Maximum Integer Flow in Directed Planar Graphs with Vertex Capacities and Multiple Sources and Sinks

Yipu Wang

University of Illinois at Urbana-Champaign

Jan 6, 2019
Maximum flow problem

Input:
- Directed graph \( G = (V, E) \)
- source \( s \in V \), sink \( t \in V \)
- capacity \( c : E \to \mathbb{R}_{\geq 0} \)

Output: a flow \( f : E \to \mathbb{R} \) such that
- \( \sum_{uv} f(uv) = \sum_{vw} f(vw) \quad \forall v \in V \setminus \{s, t\} \) (flow conservation)
- \( 0 \leq f(e) \leq c(e) \quad \forall e \in E \) (arc capacities)
- \( \sum_{sv} f(sv) - \sum_{us} f(us) \) is maximized

Solvable in \( O(mn) \) time [Orlin ’13]
Our maximum flow problem

- $G$ is planar
- $S$ is a set of sources, $T$ is a set of sinks
  - Maximize
  
  \[ \sum_{s \in S} \left( \sum_{sv} f(sv) - \sum_{us} f(us) \right) \]

- $k = |S| + |T|
- Each vertex has a capacity
  - \[ \sum_{uv} f(uv) \leq c(v) \quad \forall v \in V \setminus (S \cup T) \] (vertex capacities)

Applications in image processing/computer vision/vertex-disjoint paths.
Can we do better than $O(n^2 / \log n)$ time?
Past results

- Max \( st \)-flow in planar digraphs with vertex capacities can be solved in \( O(n \log n) \) time [Kaplan & Nussbaum '11]
- Reduction that turns multiple sources/sinks into single source/sink does not preserve planarity
Past results

- Max flow in planar digraphs with only $\alpha$ vertex capacities can be solved in $O(\alpha^3 n \log^3 n)$ time [Borradaile, Klein, Mozes, Nussbaum, Wulff-Nilsen ’17]
- Reduction that eliminates vertex capacities does not preserve planarity [Ford & Fulkerson ’62]
Previously, no near-linear-time algorithms were known, even for unit capacities when $k = 3$

<table>
<thead>
<tr>
<th></th>
<th>Previous result</th>
<th>Our result</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unit capacities</strong></td>
<td>$\tilde{O}(n^{10/7})$ $[\text{M'{a}dry '13}]$</td>
<td>$O(n \log^3 n + kn)$</td>
</tr>
<tr>
<td><strong>Integer capacities</strong></td>
<td>$O(n^{3/2} \log n \log U)$ $[\text{Goldberg/Rao '98}]$</td>
<td>$O(k^5 n \text{ polylog}(nU))$</td>
</tr>
<tr>
<td>$\leq U$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Real capacities</strong></td>
<td>$O(n^2 / \log n)$ $[\text{Orlin '13}]$</td>
<td>$O(n \log n)$ if $k = 3$</td>
</tr>
</tbody>
</table>

Our strategy: Extend Kaplan/Nussbaum’s $O(n \log n)$-time algorithm for single source/sink
Assume for this talk that max degree $= 4$

A *saddle* is a vertex where incident arcs change direction $\geq 4$ times

A vertex is *bad* if its capacity is being violated.

The *excess* of a bad vertex is the amount by which its capacity is being violated.

The *excess* of a flow is the maximum excess of a vertex.
Preliminaries

$\overline{G}$: replace each $v$ with edge of capacity $c(v)$

$G^\circ$: replace each $v$ with undirected cycle of capacity $c(v)/2$
Algorithm for single source/sink

**Theorem (Kaplan & Nussbaum ’11)**

*Can find max flow in directed planar graphs with vertex capacities and a single source/sink in \(O(n \log n)\) time.*

Idea: Find a maximum flow \(f^*\) in \(G^*\), “project” this flow back into \(G\) to get a flow \(f\)

**Lemma (Khuller & Naor ’94)**

*If single source/sink, then \(G\) and \(G^*\) have the same maximum flow value*

Problem: \(f\) may violate vertex capacities at its saddles
Problem: $f$ may violate vertex capacities at its saddles
Getting rid of saddles in $f$

Lemma (Guattery & Miller ’92)

Any planar DAG with $k$ sources and sinks has at most $k - 2$ saddles
Getting rid of saddles in \( f \)

Vertex capacities are only violated at saddles, so:

**Lemma**

If \( f^\circ \) is a max flow in \( G^\circ \) such that its projection \( f \) to \( G \) is acyclic, then \( f \) has at most \( k - 2 \) saddles (and thus violates at most \( k - 2 \) vertex capacities).

We can find such an \( f^\circ \), essentially by cancelling flow-cycles:

**Lemma (Kaplan & Nussbaum '11)**

In \( O(n) \) time, we can convert any flow \( f^\circ \) in \( G^\circ \) to another flow in \( G^\circ \) of the same value as \( f^\circ \) whose projection to \( G \) is acyclic.
Kaplan and Nussbaum’s $O(n \log n)$-time algorithm for single source and sink:

- Find max flow $f^\circ$ in $G^\circ$. ($O(n \log n)$ time)
- Convert $f^\circ$ to a max flow $f_1^\circ$ in $G^\circ$ whose projection $f_1$ to $G$ is acyclic. ($O(n)$ time)
- Return $f_1$. 
Problems:

- Value of max flow in $G^\circ$ might be larger than value of max flow in $G$
- We only know how to find a max flow $f^\circ$ in $G^\circ$ whose projection $f$ in $G$ violates at most $k - 2$ vertex capacities

On the other hand, if a vertex $v$ is bad, then excess of $v$ is at most $c(v)$
Find a maximum flow $f^\circ$ in $G^\circ$ whose restriction $f$ to $G$ violates at most $k - 2$ vertex capacities ($O(n \log^3 n)$ time)

2. From $f$, remove one unit of flow through each bad vertex, to get a flow $f'$ satisfying vertex capacities ($O(n)$ time)

3. In residual graph of $\overline{G}$ with respect to $f'$, find max flow $f''$ using Ford-Fulkerson algorithm ($O(kn)$ time)

4. Return $f' + f''$

Total: $O(n \log^3 n + kn)$ time
Let $\lambda^*$ be the value of max flow in $G$.

- Guess $\lambda^*$ using binary search
  1. Suppose the guess is $\lambda$
  2. Find a flow $f^\circ$ in $G^\circ$ of value $\lambda$ such that its projection $f$ in $G$
     violates at most $k - 2$ vertex capacities
  3. While $\text{excess}(f) > 2k$
     - “Improve” $f$ (i.e., cut $\text{excess}(f)$ by factor $k/(k - 1)$)
  4. Get rid of remaining excess using idea from unit-capacity case.
Integer capacity case

Let $\lambda^*$ be the value of max flow in $G$.

- Guess $\lambda^*$ using binary search
  1. Suppose the guess is $\lambda$
  2. In $G^\circ$: find a flow $f^\circ$ of value $\lambda$ s.t. $f$ acyclic
  3. While $\text{excess}(f) > 2k$
     - “Improve” $f$ (i.e., cut $\text{excess}(f)$ by factor $k/(k - 1)$)
  4. Get rid of remaining excess using idea from unit-capacity case

Running time analysis

- While-loop takes $O(k^4 n \log^3 n)$ time per iteration, $O(k \log U)$ iterations.
- Binary search for $\lambda^*$ contributes $\log(nU)$ factor.
- Step 4 takes $O(k^2 n)$ time
- Total time $O(k^5 n \log^5 (nU))$. 
Let $x_1, \ldots, x_{k-2}$ be bad vertices.

First we want circulations $\phi_1, \ldots, \phi_{k-2}$ where $\phi_i$ eliminates excess flow through $x_i$ without increasing flow through other bad vertices.

To compute $\phi_i$, find a flow in a modified residual graph $H_i$.

$H_i$ is a graph that can become planar after removing $O(k)$ vertices (i.e., an $O(k)$-apex graph), so computing $\phi_i$ takes $O(k^3 n \log^3 n)$ time. [Borradaile et al. ’17]
Construction of $H_i$

First construct $G^\times$ from $G$ as a “hybrid” of $\overline{G}$ and $G^o$

- good vertex $v$ becomes cycle
- bad vertex $x$ becomes arc

$H_i$ is the residual graph of $G^\times$ with respect to $f^\times$, except that:

- $f$ in $G$
- $H_i$ with source $x_i^{in}$, sink $x_i^{out}$
Let $\gamma = \phi_1 + \cdots + \phi_{k-2}$

- $f$ has excess on $x_1, \ldots, x_{k-2}$ but no excess on other vertices.
- $f + \gamma$ has no excess on $x_1, \ldots, x_{k-2}$ but has excess on other vertices.
- Take a weighted average: $\text{excess}(f + \gamma/k)$ is at most $\frac{k-1}{k} \text{excess}(f)$.
- Convert $(f + \gamma/k)^\circ$ to a flow of the same value whose projection to $G$ is acyclic; projection is desired improved $f$. 
Open problems

- Surface graphs or minor-free graphs
- Unit capacities $k$ not fixed
- Real capacities with fixed $k > 3$

Thanks to Microsoft Research for SIAM Student Travel Award