Washington University in St. Louis

Engineering

Max Flow Problem

Preflow-Push Method

Jon Turner
Computer Science & Engineering
Washington University

www.arl.wustl.edu/~jst

Distance Labels

- A set of *distance labels* $d(u)$ is *valid* if $d(t)=0$, and $d(u) \le d(v)+1$, for every edge (u,v) with $res(u,v)>0$
  - »distance labels are said to be *exact* if $d(u)$ equals the length of a shortest path from u to t, for all u
  - »an edge (u,v) is *admissible* with respect to distance labels d if $d(u)=d(v)+1$ and $res(u,v)>0$ (path is *admissible* if its edges are)
- Properties of distance functions
  - »$d(u) \le$ length of shortest path from u to t with residual capacity>0
  - »if $d(s) \ge n$, there is no augmenting path from s to t
  - »an admissible s-t path is a shortest augmenting path
    - can use distance labels to construct an algorithm similar to Dinic's algorithm – no explicit phases
- The *preflow-push* algorithms uses distance labels to decide which edges to add flow to in a *preflow*

2

General Preflow-Push Method

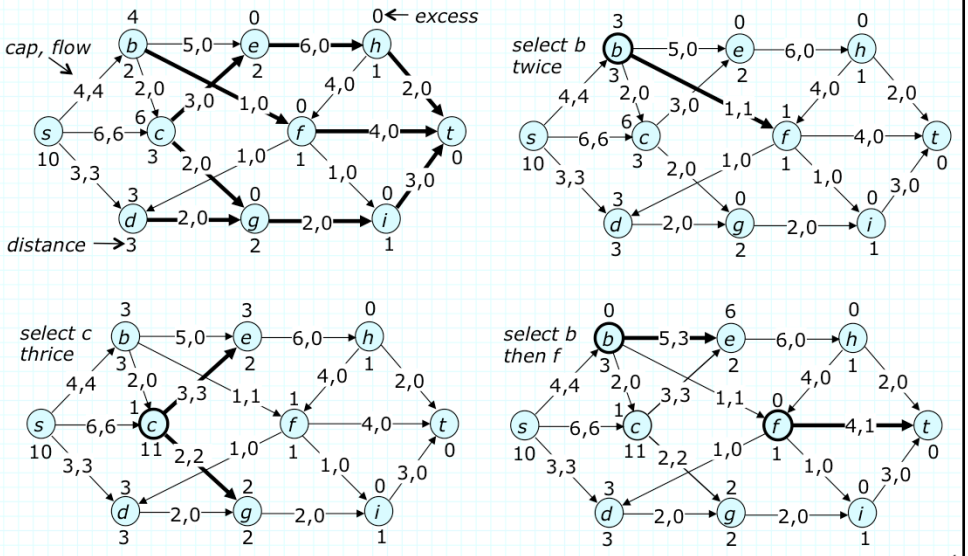
- A *preflow f* is a flow function in which we allow the flow conservation property to be violated
  - » for each vertex v, $\Delta f(v)$ is defined as the *excess* flow into v; a vertex v is *balanced* if $\Delta f(v) = 0$
- The preflow-push method starts with a preflow and converts it to a valid flow function with maximum value
- Start by letting $d(u)$=length of a shortest path from u to t for all $u \in V-\{s\}$ and $d(s) = n$; let $f(s,v)=cap(s,v)$ for all $(s,v) \in E$ and then repeat the following step
  - »select an unbalanced vertex u; if there is an admissible edge (u,v), push flow on (u,v) until it becomes saturated or u becomes balanced; otherwise, increase $d(u)$ enough to create at least one admissible edge
- Note: d is always a valid distance function and that on termination, the preflow is a valid flow

3

Washington University in St.Louis

Engineering

Example of Preflow-Push



4

Finding Admissible Edges

- The basic step requires identifying an admissible edge in the residual graph incident to the selected vertex
- Maintain a *nextedge* pointer for each vertex
  - »*nextedge*(*u*) is initialized to first edge in *u*'s adjacency list whenever *u* is relabeled
  - »find an admissible edge by scanning adjacency list, starting at *nextedge*(*u*)
  - »*nextedge*(*u*) is advanced to the admissible edge found in scan
- So, one pass thru adjacency list for each relabeling of *u*
  - »if *u* has m_u incident edges, spend $O(m_u)$ time looking for admissible edges at *u* between successive relablelings of *u*
  - »if *u* is relabeled $\leq k$ times, spend $O(km_u)$ time finding admissible edges at *u*
  - »if every vertex is relabeled at most *k* times, we spend a total of $O(km)$ time finding admissible edges

5

Analysis of General Method

- At each step, there is a path from u to s with residual capacity>0, for all unbalanced vertices $u \in V-\{s,t\}$
  - » can be proved by induction on the number of steps
- Whenever a vertex u is relabeled, it is unbalanced, and so there is a path from u to s with residual capacity>0
  - » by validity of labels, then $d(u)<2n$ for all u
- Consequently, the number of relabelings is $<2n^2$ and
  - » the time spent finding admissible edges is $O(mn)$
  - » the number of steps that saturate an edge is $O(mn)$
  - » the time spent relabeling vertices is $O(mn)$
- Steps that add flow but do not saturate edges cannot be bounded by time spent finding admissible edges
  - » these steps do not make an edge inadmissible & don't advance the *nextedge* pointer

6

- *Lemma*. The number of steps that add flow and don't saturate edges is $O(mn^2)$

  Proof. Define *potential function P*, to be sum of distance labels of all unbalanced vertices; $P<2n^2$ initially and zero on termination

  Case analysis for one step – let u be the selected vertex
  - » steps that add flow to (u,v) without saturating it – these steps make u balanced and may make v unbalanced, so P decreases by $\geq d(u)-d(v)=1$, since (u,v) is admissible; number of these steps is $<2n^2$ plus increase in P from other steps
  - » steps that relabel u – these steps increase P, but since distance labels never decrease and are $<2n$, the total increase from all steps is $\leq 2n^2$
  - » steps that saturate an edge (u,v) – these may make v unbalanced, increasing P by $d(v)$; since there are $O(mn)$ steps that saturate an edge and $d(v)<2n$, the total increase in P from these steps is $O(mn^2)$

  So, number that add flow without saturating edge is $O(mn^2)$ ■
- Running time of preflow-push method is $O(mn^2)$

7

The FIFO Algorithm

- Different rules for selecting an unbalanced vertex produce different variants of the preflow push method
  - » one common feature of most variants is that they repeatedly select same vertex until it is either balanced or is relabeled
- The FIFO preflow-push algorithm selects the next unbalanced vertex from the front of a queue
  - » whenever a vertex becomes unbalanced or is relabeled, it is put at the end of the queue
- For analysis, divide the execution into phases
  - » each phase ends when the vertices on the queue at the start of the phase have been selected
- Each phase contains at most n steps that add flow to an edge without saturating it
  - » so, if number of phases is $O(n^2)$, the total running time is $O(n^3)$

- Since number of relabelings is $<2n^2$, the number of phases that include at least one relabeling op is $<2n^2$
- To bound number of phases that do not relabel a vertex, define a potential function, P equal to the largest label among the unbalanced vertices
  - » $P<2n$ initially and $P=0$ when the algorithm terminates
- In every phase with no relabeling op, vertices that were unbalanced at start of phase become balanced
  - » since the nodes they push their flow to have smaller distance labels, P must decrease by ≥ 1 during such a phase
  - » so, number of phases with no relabeling op is $\leq 2n$ plus the increase in potential during the phases with a relabeling op
- In phases with a relableling op, only steps that relabel a vertex can increase P
  - » since labels never decrease and are bounded by $2n$, total increase in potential during these phases is $<2n^2$
- So, number of phases is $O(n^2)$ and total time is $O(n^3)$

9

Batch Relabeling

- The relabeling procedure can cause the algorithm to waste a lot of time in common situations
  - » removing excess flow from a long dead-end
  - » avoid by relabeling in batches, rather than incrementally
  - » allows excess flow to drain quickly from dead-ends
- Procedure:
  - » skip the incremental relabeling step
  - » do not add more flow into a vertex *u* if *nextedge(u)* is *null* (at end of list)
  - » when all unbalanced vertices have *nextedge=null*, re-compute exact vertex labels
    - compute shortest paths in res. graph to *t* & shortest paths to *s*
- This does not improve worst-case performance, but can significantly improve typical performance

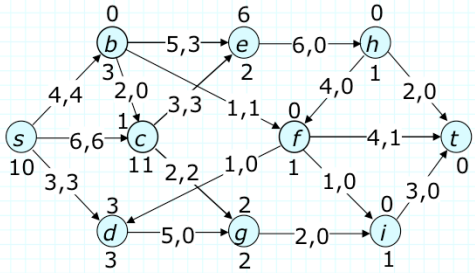
Highest Label Preflow Push

- Selects a new vertex to operate on by selecting the unbalanced vertex with the largest label
- A heap can be used to find the desired vertex, but since labels are bounded by $2n$, we don't really need heap
  - » maintain an array of lists of unbalanced vertices: a vertex is placed in the i-th list, if its distance label is i
  - » maintain a variable *top* which points to the list of vertices with maximum distance label; this may require increasing *top* when a vertex is relabeled or decreasing *top* by scanning down the array when a vertex becomes balanced
  - » since total increase in top is $2n^2$, the total decrease is $O(n^2)$, so total time to maintain top is $O(n^2)$
- Can be shown that overall running time is $O(m^{1/2}n^2)$

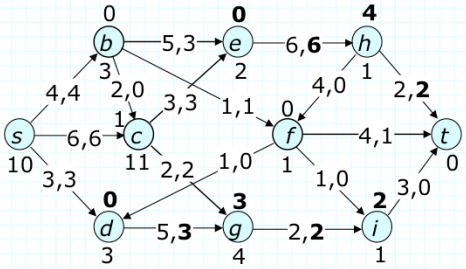
11

Exercises

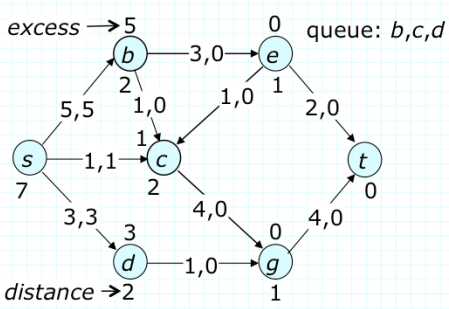
1. The diagram below shows an intermediate state in the execution of the general preflow-push algorithm. The excess appears above each node and the distance label appears below.



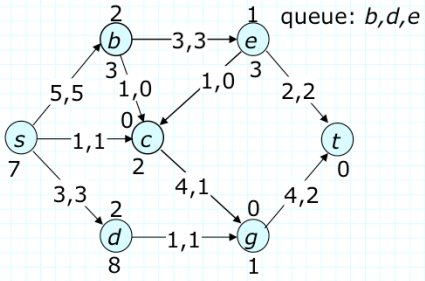
Show the state after nodes *d*, *e*, *h* and then *g* are selected.



2. The diagram below shows the initial state in the execution of the fifo preflow-push method.



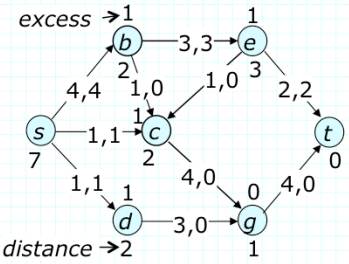
Assuming that batch relabeling is used, show the state right after the first batch relabeling step.



What are the distance labels after the next batch relabeling step?

At that point, b, c and d have labels of 8 and e and g have labels of 9.

3. The diagram below shows an intermediate state in the execution of the preflow-push method. What is the value of the potential function used in the analysis, in this state? How does the potential function change following a single step at node *d*?



The potential function is the sum of the distance values at the unbalanced vertices (b,c,d,e), so the value is 9. A step at node d will push one unit of flow to g, making d balanced and g unbalanced. This reduces the potential function by 1.

Suppose we view this as a state in the execution of the fifo version of the preflow push algorithm. What is the value of the potential function used in the analysis of the fifo version? How does the value change if we perform a single step at vertex *b?*

A step at vertex b would re-label node b, changing its distance label from 2 to 3. This does not change the potential, as vertex e already has a distance label of 3.

4. Suppose we apply the highest label preflow push algorithm to a path with n vertices, where all edges except the last have capacity 2 and the last one has capacity 1. Show that the number of steps taken by the preflow-push algorithm on this network is $W(n^2)$, if incremental relabeling is used.

   At the start of the algorithm the "inner" vertices have labels of n–2,...,2,1 giving a distance label sum of (n–1)(n–2)/2. The excess unit of flow will eventually be pushed back from the "penultimate" vertex to the vertex adjacent to the source. This happens only after all the labels of the inner vertices are at least n–2, giving a distance label sum at least $(n-2)^2$.

   The relabeling operations leading up to this point never increase a label by more than 2, Consequently, the number of relabeling steps is about $(n-2)^2/4$, which is $W(n^2)$.

   Show that the total running time is $O(n)$ in this case, if batch relabeling is used.

   In this case it takes about n steps to push 2 units of flow forward to the "penultimate" vertex and 1 unit to the sink. At this point, the penultimate vertex is the only unbalanced vertex and the batch relabeling takes place. After the relabeling, it takes about n more steps to push the excess flow back to the source. The batch relabeling procedure takes O(n) time in this case, so the overall time is O(n).