COMP4600

Advanced Algorithms

Network Flow

Lecture Notes* by

Paulette Lieby

ANU

paulette.lieby@anu.edu.au

*based on *Introduction to Algorithms*, 2nd edition, T.H. Cormen, C.E. Leiserson, R.L. Rivest.

Outline

- Flow networks:
  - what is a flow
  - the max-flow min-cut theorem
- Finding max flows:
  - the Ford-Fulkerson algorithm
    (using augmenting paths)
  - the generic Goldberg algorithm
    (using push relabel techniques)
- Application: maximum matching

Flow Networks

Note that the definition of a flow follows the one given in *Introduction to Algorithms*, **2nd edition**, by Cormen et al.

This definition is **different** from the one given in the **3rd edition** of the same book.

You are free to use any one of these definitions.

Flow Networks

Definition. A flow network $G = (V, E)$ is a directed graph in which each edge $(u, v) \in E$ has non-negative capacity $c(u, v) \geq 0$. By convention, if edge $(u, v) \notin E$, then $c(u, v) = 0$.

- we distinguish two vertices, the source s and the sink t
- we assume that every vertex $v \in V$ lies on a path $s \rightsquigarrow v \rightsquigarrow t$ from the source to the sink
- thus $|E| \geq |V| - 1$

Flow

Definition. Let $G = (V, E)$ be a flow network with source s and sink t and capacity function c. A flow in G is a function $f : V \times V \to R$ that satisfies:

- Capacity constraint: $\forall u, v \in V, f(u, v) \leq c(u, v)$
- Skew symmetry: $\forall u, v \in V, f(u, v) = -f(v, u)$
- Flow conservation: $\forall u \in V - \{s, t\}$,

$$\sum_{v \in V} f(u, v) = 0$$

- note: (u, v) means an edge *from* u *to* v
- the value of a flow f is $F_f = \sum_{v \in V} f(s, v)$
- the maximum flow problem: find a flow of maximum value from s to t
- note: implicit summation:
  $f(X, Y) = \sum_{x \in X} \sum_{y \in Y} f(x, y)$ where X, Y are sets of vertices of V
- also, $c(X, Y) = \sum_{x \in X} \sum_{y \in Y} c(x, y)$

Lemma 1. *Let* $G = (V, E)$ *be a flow network with flow* f. *Then, for* $X, Y, Z \subseteq V$ *with* $X \cap Y = \emptyset$,

$$f(X, X) = 0$$

$$f(X, Y) = -f(Y, X)$$

$$f(X \cup Y, Z) = f(X, Z) + f(Y, Z)$$

$$f(Z, X \cup Y) = f(Z, X) + f(Z, Y)$$

- note: also, $f(u, V) = 0 \ \ \forall u \in V - \{s, t\}$

Residual Network

Definition. Let $G = (V, E)$ be a flow network with flow f.

- Given a pair of vertices (u, v), the residual capacity c_f of (u, v) is given by $c_f(u, v) = c(u, v) - f(u, v)$.
- The residual network induced by f is $G_f = (V, E_f)$ where $E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$.

- question: when do we have $(u, v) \in E_f$ but $(u, v) \notin E$?

Augmenting Paths

Definition. Given a flow network $G = (V, E)$ and a flow f, an augmenting path is a simple path P from s to t in the residual network G_f. The residual capacity of P is given by

$c_f(P) = \min\{c_f(u, v) : (u, v) \in P\}$.

- note: all edges (u, v) in an augmenting path in G_f have $c_f(u, v) > 0$

Cuts

Definition. A cut of a flow network graph (V, E) is a partition of V into S and $T = V - S$ such that $s \in S$ and $t \in T$.

- $f(S, T)$ is the flow across the cut (S, T) and $c(S, T)$ is its capacity

Lemma 2. *Given a flow network $G = (V, E)$, a flow f, and a cut (S, T) of G, we have $f(S, T) = F_f$.*

Proof. Using Lemma 1,

$$\begin{aligned} f(S,T) &= f(S,V) - f(S,S) \\ &= f(S,V) \\ &= f(s,V) + f(S - \{s\}, V) \\ &= f(s,V) \text{ (why is } f(S - \{s\}, V) = 0\text{?)} \\ &= F_f. \end{aligned}$$

Max-flow Min-cut Theorem

Theorem 1. *Let f be a flow in a flow network $G = (V, E)$ with source s and sink t. Then the following conditions are equivalent:*

1. *f is a maximum flow in G.*
2. *Residual network G_f contains no augmenting paths.*
3. *$F_f = c(S, T)$ for some cut (S, T) of G.*

Proof. (1) $\Rightarrow$ (2) Suppose there exists an augmenting path P in G_f with residual capacity $c_f(P)$. Then one may push a flow of value $c_f(P)$ along P to obtain a total flow of value $F_f + c_f(P)$. Contradition.

Proof cont. (2) $\Rightarrow$ (3) Suppose there is no augmenting path in G_f. Let

$S = \{v \in V$: there exists a path from s to v in $G_f\}$.

Then (S, T) (where $T = V \setminus S$) is a cut where for each edge (u, v), $u \in S$ and $v \in T$, $f(u, v) = c(u, v)$ (why?).

This implies that $F_f = f(S, T) = c(S, T)$ (by Lemma 2).

(3) $\Rightarrow$ (1) From Lemma 2 we have $F_f = f(S, T)$, and by definition we have $f(S, T) \leq c(S, T)$ for all cuts (S, T).

If $F_f = c(S, T)$ then the flow f must be maximal. □

The Ford-Fulkerson Algorithm

for each edge $(u, v) \in E$
  $f(u, v) = 0$
  $f(v, u) = 0$
while there exists an augmenting path P in G_f
  $c_f(P) = \min\{c_f(u, v) : (u, v) \in P\}$
  for each edge $(u, v) \in E$ in P
    $f(u, v) \mathrel{+}= c_f(P)$
    $f(v, u) = -f(u, v)$
    update $c_f(u, v)$ and $c_f(v, u)$

The Edmonds-Karp Algorithm

[...]
while there exists a *shortest* augmenting path P
[...]

Important:

- if f is a flow in G before update in Ford-Fulkerson/Edmonds-Karp, then f is a flow afterwards
- capacity constraint, skew symmetry and flow conservation are preserved

The Edmonds-Karp Algorithm: Analysis

Lemma 3. *If the Edmonds-Karp Algorithm is run on a flow network $G = (V, E)$ with source s and sink t then for all vertices $v \in V - \{s, t\}$, the shortest path distance $\delta_f(s, v)$ in G_f increases monotonically with each flow augmentation.*

Proof. Suppose that for some vertex $v \in V - \{s, t\}$ there is a flow augmentation that causes $\delta_f(s, v)$ to decrease. Let f be the flow before augmentation, and f' be the flow after. Then $\delta_{f'}(s, v) < \delta_f(s, v)$.

Also assume WLOG that v is the vertex for which $\delta_{f'}(s, v)$ is smallest among the vertices w where $\delta_{f'}(s, w) < \delta_f(s, w)$. That is,

$$\delta_{f'}(s, u) < \delta_{f'}(s, v) \text{ implies } \delta_{f'}(s, u) \geq \delta_f(s, u) \qquad (1)$$

Proof (cont.). Now consider a shortest path P in $G_{f'}$, $s \rightsquigarrow u \to v$. Then

$$\delta_{f'}(s,u) = \delta_{f'}(s,v) - 1. \tag{2}$$

From (1) we get

$$\delta_{f'}(s,u) \geq \delta_f(s,u). \tag{3}$$

Assume $(u,v) \in E_f$. Then

$$\begin{aligned}\delta_f(s,v) &\leq \delta_f(s,u) + 1 \text{ (why?)} \\ &\leq \delta_{f'}(s,u) + 1 \text{ by (3)} \\ &= \delta_{f'}(s,v).\end{aligned}$$

Contradiction, so $(u,v) \notin E_f$.

Proof cont. Now, $(u,v) \notin E_f$ while $(u,v) \in E_{f'}$: the flow augmentation from f to f' by f'' has increased the flow from v to u ($c_{f'}(u,v) = c_f(u,v) - f''(u,v) > 0$ with $c_f(u,v) = 0$ which implies that $0 < f''(v,u) \leq c_f(v,u)$).

Thus (v,u) is an edge in the shortest path $s \rightsquigarrow v \rightarrow u$ in G_f (because Edmund-Karps always augments flow along shortest paths):

$$\begin{aligned}\delta_f(s,v) &= \delta_f(s,u) - 1 \\ &\leq \delta_{f'}(s,u) - 1 \text{ by (3)} \\ &= \delta_{f'}(s,v) - 2 \text{ by (2).}\end{aligned}$$

This contradicts our assumption $\delta_{f'}(s,v) < \delta_f(s,v)$. □

Theorem 2. *The total number of flow augmentations performed by the Edmonds-Karp algorithm of a flow network* $G = (V, E)$ *is* $O(VE)$.

Definition. Let (u, v) be an edge in an augmenting path P in G_f. We say the edge is critical if $c_f(u, v) = c_f(P)$.

Proof. Let (u, v) be a critical edge on a path P in G_f. We have $\delta_f(s, v) = \delta_f(s, u) + 1$ (because P is a shortest path). (u, v) will disappear from the residual network after the next flow augmentation.

(u, v) can become critical again only if some flow is pushed from v to u, that is when (v, u) is on an augmenting path. Let f' be the flow when this occurs. Then

$$\begin{aligned}\delta_{f'}(s, u) &= \delta_{f'}(s, v) + 1 \\ &\geq \delta_f(s, v) + 1 \text{ by Lemma 3} \\ &= \delta_f(s, u) + 2.\end{aligned}$$

Proof cont. We have $0 \leq \delta(s,u) \leq |V| - 2$ ($\delta(s,u)$ is the distance from s to u), so that an edge (u,v) can become critical at most $|V|/2$ times. There are $O(E)$ pairs of vertices (u,v) that are potential critical edge candidates, thus in total there are $O(VE)$ potential critical edges.

Since each augmenting path has a least one critical edge, there are at most $O(VE)$ flow augmentations. □

Theorem 3. *Edmonds-Karp is* $O(VE^2)$.

Proof. Finding a shortest path in G_f takes $O(E)$ (BFS).

The result follows from Theorem 2. □

Push-Relabel Algorithms

- efficient: $O(V^2E)$ for the generic Goldberg algorithm, which can refined to obtain $O(V^3)$
- work locally, looking only at the vertex's neighbours in the residual network
- do *not* maintain flow conservation
- they maintain a preflow

Preflow

Definition. A preflow is a function $f : V \times V \to R$ that satisfies:

- Capacity constraint: $\forall u, v \in V, f(u, v) \leq c(u, v)$
- Skew symmetry: $\forall u, v \in V, f(u, v) = -f(v, u)$
- Relaxed flow conservation:

$$\forall u \in V - \{s\}, \sum_{v \in V} f(v, u) \geq 0$$

The quantity $\sum_{v \in V} f(v, u)$ is called the excess flow $e(u)$ for vertex u. When $e(u) > 0$ we say that u is overflowing (for $u \in V - \{s, t\}$).

The Idea

- picture vertices as reservoirs and edges as pipes
- raise 'reservoir' s at a certain height
- 'push' water/flow from u to v down the 'pipes' linking u to v
- 'push' water only if $e(u) > 0$ and u is higher than v
- if u is not higher than any of its neighbours v then raise u sufficiently to allow water to flow down
- repeat until all reservoirs except the source are empty

The Height Function

Definition. Let $G = (V, E)$ be a flow network with source s and sink t and preflow f. A function $h : V \to N$ is a height function if $h(s) = |V|$, $h(t) = 0$, and

$h(u) \leq h(v) + 1$

for every residual edge $(u, v) \in G_f$.

Lemma 4. *Let $G = (V, E)$ be a flow network with preflow f and height function h. For any pair (u, v) of vertices of V, if $h(u) > h(v) + 1$ then (u, v) is not a residual edge in G_f.*

The Two Basic Operations

- the push operation on an edge (u, v) applies when
  - $e(u) > 0$
  - $h(u) = h(v) + 1$
  - action: push $d_f(u, v) = \min(e[u], c_f(u, v))$ units of flow from u to v
- the relabel operation on a vertex u applies when
  - u is overflowing: $e(u) > 0$
  - for all $(u, v) \in E_f$, $h(u) \leq h(v)$
  - action: increase the height of u

The Push Operation

Push(u, v)

if $e(u) > 0$ **and** $h(u) = h(v) + 1$

$d_f(u, v) = \min(e[u], c_f(u, v))$

$f(u, v) \mathrel{+}= d_f(u, v)$

$f(v, u) = -f(u, v)$

$e(u) \mathrel{-}= d_f(u, v)$

$e(v) \mathrel{+}= d_f(u, v)$

- note: excess flow is only pushed downhill by a height differential of 1
- note: if f is a preflow before the push operation is applied, then f is a preflow afterward

The Relabel Operation

Relabel(u)

if $e(u) > 0$ **and** $\forall (u,v) \in E_f$, $h(u) \leq h(v)$

$h(u) = 1 + \min\{h(v) : (u,v) \in E_f\}$

- note: $\{h(v) : (u,v) \in E_f\} \neq \emptyset$, that is, there is at least one edge in E_f that leaves u:
  - $e(u) > 0$ implies that $f(V,u) > 0$
  - so there exists at least one v such that $f(v,u) > 0$
  - but then,

    $c_f(u,v) = c(u,v) - f(u,v) = c(u,v) + f(v,u) > 0$

The Generic Algorithm: Initialise

Preflow(G, s)

for each $u \in V$
  $h(u) = 0$
  $e(u) = 0$
for each $(u, v) \in E$
  $f(u, v) = 0$
  $f(v, u) = 0$
$h(s) = |V|$
for each vertex u neighbour of s
  $f(s, u) = c(s, u)$
  $f(u, s) = -c(s, u)$
  $e(u) = c(s, u)$

- note: h is a height function

The Generic Push-Relabel Algorithm

Initialise Preflow(G, s)

while `there exists an applicable push or relabel operation`

  do `select an applicable push or relabel op and perform it`

Correctness of the Generic Push-Relabel Algorithm

Lemma 5 (An overflowing vertex can be either pushed or relabeled). *Let $G = (V, E)$ be a flow network with source s and sink t, a preflow f and a height function h for f. If u is any overflowing vertex, then either a push or a relabel operation applies to it.*

Proof. If u is overflowing, then there exists at least one edge (u, v) in G_f. Since h is a height function, we have $h(u) \leq h(v) + 1$. If no push operation applies to u then $h(u) < h(v) + 1$ for all $(u, v) \in G_f$. But then a relabel operation applies to u. □

Lemma 6 (Vertex heights never decrease). *During execution of the push-relabel algorithm on a flow network $G = (V, E)$, the height $h(u)$ of a vertex $u \in V$ never decreases. Whenever a relabel operation applies to u, its height $h(u)$ increases by at least 1.*

Proof. Only push or relabel operations may apply to u, of which only the latter modifies $h(u)$. A relabel operation applies when $e(u) > 0$ and $\forall (u, v) \in E_f$ and $h(u) \leq h(v)$, that is when $h(u) < 1 + \min\{h(v) : (u, v) \in E_f\}$. Thus the relabel operation increases $h(u)$. □

Lemma 7 (Maintenance of the height function h). *During execution of the push-relabel algorithm on a flow network $G = (V, E)$ with source s and sink t, h is maintained as a height function.*

Proof. The proof is by induction on the number of operations performed. We have seen that h is a height function at initialisation.

Assume that relabel(u) applies. Then all residual edges $(u, v) \in E_f$ have $h(u) \leq h(v)$. After the relabel operation we'll have $h(u) \leq h(v) + 1$ for some of those edges.

Proof cont. Further, if (w, u) is a residual edge entering u, then $h(w) \leq h(u) + 1$ before relabel(u) is applied, and $h(w) < h(u) + 1$ after it is applied (by Lemma 6).

Now, a push(u,v) operation may remove the edge (u, v) from E_f, it which case h remains a height function. If $(u, v) \in E_f$, h again remains a height function. Or push(u, v) may add (v, u) to E_f. In this case, we have $h(v) = h(u) - 1$, and so h again remains a height function. □

Lemma 8 (There is no s-t path in G_f). *Let $G = (V, E)$ be a flow network with source s and sink t, a preflow f and a height function h for f. Then there is no path from s to t in G_f.*

Proof. Assume there is a path v_0, v_1, $\ldots$, v_k in G_f with $v_0 = s$ and $v_k = t$. WLOG we assume that the path is simple, so that $k < |V|$. Then for $i = 0, 1, \ldots, k-1$, $h(v_i) \leq h(v_{i+1}) + 1$, which implies that $h(s) \leq h(t) + k$; that is $h(s) < |V|$ since $h(t) = 0$. But this contradicts the definition of h. □

Theorem 4 (Correctness of the generic push-relabel algorithm). *If the generic push-relabel algorithm terminates when run on a flow network $G = (V, E)$ with source s and sink t, then the preflow f it computes is a maximum flow for G.*

Proof. Recall that f is maintained as a preflow throughout the algorithms's execution. If the algorithm terminates, then, by Lemmas 5 and 7, $e(u) = 0$ for $u \in V - \{s, t\}$. Thus f is a flow. By Lemmas 7 and 8, and the max-flow min-cut theorem (Theorem 1) f is a maximum flow. □

Generic push-relabel algorithm does terminate and has order $O(V^2E)$. One can show this by bounding the number of push and relabel operations.

Maximum Bipartite Matching

- an application of max-flow

- we consider undirected graphs
- a matching is a set $M \subseteq E$ of edges so that $\forall u \in V$, there is at most one edge in M incident on u (we then say that u is matched by M)
- a maximum matching is a matching of maximum cardinality

- here we only consider bipartite graphs

- we use max flow to find a maximum matching
- for a graph $G = (V, E)$ we construct a corresponding flow network $G' = (V', E')$:
  - $V' = V' \cup \{s, t\}$
  - $E' = \{(s, u) : u \in L\} \cup E \cup \{(u, t) : u \in R\}^*$
  - each edge is assigned unit capacity
  - $*$ L and R are the two parts of V
- concept of integer-valued flow

Lemma 9. *Let $G = (V, E)$ be a bipartite graph with vertex partition $V = L \cup R$, and let G' be its corresponding flow network. Then there is a matching M iff there is an integer-valued flow in G' of value $|M|$.*

Proof. $\rightarrow$: define a flow $f(u, v) = 1$ for each $(u, v) \in M$ and $f(u, v) = 0$ for $(u, v) \notin M$ and consider the cut $(L \cup \{s\}, R \cup \{t\})$.

$\leftarrow$: use the fact that for each vertex $u \in V$ one unit of flow can enter or exit u on at most one edge. □

Theorem 5. *The cardinality of a maximum matching* M *in a bipartite graph* G *equals the value of a maximum flow* f *in its corresponding flow network* G'.

- this follows from the important fact that the Ford-Fulkerson and Edmonds-Karp flow algorithms produce an integer-valued flow when applied to flow networks with integer-valued capacities

Perfect Matching and Hall's Theorem

- a perfect matching is a matching in which every vertex is matched

Theorem 6 (Hall's Marriage Theorem)**.** *Let* $G = (V, E)$ *be an undirected bipartite graph with vertex partition* $V = L \cup R$*, where* $|L| = |R|$*. For any* $X \subseteq V$*, define the neighbourhood of* X *as* $N(X) = \{y \in R : (x, y) \in E \text{ for some } x \in X\}$*. There exists a perfect matching in* G *iff* $|A| \leq |N(A)|$ *for every subset* $A \subseteq L$*.*

Hall's Theorem: Proof Sketch

Proof. Find a matching M in G and use the method of alternating paths (paths starting at $u \in L$, starting with an edge not in M, and composed alternatively by an edge not in M, followed by an edge in M). □

Other Topics

- networks with multiple sources, multiple sinks, multiple commodities
- relabel-to-front algorithm ($O(V^3)$)
  - concept of admissible edges
  - concept of vertex discharge (walking through the list of neighbours)
  - putting a lifted vertex to the front of the vertex list