N$, solve a single-source shortest path problem with node $i$ as the source using any appropriate algorithm.
- Compute the shortest path distances in the original network from the shortest path distances in the transformed network.
- **Complexity**: $O(nm + n \cdot S(n, m, C)) = O(n \cdot S(n, m, C))$. 
Shortest Path Optimality Conditions

**Theorem 1.** For every pair of nodes \([i, j] \in N \times N\), let \(d[i, j]\) represent the length of some directed path from node \(i\) to node \(j\) satisfying \(d[i, i] = 0 \ \forall i \in N\) and \(d[i, j] \leq c_{ij} \ \forall (i, j) \in A\). These distances represent shortest path distances if and only if they satisfy

\[ d[i, j] \leq d[i, k] + d[k, j] \ \forall i, j, k \in N. \]

**PROOF:**

⇒ If these distances represent shortest path distances, they satisfy \(d[i, j] \leq d[i, k] + d[k, j] \ \forall i, j, k \in N\).

⇌ If a set of distance labels satisfy \(d[i, j] \leq d[i, k] + d[k, j] \ \forall i, j, k \in N\), then they represent shortest path distances.


All-Pairs Label-Correcting Algorithm

Input: A network \( G = (N, A) \) and a vector of arc lengths \( c \in \mathbb{Z}^A \)
Output: \( d[i, j] \) is the length of a shortest path from node \( i \) to node \( j \) for pairs \( i \) and \( j \).

\[
\begin{align*}
&d[i, j] \leftarrow \infty \text{ for all } [i, j] \in N \times N \\
&d[i, j] \leftarrow 0 \text{ for all } i \in N \\
&\text{for } (i, j) \in A \text{ do} \\
&\quad d[i, j] \leftarrow c_{ij} \\
&\text{end for} \\
&\text{while } \exists (i, j, k) \text{ satisfying } d[i, j] > d[i, k] + d[k, j] \text{ do} \\
&\quad d[i, j] := d[i, k] + d[k, j] \\
&\text{end while}
\end{align*}
\]
Floyd-Warshall Algorithm

- $O(n^3C)$ iteration complexity of algorithm is not appealing(!)
- Given matrix of distances $d[i,j]$, we need to perform $n^3$ comparisons just to test optimality
- Floyd-Warshall cleverly obtains matrix of shortest path distances within $O(n^3)$ computations
Floyd-Warshall Algorithm

Input: A network $G = (N, A)$ and a vector of arc lengths $c \in \mathbb{Z}^A$

Output: $d[i, j]$ is the length of a shortest path from node $i$ to node $j$ for pairs $i$ and $j$.

for $(i, j) \in N \times N$ do
    $d[i, j] \leftarrow \infty$ and $pred[i, j] \leftarrow 0$
end for

for $i \in N$ do
    $d[i, i] \leftarrow 0$
end for

for $(i, j) \in A$ do
    $d[i, j] \leftarrow c_{ij}$ and $pred[i, j] := i$
end for

for $k = 1$ to $n$ do
    for $[i, j] \in N \times N$ do
        if $d[i, j] > d[i, k] + d[k, j]$ then
            $d[i, j] \leftarrow d[i, k] + d[k, j]$
            $pred[i, j] \leftarrow pred[k, j]$
        end if
    end for
end for
Proof of Correctness

Claim 1. After iteration $k$, $d[i, j]$ is the shortest path distance from node $i$ to node $j$ subject to the condition that the path uses only nodes $1, 2, \ldots, k$ as internal nodes.

PROOF: (by induction)
Floyd-Warshall Algorithm

• Complexity?
Detecting Negative Cost Cycles

• Network contains negative cost cycle if
  – $d[i, i] < 0$ for some $i \in N$
  – $d[i, j] < -nC$ for some $[i, j] \in N \times N$

• For F-W, simply check $d[i, i] < 0$ when updating $d[i, i]$.

• How else could we check?