Multiple-Source Multiple-Sink Maximum Flow in Directed Planar Graphs in Near-Linear Time

Shay Mozes

joint work with Cora Borradaile, Philip Klein, Yahav Nussbaum and Christian Wulff-Nilsen
Planar Graphs
Planar Graphs
Planar Graphs
Planar Graphs

• arise in many applications
Planar Graphs

• arise in many applications

• admit faster algorithms
Planar Graphs

- arise in many applications
- admit faster algorithms
- interesting structural properties
Maximum Flow

**input:** a graph $G$ with arc capacities and nodes $s,t$

**output:** an assignment of flow to arcs such that:

- **conservation** at non-terminals
- **respects capacity** at all arcs
- **maximizes** the amount of flow entering $t$
Maximum Flow

**input:** a graph $G$ with arc capacities and nodes $s, t$

**output:** an assignment of flow to arcs such that:

- **conservation** at non-terminals
- **respects capacity** at all arcs
- **maximizes** the amount of flow entering $t$
Main Result

multiple-source, multiple-sink maximum flow in directed planar graphs in $O(n \log^3 n)$ time.
Applications
Multiple Sources and Sinks

- transportation networks (Soviet railroad system)
- computer vision - image segmentation, restoration, stereo, object recognition, texture synthesis (grid)
- maximum bipartite matching
Reduction to Single Source and Sink
Reduction to Single Source and Sink
Reduction to Single Source and Sink

• reduction does not preserve planarity

• [Miller, Naor ’91] - sources and sinks on a small number of faces
Known Results for Single Source/Sink

general graphs:
• $\tilde{O}(nm)$ - many results (blocking flow, push relabel)
• $O(m^{3/2} \log(n^2/m) \log U)$ - [Goldberg, Rao ’97]

directed planar graphs:
• $O(n)$ - $s$ and $t$ on the same face [Hassin ’81 + Henzinger et al. ’94]
• $O(n \log n)$ [Borradaile, Klein ’06]
Outline

• a few tools and definitions
• high-level description of recursive algorithm
• main ingredients for near-linear time
Multiple Sinks on a path
Multiple Sinks on a path

reduces to the single sink case -
connect all sinks with infinite-capacity edges

preserves planarity!
The Residual Graph

• given flow $f$ in graph $G$ with capacities $c(a)$, the residual graph $G_f$ has same nodes and arcs as $G$ and capacities $c_f(a) = c(a) - f(a)$

• a path $P$ is residual if every arc of $P$ has positive capacity
The Residual Graph

• given flow $f$ in graph $G$ with capacities $c(a)$, the residual graph $G_f$ has same nodes and arcs as $G$ and capacities $c_f(a) = c(a) - f(a)$

• a path $P$ is residual if every arc of $P$ has positive capacity
a flow $f$ is maximum iff there are no residual paths from sources to sinks in $G_f$
A flow $f$ is maximum iff there are no residual paths from sources to sinks in $G_f$. 
a flow $f$ is maximum iff there are no residual paths from sources to sinks in $G_f$
Flow Zoo

- **excess flow** at node $v$ is the difference between amount of flow entering $v$ and leaving $v$.
  Conservation $\Rightarrow$ excess flow is zero.

- **pseudoflow**: arc capacities are respected (conservation may not).

- **feasible flow**: pseudoflow that obeys conservation everywhere except sources and sinks.

- **circulation**: pseudoflow that obeys conservation everywhere (even at sources and sinks).

- given a pseudoflow, it is possible to push back all positive/negative excess flow to/from its origin in linear time.
think of sources as having excess flow $+\infty$
think of sinks as having excess flow $-\infty$

a pseudoflow corresponds to a maximum flow iff there are no residual paths from $+$ to $-$ in the residual graph
think of sources as having excess flow $+\infty$
think of sinks as having excess flow $-\infty$
a pseudoflow corresponds to a maximum flow iff there are no residual paths from $+$ to $-$ in the residual graph
think of sources as having excess flow $+\infty$
think of sinks as having excess flow $-\infty$

a pseudoflow corresponds to a maximum flow iff there are no residual paths from $+$ to $-$ in the residual graph.
Cycle Separators [Miller ’86]

- simple cycle in a triangulated 2-connected planar graph
- balanced - between $n/3$ and $2n/3$ nodes on each side
- small: consists of $O(\sqrt{n})$ nodes
- can be found in $O(n)$ time
Outline

• a few tools and definitions
• high-level description of recursive algorithm
• main ingredients for near-linear time
Recursion, First try
Recursion, First try

- find separator
Recursion, First try

- find separator
- find maximum MSMS flow inside and outsider recursively
Recursion, First try

- find separator
- find maximum MSMS flow inside and outsider recursively
- no residual paths from sources to sinks in each subgraph
Recursion, Second try
Recursion, Second try

• find separator
Recursion, Second try

- find separator
- recursive problem (almost):
  - eliminate residual paths:
    - from sources to sinks
    - from sources to separator
    - from separator to sinks
Recursion, Second try

• find separator
• recursive problem (almost): eliminate residual paths
  • from sources to sinks
  • from sources to separator
  • from separator to sinks
Recursion, Second try

- find separator
- recursive problem (almost): eliminate residual paths
  - from sources to sinks
  - from sources to separator
  - from separator to sinks
Recursion, Second try

• find separator
• recursive problem (almost): eliminate residual paths
  • from sources to sinks
  • from sources to separator
  • from separator to sinks

• eliminate residual paths from + to - on separator
Recursion, Second try

- find separator
- recursive problem (almost):
  - eliminate residual paths
    - from sources to sinks
    - from sources to separator
    - from separator to sinks

- eliminate residual paths from + to - on separator
Recursion, Second try

- find separator
- recursive problem (almost): eliminate residual paths
  - from sources to sinks
  - from sources to separator
  - from separator to sinks

- eliminate residual paths from + to - on separator

- return flow from + to sources and from sinks to -
Recursion, Second try

- find separator
- recursive problem (almost):
  - eliminate residual paths
    - from sources to sinks
    - from sources to separator
    - from separator to sinks
- eliminate residual paths from + to - on separator
- return flow from + to sources and from sinks to -
Recursion, Second try

- find separator
- recursive problem (almost):
  - eliminate residual paths
    - from sources to sinks
    - from sources to separator
    - from separator to sinks
- eliminate residual paths from + to - on separator
- return flow from + to sources and from sinks to -
Fixing the Separator

eliminate residual paths from + to - on separator
Fixing the Separator

eliminate residual paths from + to - on separator

• make capacity of separator edges infinite
• handle nodes one by one:
Fixing the Separator

eliminate residual paths from + to - on separator

• make capacity of separator edges infinite
• handle nodes one by one:
• reduce capacity of incident edges back to original
Fixing the Separator

eliminate residual paths from + to - on separator

• make capacity of separator edges infinite
• handle nodes one by one:
  • reduce capacity of incident edges back to original
  • push + excess to neighbor using max-flow in residual graph
Fixing the Separator

eliminate residual paths from + to - on separator

• make capacity of separator edges infinite
• handle nodes one by one:

• reduce capacity of incident edges back to original
• push + excess to neighbor
• push - excess from neighbor using max-flow in residual graph
Fixing the Separator

eliminate residual paths from + to - on separator

• make capacity of separator edges infinite
• handle nodes one by one:
  • reduce capacity of incident edges back to original
  • push + excess to neighbor
  • push - excess from neighbor using max-flow in residual graph
Fixing the Separator

eliminate residual paths from + to - on separator

• make capacity of separator edges infinite
• handle nodes one by one:
  • reduce capacity of incident edges back to original
  • push + excess to neighbor
  • push - excess from neighbor

running time: \( O(\sqrt{n}) \cdot O(n) = O(n^{3/2}) \)

separator nodes \quad \text{time for max-flow between neighbors [Hassin + Henzinger et al.]}
Outline

- a few tools and definitions
- high-level description of recursive algorithm
- main ingredients for near-linear time
bottleneck is fixing step which consists of $O(\sqrt{n})$ max-flow computations in residual graph between neighbor nodes on a simple cycle

can represent the flow compactly: flow is in graph with $O(n)$ edges representation has size $O(\sqrt{n})$
maintain flow only on separator edges flow elsewhere represented implicitly

$\Rightarrow$ can perform each max-flow computation in $O(\sqrt{n} \log^2 n)$ instead of $O(n)$
Planar Duality
Planar Duality
Planar Duality
Max-Flow between Neighbors
[Hassin 1981]
to compute max flow from $s$ to $t$: 
Max-Flow between Neighbors

[Hassin 1981]

to compute max flow from $s$ to $t$:

• make capacity of arc $ts$ infinite
Max-Flow between Neighbors [Hassin 1981]

to compute max flow from $s$ to $t$:

- make capacity of arc $ts$ infinite
- $\phi_0 =$ the face to the left of arc $ts$
Max-Flow between Neighbors
[Hassin 1981]

to compute max flow from $s$ to $t$:

• make capacity of arc $ts$ infinite
• $\phi_0 =$ the face to the left of arc $ts$
• consider capacity of an arc in the primal as its length in the dual
Max-Flow between Neighbors
[Hassin 1981]

to compute max flow from $s$ to $t$:

- make capacity of arc $ts$ infinite
- $\phi_0 =$ the face to the left of arc $ts$
- consider capacity of an arc in the primal as its length in the dual
- compute:
  \[ d(\phi) = \text{distance of } \phi \text{ from } \phi_0 \text{ in dual} \]
Max-Flow between Neighbors

[Hassin 1981]

to compute max flow from $s$ to $t$:

- make capacity of arc $ts$ infinite
- $\phi_0 =$ the face to the left of arc $ts$
- consider capacity of an arc in the primal as its length in the dual
- compute:
  $d(\phi) =$ distance of $\phi$ from $\phi_0$ in dual
- define flow on arc $a$ by:
  $\sigma(a) = d(\text{face right of } a) - d(\text{face left of } a)$

$\sigma$ is a feasible circulation that maximizes the flow on arc $ts$
$\sigma$ is a feasible circulation
$\sigma$ is a feasible circulation

conservation:

$\sigma(a) = d(\text{face right of } a) - d(\text{face left of } a)$

flows on arcs outgoing from a node cancel to zero
\( \sigma \) is a feasible circulation

**conservation:**
\[ \sigma(a) = d(\text{face right of } a) - d(\text{face left of } a) \]
flows on arcs outgoing from a node cancel to zero

feasibility guaranteed by shortest paths inequality:
$\sigma$ is a feasible circulation

conservation:
$\sigma(a) = d(\text{face right of } a) - d(\text{face left of } a)$
flows on arcs outgoing from a node cancel to zero

feasibility guaranteed by shortest paths inequality:
$d(\text{head of dual of } a) \leq d(\text{tail of dual of } a) + \text{length(dual of } a)$
\( \sigma \) is a feasible circulation

**conservation:**
\[
\sigma(a) = d(\text{face right of } a) - d(\text{face left of } a)
\]
flows on arcs outgoing from a node cancel to zero

**feasibility guaranteed by shortest paths inequality:**
\[
d(\text{head of dual of } a) \leq d(\text{tail of dual of } a) + \text{length(dual of } a)
\]
\[
d(\text{head of dual of } a) - d(\text{tail of dual of } a) \leq \text{capacity of } a
\]
$\sigma$ is a feasible circulation

conservation:
$\sigma(a) = d(\text{face right of } a) - d(\text{face left of } a)$
flows on arcs outgoing from a node cancel to zero

feasibility guaranteed by shortest paths inequality:
$d(\text{head of dual of } a) \leq d(\text{tail of dual of } a) + \text{length(dual of } a)$
$d(\text{head of dual of } a) - d(\text{tail of dual of } a) \leq \text{capacity of } a$

$\sigma(a) = d(\text{face right of } a) - d(\text{face left of } a)$
$= d(\text{head of dual of } a) - d(\text{tail of dual of } a)$
$\leq \text{capacity of } a$
Max-Flow between Neighbors

[Hassin 1981]

to compute max flow from $s$ to $t$:

- make capacity of arc $ts$ infinite
- $\phi_0 = \text{the face to the left of arc } ts$
- consider capacity of an arc in the primal as its length in the dual
- compute:
  $d(\phi) = \text{distance of } \phi \text{ from } \phi_0 \text{ in dual}$
- define flow on arc $a$ by:
  $\sigma(a) = d(\text{face right of } a) - d(\text{face left of } a)$

$\sigma$ is a feasible circulation that maximizes the flow on arc $ts$
Max-Flow between Neighbors

[Hassin 1981]

to compute max flow from $s$ to $t$:

- make capacity of arc $ts$ infinite
- $\phi_0 =$ the face to the left of arc $ts$
- consider capacity of an arc in the primal as its length in the dual
- compute:
  \[ d(\phi) = \text{distance of } \phi \text{ from } \phi_0 \text{ in dual} \]
- define flow on arc $a$ by:
  \[ \sigma(a) = d(\text{face right of } a) - d(\text{face left of } a) \]
  \( \sigma \) is a feasible circulation that maximizes the flow on arc $ts$
- don’t push flow on $ts$
Flow Representation 1/4
Flow Representation 1/4

flow between endpoints of arc $a_i$ of $C$ can be represented by:
Flow Representation 1/4

flow between endpoints of arc $a_i$ of $C$
can be represented by:

• face labels $d_i(\phi)$ for each face $\phi$
Flow Representation 1/4

flow between endpoints of arc $a_i$ of $C$
can be represented by:
• face labels $d_i(\phi)$ for each face $\phi$
• flow on $a_i$
Flow Representation 1/4

flow between endpoints of arc $a_i$ of $C$ can be represented by:

- face labels $d_i(\phi)$ for each face $\phi$
- flow on $a_i$
flow between endpoints of arc \( a_i \) of \( C \) can be represented by:

- face labels \( d_i(\phi) \) for each face \( \phi \)
- flow on \( a_i \)

...to represent sum of flows for all iterations of fixing step:
Flow Representation 1/4

flow between endpoints of arc $a_i$ of $C$ can be represented by:
• face labels $d_i(\phi)$ for each face $\phi$
• flow on $a_i$

to represent sum of flows for all iterations of fixing step:
• accumulate face labels over all iterations (linearity)
Flow Representation 1/4

flow between endpoints of arc $a_i$ of $C$ can be represented by:

• face labels $d_i(\phi)$ for each face $\phi$
• flow on $a_i$

to represent sum of flows for all iterations of fixing step:

• accumulate face labels over all iterations (linearity)
• explicitly store flow on arcs of separator $C$
flow between endpoints of arc $a_i$ of $C$
can be represented by:
• face labels $d_i(\phi)$ for each face $\phi$
• flow on $a_i$

to represent sum of flows for all iterations of fixing step:
• accumulate face labels over all iterations (linearity)
• explicitly store flow on arcs of separator $C$

will show it suffices to store face labels
for just the faces adjacent to separator $C$
$f_0$ - flow after recursive calls

$f$ - flow on C’s arcs up to current iteration

d - accumulated face labels up to current iteration
$f_0$ - flow after recursive calls
$f$ - flow on $C$’s arcs up to current iteration
$d$ - accumulated face labels up to current iteration
Flow Representation 2/4

\( f_0 \) - flow after recursive calls

\( f \) - flow on \( C \)’s arcs up to current iteration

\( d \) - accumulated face labels up to current iteration

- for an arc \( a \) not on \( C \), flow is:
  \[ f_0(a) + d(\text{face right of } a) - d(\text{face left of } a) \]
Flow Representation 2/4

\( f_0 \) - flow after recursive calls
\( f \) - flow on \( C \)'s arcs up to current iteration
\( d \) - accumulated face labels up to current iteration

- for an arc \( a \) not on \( C \), flow is:
  \[ f_0(a) + d(\text{face right of } a) - d(\text{face left of } a) \]

- residual capacity of \( a \) is:
  \[ c(a) - f_0(a) - d(\text{face right of } a) + d(\text{face left of } a) \]
\( f_0 \) - flow after recursive calls
\( f \) - flow on \( C \)'s arcs up to current iteration
\( d \) - accumulated face labels up to current iteration

- for an arc \( a \) not on \( C \), flow is:
  \[
  f_0(a) + d(\text{face right of } a) - d(\text{face left of } a)
  \]

- residual capacity of \( a \) is:
  \[
  c(a) - f_0(a) - d(\text{face right of } a) + d(\text{face left of } a)
  \]

- length of dual of \( a \) is:
  \[
  c(a) - f_0(a) - d(\text{head of dual of } a) + d(\text{tail of dual of } a)
  \]
Flow Representation 3/4

- length of dual of $a$ is:
  
  $c(a) - f_0(a) - d(\text{head of dual of } a) + d(\text{tail of dual of } a)$
• length of dual of $a$ is:

$$c(a) - f_0(a) - d(\text{head of dual of } a) + d(\text{tail of dual of } a)$$
Flow Representation 3/4

- length of dual of \( a \) is:
  \[ c(a) - f_0(a) - d(\text{head of dual of } a) + d(\text{tail of dual of } a) \]
- length of any dual path \( P \) that does not use dual arcs of \( C \) is:
  \[ \sum c(a) - f_0(a) - d(\text{head of dual of } a) + d(\text{tail of dual of } a) \]
  \[ = d(\text{end of } P) - d(\text{start of } P) + \sum c(a) - f_0(a) \]
• length of dual of $a$ is:
  $$c(a) - f_0(a) - d(\text{head of dual of } a) + d(\text{tail of dual of } a)$$

• length of any dual path $P$ that does not use dual arcs of $C$ is:
  $$\sum c(a) - f_0(a) - d(\text{head of dual of } a) + d(\text{tail of dual of } a)$$
  $$= d(\text{end of } P) - d(\text{start of } P) + \sum c(a) - f_0(a)$$

• ignoring arcs of $C$, shortest paths are independent of $d$
  note: length of shortest path does change by $d(\text{end of } P) - d(\text{start of } P)$
Flow Representation 4/4

- $X =$ set of faces adjacent to separator $C$
  - $= \text{set of endpoints of dual arcs of } C$
Flow Representation 4/4
Flow Representation 4/4

- $X$ = set of faces adjacent to separator $C$
  = set of endpoints of dual arcs of $C$
- $H$ - dual graph without dual arcs of $C$
Flow Representation 4/4

- \( X = \) set of faces adjacent to separator \( C \)
  = set of endpoints of dual arcs of \( C \)
- \( H \) - dual graph without dual arcs of \( C \)
- any shortest path in dual graph can be decomposed into:
  - shortest paths in \( H \)
  - dual arcs of \( C \)
Flow Representation 4/4

- $X = \text{set of faces adjacent to separator } C$
  - $= \text{set of endpoints of dual arcs of } C$
- $H$ - dual graph without dual arcs of $C$
- any shortest path in dual graph can be decomposed into:
  - shortest paths in $H$
  - dual arcs of $C$
- precompute all-pair shortest paths between nodes of $X$ in $H$
  - can be done in $O(n \log n)$ time [Klein SODA’05]
  - these shortest paths do not change
  - for $x,y \in X$, length of $x$-to-$y$ path changes by $d(x) - d(y)$
Flow Representation 4/4

- $X = \text{set of faces adjacent to separator } C$
  $= \text{set of endpoints of dual arcs of } C$
- $H$ - dual graph without dual arcs of $C$
- any shortest path in dual graph can be decomposed into:
  - shortest paths in $H$
  - dual arcs of $C$
- precompute all-pair shortest paths between nodes of $X$ in $H$
  - can be done in $O(n \log n)$ time [Klein SODA’05]
  - these shortest paths do not change
  - for $x, y \in X$, length of $x$-to-$y$ path changes by $d(x) - d(y)$
- suffices to maintain face labels for $X$ and explicit flow for $C$
Efficient Implementation

- precompute all-pair shortest paths between nodes of $X$ in $H$ \( \mathcal{O}(n) \) pairs
- maintain:
  - face labels for $X$ \( \mathcal{O}(\sqrt{n}) \) faces
  - explicit flow for $C$ \( \mathcal{O}(\sqrt{n}) \) arcs
- can implement Dijkstra’s algorithm with this representation in \( \mathcal{O}(\sqrt{n} \log^2 n) \) time using a modification of a data-structure of Fakcharoenphol and Rao [FOCS’01]

running time: \( \mathcal{O}(\sqrt{n}) \cdot \mathcal{O}(\sqrt{n} \log^2 n) = \mathcal{O}(n \log^2 n) \)
• with compact representation we have:
  - explicit flow $f$ on all arcs of $C$
  - accumulated face labels only for faces adjacent to $C$
• need to extend face labels to all faces
• can be done using one more shortest-path computation in the dual which takes linear time
Recall High-Level Algorithm

- find separator
- recursive problem (almost): eliminate residual paths
  - from sources to sinks
  - from sources to separator
  - from separator to sinks
- eliminate residual paths from + to - on separator
- return flow from + to sources and from sinks to -

running time: $O(n \log^3 n)$
Open Questions/Directions

• can running time be improved?  
  (bottleneck is Fakcharoenphol and Rao’s data structure and its modification)
• can this technique be adapted to bounded-genus graphs?
• implementation
Thank You!