Maximum Flow Problems III

AU4606: Network Optimization

AI4702: Network Intelligence and Optimization

Xiaoming Duan
Department of Automation
Shanghai Jiao Tong University

October 16, 2023

Last few lectures

- Maximum flow problems: important concepts
  - Residual graphs
  - s-t cut
  - Augmenting paths
- Generic augmenting path algorithms
  - $O(mnU)$
- Most improving augmenting path algorithms
  - $O(m \log(mU)(m \log n))$
- Capacity scaling algorithms
  - $O(m^2 \log U)$
- Shortest path augmenting path algorithms
  - $O(m^2 n)$

Today

1. An application to a combinatorial problem

2. Push-relabel algorithm

Today

1. An application to a combinatorial problem

2. Push-relabel algorithm

Densest subgraph problem

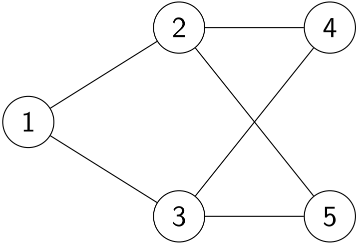
- Let $G = (V, E)$ be an undirected graph
- For a subset $S \subset V$, $G(S) = (S, E(S))$ is the subgraph induced by S
  - $E(S) = \{(i,j) \in E \mid i \in S, j \in S\}$
- The density $D(S)$ of $G(S)$ is defined by

$$D(S) = \frac{|E(S)|}{|S|}$$

Given an undirected graph G, how to find a densest subgraph S^*?

$$S^* \in \operatorname*{argmax}_{S \subset V} D(S) = \operatorname*{argmax}_{S \subset V} \frac{|E(S)|}{|S|}$$

Densest subgraph problem: an example



- $D(\{1\}) = 0$
- $D(\{1,2\}) = \frac{1}{2}$
- $D(\{1,2,3\}) = \frac{2}{3}$
- $D(\{1,2,3,4\}) = 1$
- $D(\{2,3,4,5\}) = 1$
- $D(\{1,2,3,4,5\}) = \frac{6}{5}$

Enumeration of subsets is not computationally efficient

Densest subgraph problem: solution via maximum flow

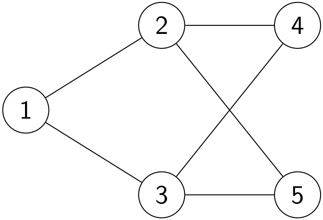
Given $G = (V, E)$, construct a flow network $G' = (V', E')$ as follows

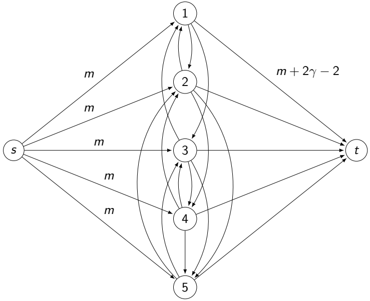
- Introduce a source s and a sink node t, and $V' = V \cup \{s, t\}$
- Add directed arcs from s to nodes in V with capacity m (arc # in G)
- Add directed arcs from nodes in V to t with capacity $m + 2\gamma - d_i$
  - γ is a parameter to be specified
- For each arc $i, j \in E$, add arcs (i, j) and (j, i) to E' with capacity 1

Densest subgraph problem: solution via maximum flow

Given $G = (V, E)$, construct a flow network $G' = (V', E')$ as follows

- Introduce a source s and a sink node t, and $V' = V \cup \{s, t\}$
- Add directed arcs from s to nodes in V with capacity m (arc # in G)
- Add directed arcs from nodes in V to t with capacity $m + 2\gamma - d_i$
  - γ is a parameter to be specified
- For each arc $i, j \in E$, add arcs (i, j) and (j, i) to E' with capacity 1





Densest subgraph problem: solution via maximum flow

Maximum flow in G' and γ

The maximum flow in G' is mn if and only if $\gamma \geq D^*$.

Densest subgraph problem: solution via maximum flow

Maximum flow in G' and γ

The maximum flow in G' is mn if and only if $\gamma \geq D^*$.

Let D' be the second largest density

Densest subgraph

If $D' \leq \gamma < D^*$ and $\{s\} \cup X$ is a minimum cut in G', then $(X, E(X))$ is a densest subgraph.

Densest subgraph problem: solution via maximum flow

Maximum flow in G' and γ

The maximum flow in G' is mn if and only if $\gamma \geq D^*$.

Let D' be the second largest density

Densest subgraph

If $D' \leq \gamma < D^*$ and $\{s\} \cup X$ is a minimum cut in G', then $(X, E(X))$ is a densest subgraph.

Algorithmic idea (binary search)

1. Note $0 \leq D^* \leq m$, let $u = m$ and $\ell = 0$
2. Start from $\gamma = \frac{\ell+u}{2}$
3. If maximum flow equals mn, then $\gamma \geq D^*$ and we set $u = \frac{u+\ell}{2}$ and $\gamma = \frac{\ell+u}{2}$
4. If maximum flow is smaller than mn, then $\gamma < D^*$ and we set $\ell = \frac{u+\ell}{2}$ and $\gamma = \frac{\ell+u}{2}$

But how to certify $D' \leq \gamma < D^*$?

Densest subgraph problem: solution via maximum flow

Algorithmic idea (binary search)

1. Note $0 \leq D^* \leq m$, let $u = m$ and $\ell = 0$
2. Start from an arbitrary $\gamma = \frac{\ell+u}{2}$
3. If maximum flow equals mn, then $\gamma \geq D^*$ and we set $u = \frac{u+\ell}{2}$ and $\gamma = \frac{\ell+u}{2}$
4. If maximum flow is smaller than mn, then $\gamma < D^*$ and we set $\ell = \frac{u+\ell}{2}$ and $\gamma = \frac{\ell+u}{2}$

But how to certify $D' \leq \gamma < D^*$?

Densest subgraph problem: solution via maximum flow

Algorithmic idea (binary search)

1. Note $0 \leq D^* \leq m$, let $u = m$ and $\ell = 0$
2. Start from an arbitrary $\gamma = \frac{\ell + u}{2}$
3. If maximum flow equals mn, then $\gamma \geq D^*$ and we set $u = \frac{u+\ell}{2}$ and $\gamma = \frac{\ell+u}{2}$
4. If maximum flow is smaller than mn, then $\gamma < D^*$ and we set $\ell = \frac{u+\ell}{2}$ and $\gamma = \frac{\ell+u}{2}$

But how to certify $D' \leq \gamma < D^*$?

Bound on $D^* - D'$

For any graph G, we have $D^* - D' \geq \frac{1}{n^2}$

Densest subgraph problem: solution via maximum flow

Algorithmic idea (binary search)

1. Note $0 \le D^* \le m$, let $u = m$ and $\ell = 0$
2. Start from an arbitrary $\gamma = \frac{\ell+u}{2}$
3. If maximum flow equals mn, then $\gamma \ge D^*$ and we set $u = \frac{u+\ell}{2}$ and $\gamma = \frac{\ell+u}{2}$
4. If maximum flow is smaller than mn, then $\gamma < D^*$ and we set $\ell = \frac{u+\ell}{2}$ and $\gamma = \frac{\ell+u}{2}$

But how to certify $D' \le \gamma < D^*$?

Bound on $D^* - D'$

For any graph G, we have $D^* - D' \ge \frac{1}{n^2}$

If $u - \ell \le \frac{1}{n^2}$, then we know $D' \le \gamma < D^*$

Running time?

Densest subgraph problem: solution via maximum flow

Algorithmic idea (binary search)

1. Note $0 \le D^* \le m$, let $u = m$ and $\ell = 0$
2. Start from an arbitrary $\gamma = \frac{\ell+u}{2}$
3. If maximum flow equals mn, then $\gamma \ge D^*$ and we set $u = \frac{u+\ell}{2}$ and $\gamma = \frac{\ell+u}{2}$
4. If maximum flow is smaller than mn, then $\gamma < D^*$ and we set $\ell = \frac{u+\ell}{2}$ and $\gamma = \frac{\ell+u}{2}$

But how to certify $D' \le \gamma < D^*$?

Bound on $D^* - D'$

For any graph G, we have $D^* - D' \ge \frac{1}{n^2}$

If $u - \ell \le \frac{1}{n^2}$, then we know $D' \le \gamma < D^*$

Running time? $O(\log n)$ maximum flow computations

Today

1. An application to a combinatorial problem

2. Push-relabel algorithm

Augmenting path algorithms

Two different types of algorithms

1. Augmenting path algorithms where balance constraints are maintained
2. Push-relabel algorithms where some nodes have excesses

Algorithm Augmenting path algorithm

1: $f \leftarrow 0$
2: **while** $G(f)$ contains a directed path from s to t **do**
3:     Identify an augmenting path P from s to t
4:     $\delta(P) = \min\{r_{ij}, (i,j) \in P\}$
5:     Augment $\delta(P)$ units of flow along P and update $G(f)$
6: **end while**

- In augmenting path algorithms, flows are pushed along paths, and flow balance equations are satisfied all the time

Can we push flows on individual arcs?

Push-relabel algorithms: preflows and excesses

Preflows and excesses

An s-t preflow $f : A \to \mathbb{R}_{\geq 0}$ is an assignment of nonnegative reals to arcs such that

1. $0 \leq f_{ij} \leq u_{ij}$ for all $(i,j) \in A$
2. for all $i \in N \setminus \{s,t\}$,

$$\sum_{j:(j,i)\in A} f_{ji} - \sum_{j:(i,j)\in A} f_{ij} \geq 0$$

For $i \in N \setminus \{s,t\}$, the excess $e(i) = \sum_{j:(j,i)\in A} f_{ji} - \sum_{j:(i,j)\in A} f_{ij} \geq 0$.

Push-relabel algorithms: preflows and excesses

Preflows and excesses

An s-t preflow $f : A \to \mathbb{R}_{\geq 0}$ is an assignment of nonnegative reals to arcs such that

1. $0 \leq f_{ij} \leq u_{ij}$ for all $(i,j) \in A$
2. for all $i \in N \setminus \{s, t\}$,

$$\sum_{j:(j,i)\in A} f_{ji} - \sum_{j:(i,j)\in A} f_{ij} \geq 0$$

For $i \in N \setminus \{s, t\}$, the excess $e(i) = \sum_{j:(j,i)\in A} f_{ji} - \sum_{j:(i,j)\in A} f_{ij} \geq 0$.

Push-relabel algorithm maintains a preflow at intermediate stages

- A node $i \in N \setminus \{s, t\}$ is active if $e(i) > 0$
- Push flows along an arc when node i is active
- When no nodes are active, a feasible flow is established

Since a preflow f satisfies $0 \leq f_{ij} \leq u_{ij}$, residual graph $G(f)$ can be defined

Push-relabel algorithms: distance labels

Since a preflow f satisfies $0 \leq f_{ij} \leq u_{ij}$, residual graph $G(f)$ can be defined

Distance labels

Given a flow (preflow) f, a distance function $d : N \to \mathbb{Z}_{\geq 0}$ satisfies

1. $d(t) = 0$
2. $d(i) \leq d(j) + 1$ for arcs (i, j) in residual graph $G(f)$

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

Push-relabel algorithms: distance labels

Since a preflow f satisfies $0 \leq f_{ij} \leq u_{ij}$, residual graph $G(f)$ can be defined

Distance labels

Given a flow (preflow) f, a distance function $d : N \to \mathbb{Z}_{\geq 0}$ satisfies

1. $d(t) = 0$
2. $d(i) \leq d(j) + 1$ for arcs (i, j) in residual graph $G(f)$

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

Distance labels

Let $d(i)$ for $i \in N$ be distance labels, the following two statements hold

1. $d(i)$ is a lower bound on the shortest path length from i to t in $G(f)$
2. if $d(i) = n$, then there is no directed path from i to t in $G(f)$

Push-relabel algorithms: distance labels

Since a preflow f satisfies $0 \leq f_{ij} \leq u_{ij}$, residual graph $G(f)$ can be defined

Distance labels

Given a flow (preflow) f, a distance function $d : N \to \mathbb{Z}_{\geq 0}$ satisfies

1. $d(t) = 0$
2. $d(i) \leq d(j) + 1$ for arcs (i, j) in residual graph $G(f)$

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

Distance labels

Let $d(i)$ for $i \in N$ be distance labels, the following two statements hold

1. $d(i)$ is a lower bound on the shortest path length from i to t in $G(f)$
2. if $d(i) = n$, then there is no directed path from i to t in $G(f)$

Why distance labels here are just lower bounds?

Push-relabel algorithms: procedure

Algorithm Push-relabel algorithm

$f \leftarrow 0$
$d(s) \leftarrow n$
$d(i) \leftarrow 0$ for $i \in N \setminus \{s\}$
$f_{sj} \leftarrow u_{sj}$ for all $(s,j) \in A$
while there is an active node i **do**
    if there is j such that (i,j) is admissible $(d(i) = d(j) + 1)$ **then**
        $\delta \leftarrow \min\{e(i), r_{ij}\}$
        $f_{ij} \leftarrow f_{ij} + \delta$
    else
        $d(i) \leftarrow \min_{j:(i,j)\in G(f)}\{d(j) + 1\}$
    end if
end while

- Send flows from active nodes to neighboring nodes with smaller label
    1. Nodes with excesses are active (having flows accumulated at the nodes)
    2. Distance labels are estimates of distances to sink
- Relabel amounts to "move a node upward" (water flowing downhill)
- "Water" either flows into the sink or back to the source

Push-relabel algorithms: procedure

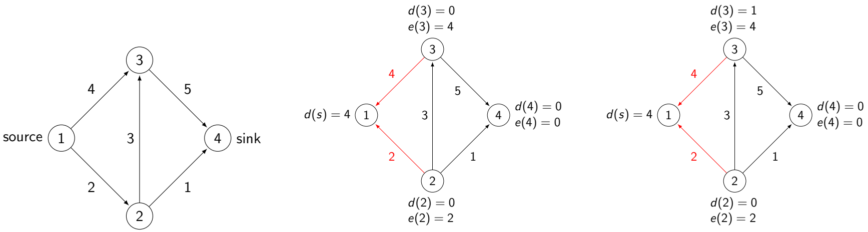
Algorithm Push-relabel algorithm

```
f ← 0
d(s) ← n
d(i) ← 0 for i ∈ N \ {s}
f_sj ← u_sj for all (s, j) ∈ A
while there is an active node i do
    if there is j such that (i, j) is admissible (d(i) = d(j) + 1) then
        δ ← min{e(i), r_ij}
        f_ij ← f_ij + δ
    else
        d(i) ← min_{j:(i,j)∈G(f)} {d(j) + 1}
    end if
end while
```

At the beginning:

- All out-going neighbors of s are active
- The distance labels are valid because $(s, j) \notin G(f)$
- An active node is relabeled in the first iteration

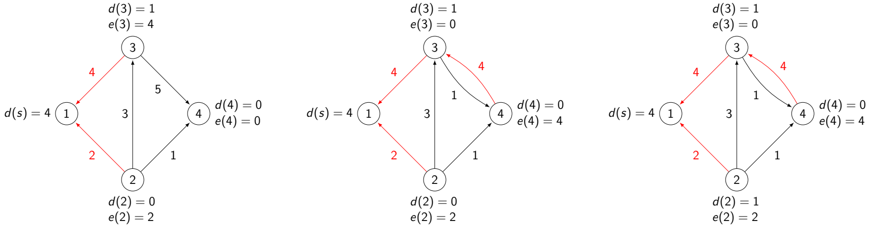
Push-relabel algorithms: example



(a) Graph G (b) Initially (c) Iteration 1

In (b):

- Pick active node 3: no admissible arcs, increase $d(3)$ and obtain (c)



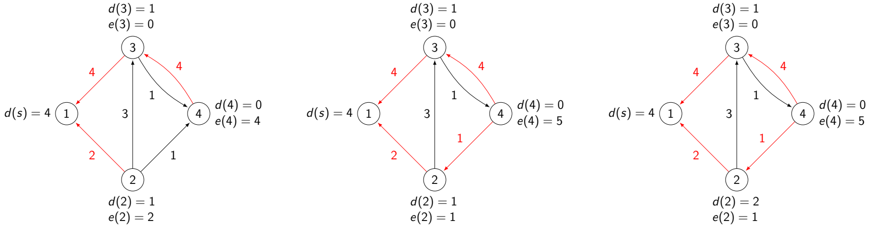
(c) Iteration 1 (d) Iteration 2 (e) Iteration 3

In (c):

- Pick active node 3: push flow $\min\{4, 5\} = 4$ on $(3, 4)$, obtain (d)

In (d):

- Pick active node 2: no admissible arcs, increase $d(2)$ and obtain (e)



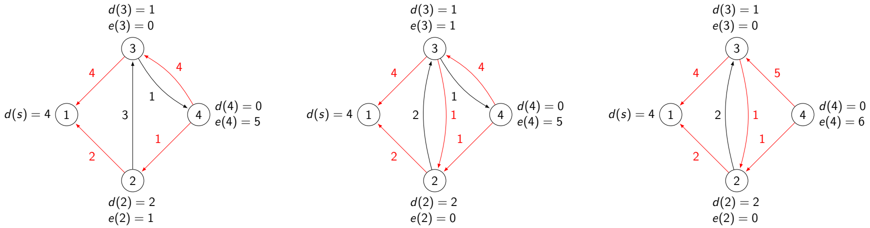
(e) Iteration 3 (f) Iteration 4 (g) Iteration 5

In (e):

- Pick active node 2: push flow $\min\{2, 1\} = 1$ on $(2, 4)$, obtain (f)

In (f):

- Pick active node 2: no admissible arcs, increase $d(2)$ and obtain (g)



(g) Iteration 5 (h) Iteration 6 (i) Iteration 7

In (g):

- Pick active node 2: push flow $\min\{1, 3\} = 1$ on $(2, 3)$, obtain (h)

In (h):

- Pick active node 3: push flow $\min\{1, 1\} = 1$ on $(3, 4)$, obtain (i)

In (i)

- No active nodes, algorithm terminates

Push-relabel algorithms: bounding number of relabeling

Preflows

The push-relabel algorithm maintains a preflow.

Valid distance labels

The push-relabel algorithm maintains a valid distance labeling.

Push-relabel algorithms: bounding number of relabeling

Preflows

The push-relabel algorithm maintains a preflow.

Valid distance labels

The push-relabel algorithm maintains a valid distance labeling.

Positive excesses and paths

At any stage of the algorithm, for each node i with positive excess $e(i) > 0$, there exists a directed path from i to s in the residual graph.

Corollary: for any $i \in N$, $d(i) \leq 2n - 1$

Push-relabel algorithms: bounding number of relabeling

Preflows

The push-relabel algorithm maintains a preflow.

Valid distance labels

The push-relabel algorithm maintains a valid distance labeling.

Positive excesses and paths

At any stage of the algorithm, for each node i with positive excess $e(i) > 0$, there exists a directed path from i to s in the residual graph.

Corollary: for any $i \in N$, $d(i) \leq 2n - 1$

Total number of relabel operations

The number of relabel operations in the push-relabel algorithm is $O(n^2)$.

Push-relabel algorithms: bounding number of pushes

Recall flow $\delta = \min\{e(i), r_{ij}\}$ is pushed in a push operation

- A saturating push is a push where $\delta = r_{ij}$
- Otherwise, it is a nonsaturating push

Total number of saturating pushes

The number of saturating pushes operations performed in the push-relabel algorithm is $O(mn)$.

Push-relabel algorithms: bounding number of pushes

Recall flow $\delta = \min\{e(i), r_{ij}\}$ is pushed in a push operation

- A saturating push is a push where $\delta = r_{ij}$
- Otherwise, it is a nonsaturating push

Total number of saturating pushes

The number of saturating pushes operations performed in the push-relabel algorithm is $O(mn)$.

Total number of nonsaturating pushes

The number of nonsaturating pushes operations performed in the push-relabel algorithm is $O(mn^2)$.

Push-relabel algorithms: bounding number of pushes

Recall flow $\delta = \min\{e(i), r_{ij}\}$ is pushed in a push operation

- A saturating push is a push where $\delta = r_{ij}$
- Otherwise, it is a nonsaturating push

Total number of saturating pushes

The number of saturating pushes operations performed in the push-relabel algorithm is $O(mn)$.

Total number of nonsaturating pushes

The number of nonsaturating pushes operations performed in the push-relabel algorithm is $O(mn^2)$.

Complexity

The complexity of the push-relabel algorithm is $O(mn^2)$.

Push-relabel algorithms: further improvements

Which active node to examine in each iteration is not specified

- FIFO push-relabel: pick active nodes in a first-in-first-out order
  - $O(n^3)$
- Highest label push-relabel: pick active node with highest dist. label
  - $O(n^2\sqrt{m})$
- Excess scaling: pick active nodes with sufficiently large excess
  - $O(nm + n^2 \log U)$

Upcoming

Week 1-8 (AU4606 & AI4702):

- Introduction (1 lecture)
- Preparations (3 lectures)
  - basics of graph theory
  - algorithm complexity and data structure
  - graph search algorithm
- Shortest path problems (3 lectures)
- Maximum flow problems (this and previous few lectures)
- Minimum cost flow problems (next two lectures)
- Introduction to multi-agent systems (1 lecture)
- Introduction to cloud networks (1 lecture)

Week 9-16 (AU4606):

- Simplex and network simplex methods (2 lectures)
- Global minimum cut problems (3 lectures)
- Minimum spanning tree problems (3 lectures)