Deterministic Max-Flows in Simple Graphs

Tianyi Zhang, Tsinghua Univ.
Max-flows in simple graphs

Graph $G = (V, E)$
- $n$ vertices
- $m$ edges
- No parallel edges

Capacities
- Unit

Terminals
- $s, t \in V$
Max-flows in simple graphs

Graph $G = (V, E)$
- $n$ vertices
- $m$ edges
- No parallel edges

Capacities
- Unit

Terminals
- $s, t \in V$

max-flow has value 2
# History

<table>
<thead>
<tr>
<th>Reference</th>
<th>Running time</th>
<th>det / rand?</th>
</tr>
</thead>
<tbody>
<tr>
<td>[KL’98]</td>
<td>$O(m + n\tau^{3/2})$</td>
<td>det</td>
</tr>
<tr>
<td>[KL’02]</td>
<td>$\tilde{O}(m + n\tau)$</td>
<td>rand</td>
</tr>
<tr>
<td>[Duan’13]</td>
<td>$\tilde{O}(n^{9/4}\tau^{1/8})$</td>
<td>det</td>
</tr>
<tr>
<td><strong>Ours</strong></td>
<td>$\tilde{O}(m + n^{5/3}\tau^{1/2})$</td>
<td>det</td>
</tr>
</tbody>
</table>

$n = \#\text{vertices}, m = \#\text{edges}$

$\tau = \text{an upper bound on max-flow}$

Fastest when $\tau > n^{0.67}$
Flow decycling
Ford-Fulkerson

- Residual graph $G_f$ of $G$ w.r.t flow $f$

- **Ford-Fulkerson:**
  - Keep finding **augmenting paths** from $s$ to $t$ in $G_f$

- Running time $= \tilde{O}(m\tau)$
  - $\tau$ is a known upper bound on the max-flow value
Flow decycling

- Lemma: [Karger & Levine ’98]
  Acyclic flow $f$ with value $|f|$ has $O(n |f|^{1/2})$ flow edges

- $G_f$ has at most $O(n |f|^{1/2})$ directed edges when $f$ is acyclic
Flow decycling

• **Lemma: [Karger & Levine ’98]**
  Acyclic flow $f$ with value $|f|$ has $O(n |f|^{1/2})$ flow edges.

• $G_f$ has at most $O(n |f|^{1/2})$ directed edges when $f$ is acyclic.

• **Algorithm: [Karger & Levine ’98]**
  While $\exists$ augmenting path in $G_f$
  contract all connected components by undi-edges in $G_f$
  BFS on the contracted $G_f$ which contains only $O(n |f|^{1/2})$ di-edges
  augment flow $f$, then decycle $f$

• Running time $= \tilde{O}(m + n \tau^{3/2})$
  $G_f$ always has $O(n \tau^{1/2})$ edges, so total time $= \tilde{O}(m + \tau \cdot n \tau^{1/2})$
Blocking flows
Blocking flows

- Form a level graph by the distance from $s$ in $G_f$
Blocking flows

- Form a level graph by the distance from $s$ in $G_f$
- Find a maximal set of shortest disjoint aug-paths

![Graph Diagram]
Blocking flows

- Form a level graph by the distance from \( s \) in \( G_f \)
- Find a maximal set of shortest disjoint aug-paths
- \( \text{Dist}(s, t) \) in \( G_f \) increases
Blocking flows

- Repeat blocking-flows until $\text{Dist}(s, t) \geq L$
- Residual flow in $G_f$ becomes at most $O(n^2/L^2)$
- Then apply Ford-Fulkerson $O(n^2/L^2)$ times

\[ \text{Running time} = L \cdot \text{BF} + \frac{n^2}{L^2} \cdot \text{FF} = mn^{2/3} \quad \text{[Goldberg & Rao '98]} \]
Decycling + Blocking-flow

• **Lemma: [Karger & Levine ’98]**
  Acyclic flow $f$ with value $|f|$ has $O(n |f|^{1/2})$ flow edges

• Exists subgraph $H \subseteq G$ with $O(n \tau^{1/2})$ edges that contains the max-flow

• If we knew $H$ beforehand, then applying blocking-flow on $H$ can compute max-flow in $\tilde{O}(n^{5/3} \tau^{1/2})$ time

• Ideally, shoot for $\tilde{O}(m + n^{5/3} \tau^{1/2})$
Decycling + blocking-flows
[Duan ’13]
Combining two techniques

- Compute a blocking-flow in $G_f$ where $f$ is acyclic
  
  Take time $\tilde{O}(n |f|^{1/2})$ since undirected edges are contracted
Combining two techniques

- Compute a blocking-flow in $G_f$ where $f$ is acyclic
  
  Take time $\tilde{O}(n |f|^{1/2})$ since undirected edges are contracted
Combining two techniques

- Compute a blocking-flow in $G_f$ where $f$ is acyclic
  Take time $\tilde{O}(n |f|^{1/2})$ since undirected edges are contracted

- Augment flow $f \leftarrow f + \Delta f$, so $\text{Dist}(s, t)$ in $G_f$ increases
  But now, $f$ might contain cycles

\[ f \leftarrow f + \Delta f \]

Dist($s$, $t$)

\[
\begin{align*}
\text{level 1} & \quad \text{level 2} & \quad \ldots \ldots & \quad \text{level L-1} & \quad \text{level L}
\end{align*}
\]
Combining two techniques

- Augment flow $f \leftarrow f + \Delta f$, so $\text{Dist}(s, t)$ in $G_f$ increases
  But now, $f$ might contain cycles
Combining two techniques

- Augment flow $f \leftarrow f + \Delta f$, so $\text{Dist}(s, t)$ in $G_f$ increases

  But now, $f$ might contain cycles
Combining two techniques

- Augment flow $f \leftarrow f + \Delta f$, so $\text{Dist}(s, t)$ in $G_f$ increases
  But now, $f$ might contain cycles

- Decycling adds *undirected edges* between level $i$ & $(i+1)$

- Blocking-flows becomes costly as #undi-edges grows
  Cannot contract these undirected edges

![Graph diagram](image-url)
Clustering

• Trouble: \#undi-edges grows larger than $n \tau^{1/2}$
  Computing blocking-flows becomes costly

• Key idea: [Duan’13] partition into star-subgraphs

- Partition vertices into $V_1 \cup V_2$
- $V_1$ is a union of star-graphs, each of size $\geq h$
- Edges between $V_2$ is at most $O(nh)$
Clustering

- Trouble: \#undi-edges grows larger than $n \tau^{1/2}$
  Computing blocking-flows becomes costly

- Key idea: [Duan’13] partition into star-subgraphs

- Partition vertices into $V_1 \cup V_2$
- $V_1$ is a union of star-graphs, each of size $\geq h$
- Edges between $V_2$ is at most $O(nh)$

greedy clustering
Clustering

- Trouble: \#undi-edges grows larger than $n \tau^{1/2}$
  Computing blocking-flows becomes costly

- Key idea: [Duan’13] partition into star-subgraphs

- Partition vertices into $V_1 \cup V_2$
- $V_1$ is a union of star-graphs, each of size $\geq h$
- Edges between $V_2$ is at most $O(nh)$
Clustering

• Trouble: #undi-edges grows larger than $n \tau^{1/2}$
  Computing blocking-flows becomes costly

• Key idea: [Duan’13] partition into star-subgraphs

  • Partition vertices into $V_1 \cup V_2$
  • $V_1$ is a union of star-graphs, each of size $\geq h$
  • Edges between $V_2$ is at most $O(nh)$

-greedy clustering
Dynamic maintenance

- **Updates**: augmentations turn undi-edgs to di-edges
  - Delete di-edges from the clustering structure

- Need dynamic maintenance of the clustering structure

**Deletion of star edges:**
- Disconnects the vertex, and move it downward
- Increase total degree by at most $O(n)$
- Rebuild if total degree exceeds $nh$
Dynamic maintenance

• **Updates:** augmentations turn undi-edgs to di-edges
  - Delete di-edges from the clustering structure

• Need dynamic maintenance of the clustering structure

**Deletion of star edges:**
- Disconnects the vertex, and move it downward
- Increase total degree by at most $O(n)$
- Rebuild if total degree exceeds $nh$
Dynamic maintenance

- **Updates:** augmentations turn undi-edgs to di-edges
  - Delete di-edges from the clustering structure

- Need dynamic maintenance of the clustering structure

**Deletion of star edges:**
- Disconnects the vertex, and move it downward
- Increase total degree by at most $O(n)$
- Rebuild if total degree exceeds $nh$
Our technique
a more careful clustering scheme
Avoid rebuilding

- Previously in [Duan’13], the clustering scheme needs frequent rebuilding procedures

- **Solution:** try to avoid rebuilding entirely

\[ \text{star of size } \geq h \quad \text{adjacent to stars} \quad \text{avg low-deg} < h \]
Avoid rebuilding

- **Solution:** try to avoid rebuilding entirely
- When a star edge is deleted, if it is not adjacent to stars, either collect a new star, or move to low-degree part

![Diagram with star conditions]

- **Star of size $\geq h$$^1$**
- **Adjacent to stars**
- **Avg low-deg $< h$**
Solution: try to avoid rebuilding entirely.

When a star edge is deleted, if it is not adjacent to stars, either collect a new star, or move to low-degree part.
Avoid rebuilding

- **Solution:** try to avoid rebuilding entirely

- When a star edge is deleted, if it is not adjacent to stars, either collect a new star, or move to low-degree part.
Avoid rebuilding

- **Solution:** try to avoid rebuilding entirely
- When a star edge is deleted, if it is not adjacent to stars, either collect a new star, or move to low-degree part
Proof of concept

When $\tau = \Theta(n)$

- [Duan’13] has running time $\tilde{O}(n^{2.375})$ for max-flow
- New clustering scheme already improves to $\tilde{O}(n^{2.25})$
A multi-layer approach

• Handling a deletion needs to scan the adjacency list
  Adjacency list can be as large as $O(n)$

• If adjacency list is large, then can have larger stars
  Use multiple layers to handle different vertex degrees
A multi-layer approach

stars adjacent to stars

\[ \text{deg} \leq n \]

low-deg

\[ \text{deg} \leq n/2 \]
A multi-layer approach

\[
\text{deg} \leq n
\]

\[
\text{deg} \leq n/2^i
\]

\[
\text{deg} \leq n/2^{i+1}
\]
A multi-layer approach

stars adjacent to stars

deg ≤ n

deg ≤ n/2^i

deg ≤ 2h  deg ≤ h
A multi-layer approach
A multi-layer approach

stars adjacent to stars

stars adjacent to stars

stars adjacent to stars

low-deg

deg ≤ \( n \)

arboricity ≤ \( n \)

deg ≤ \( n/2^i \)

arboricity ≤ \( n/2^i \)

deg ≤ 2h

arboricity ≤ 2h

deg ≤ h

arboricity ≤ h

How to utilize low arboricity?
Low arboricity

star of size $\geq n/2^i$ adjacent to stars
Low arboricity

star of size $\geq \frac{n}{2^i}$ adjacent to stars

Edge orientation such that out-degrees $\leq \frac{n}{2^{i-1}}$
Low arboricity

Two kinds of operations:
• Turn red vertices into purple vertices
• Find stars among purple vertices, and turn them red ones

star of size $\geq n/2^i$ adjacent to stars

Edge orientation such that out-degrees $\leq n/2^{i-1}$
Low arboricity

Two kinds of operations:

• Turn red vertices into purple vertices
• Find stars among purple vertices, and turn them red ones

star of size $\geq \frac{n}{2}$

Edge deletion
Low arboricity

Two kinds of operations:

• Turn red vertices into purple vertices
• Find stars among purple vertices, and turn them red ones

star of size $\geq n/2^i$ adjacent to stars
Low arboricity

Two kinds of operations:

• Turn red vertices into purple vertices
• Find stars among purple vertices, and turn them red ones

star of size $\geq n/2^i$ adjacent to new cluster
Low arboricity

- To detect large stars, need to explicitly store the induced subgraph on purple vertices.
- When red $\rightarrow$ purple, need to scan adjacency-list to find all purple neighbors which is costly.

Star of size $\geq n/2^i$ adjacent to stars.
Low arboricity

- When red $\rightarrow$ purple
  Need to scan adjacency-list to find all purple neighbors which is costly
- Arboricity helps!
- Purple vertices store in/out-purple neighbors
- Red vertices only store in-purple neighbors

star of size $\geq n/2^i$ adjacent to stars
Low arboricity

- When red $\rightarrow$ purple
  Need to scan adjacency-list to find all purple neighbors which is costly

- Arboricity helps!

- Purple vertices store in/out-purple neighbors

- Red vertices only store in-purple neighbors

- When red $\leftarrow$ purple
  Only need to scan out-neighbors, which is at most $n/2^{i-1}$

star of size $\geq n/2^i$ adjacent to stars
Further questions

• How about deterministic max-flow in multi-graphs?

• .... in weighted graphs?
Further questions

• How about deterministic max-flow in multi-graphs?

• …. in weighted graphs?

Thanks for listening