Graphs and Network Flows
IE411

Lecture 15

Dr. Ted Ralphs

Preflow-Push Algorithms

- First developed by A. V. Goldberg in 1985.
- Best preflow-push algorithms outperform best augmenting path algorithms in theory and in practice.
- *Push* flows on individual arcs instead of augmenting paths.
- Do not satisfy mass balance constraints at intermediate stages.
- Work towards *feasibility*.
- Can be seen as a generalization of SAP (SAP is a special case).

Definitions

Distance Labels Valid distance labels d with respect to a flow x satisfy $d(t) = 0$ and $d(i) \leq d(j) + 1 \ \forall (i,j) \in G(x)$

Admissible Arc An arc $(i,j) \in G(x)$ is admissible if $d(i) = d(j) + 1$

Preflow Preflow x in a network is defined as one which

1. Satisfies the capacity constraints $0 \leq x_{ij} \leq u_{ij} \ \forall (i,j)$
2. Has incoming flow at any node other than the source at least as large as outgoing flow

$$\sum_{j:(j,i)\in A} x_{ji} - \sum_{j:(i,j)\in A} x_{ij} \geq 0 \ \forall i \in N - \{s\}$$

Definitions (con't)

Excess Excess of a node i, denoted $e(i)$, is with respect to a flow x is

$$e(i) = \sum_{j:(j,i)\in A} x_{ji} - \sum_{j:(i,j)\in A} x_{ij}$$

Active Node A node $i \neq t$ is called active if $e(i) > 0$

Basic Idea of Preflow-Push Algorithm

- Select an active node i
- Try to remove the excess $e(i)$ by pushing flow on admissible arcs (push flow to neighbors of i that are closer to t as measured by d)
- If active node i has no admissible arcs, increase its distance label
- Terminate when there are no active nodes

Generic Preflow-Push Algorithm

algorithm *preflow-push*
begin
  preprocess
  while the network contains an active node **do**
    select an active node i
    push/relabel(i)
end

Algorithm Details

procedure *preprocess*

$x := 0$

compute exact distance labels $d(i)$ $\forall i \in N$

$x_{sj} := u_{sj}$ for each arc $(s, j) \in A(s)$

$d(s) := n$

end

procedure *push/relabel*(i)

if network contains an admissible arc (i, j) **then**

push $\delta := \min\{e(i), r_{ij}\}$ units from i to j

else $d(i) := \min\{d(j) + 1 : (i, j) \in A(i), r_{ij} > 0\}$

end

Example of Preflow-Push Algorithm

Correctness of Preflow-Push

Claim 1. *The generic preflow-push algorithm correctly computes a maximum flow.*

Proof:

1. The algorithm maintains valid distance labels at each step.
2. There is never a directed path from s to t in the residual network, so as soon as feasibility is attained, we are done.

Complexity of Preflow-Push

Lemma 1. [7.11] *Every node with excess is connected by a path in the residual network to the source.*

Lemma 2. [7.12] *The node labels never get larger than* $2n$

Lemma 3. [7.13] *Each label increases at most* $2n$ *times.*

Lemma 4. [7.14] *The algorithm performs at most* nm *saturating pushes.*

Lemma 5. [7.15] *The algorithm performs at most* n^2m *nonsaturating pushes.*

Complexity of Preflow-Push

Claim 2. *The generic preflow-push algorithm runs in $O(n^2m)$ time.*

Proof:

Relabel The total number of relabel operations is at most $2n^2$.

Saturating Pushes The algorithm performs at most nm saturating pushes.

Non-Saturating Pushes The algorithm performs at most $O(n^2m)$ non-saturating pushes.

Therefore, the overall complexity of the generic preflow-push algorithm is $O(n^2m)$.

Improving Empirical Performance

- Typically, the algorithm establishes a *maximum preflow* before it establishes a maximum flow
- Subsequent push/relabel operations increase distance labels of active nodes until they are larger than n
- Alternate idea: maintain a set N' of nodes that satisfy property that $G(x)$ contains no path from a node in N' to the sink
- Do not perform push/relabel operations for nodes in N' and terminate when all nodes in $N - N'$ are inactive.
- How do we add nodes to N'?

Adding Nodes to N'

- Add j to N' when $d(j) \geq n$
- "Occasionally" perform a reverse breadth-first search of $G(x)$ to obtain exact distance labels
  - Add to N' nodes without a directed path to the sink
  - How often should this be done?
- Keep track of number of nodes with distance label k in $numb[k]$
  - When $numb[k'] = 0$ for some k', any node with $d(j) > k'$ is disconnected from set of nodes with $d(i) < k'$ in $G(x)$
  - Add any node j with $d(j) > k'$ to N'

Specific Implementations

By specifying different rules for selecting active nodes for push/relabel, we can derive different algorithms, each with different worst-case complexity.

FIFO : examine active nodes in FIFO order ($O(n^3)$)

Highest-Label : always push from an active node with highest value of distance label ($O(n^2\sqrt{m})$)

Excess Scaling : push flow from node with sufficiently large excess to node with sufficiently small excess ($O(nm + n^2 log\ U)$)

Summary of Maximum Flow Algorithms

Labeling Algorithm	$O(nmU)$
Capacity Scaling Algorithm	$O(nmlog\ U)$
Generic Preflow-Push Algorithm	$O(n^2m)$
FIFO Preflow-Push Algorithm	$O(n^3)$
Highest-Label Preflow-Push Algorithm	$O(n^2\sqrt{m})$
Excess Scaling Algorithm	$O(nm + n^2log\ U)$