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

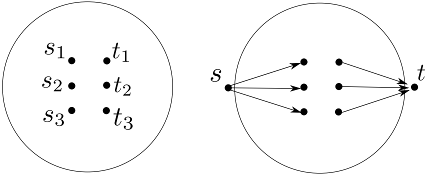
- G is planar
- S is a set of sources, T is a set of sinks
  - Maximize
    $$\sum_{s \in S} \left(\sum_{sv} f(sv) - \sum_{us} f(us) \right)$$
  - $k = |S| + |T|$
- Each vertex has a capacity
  - $\sum_{uv} f(uv) \leq c(v) \quad \forall v \in V \setminus (S \cup T)$ (vertex capacities)

Applications in image processing/computer vision/vertex-disjoint paths.

Can we do better than $O(n^2/\log n)$ time?

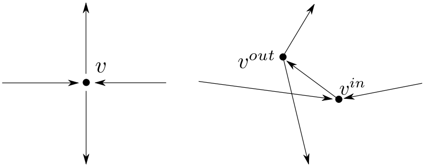
Past results

- 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 α 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]



Our results

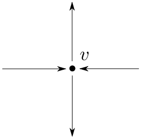
Previously, no near-linear-time algorithms were known, even for unit capacities when $k = 3$

	Previous result	Our result
Unit capacities	$\tilde{O}(n^{10/7})$ [Mądry '13]	$O(n \log^3 n + kn)$
Integer capacities $\leq U$	$O(n^{3/2} \log n \log U)$ [Goldberg/Rao '98]	$O(k^5 n \text{ polylog}(nU))$
Real capacities	$O(n^2 / \log n)$ [Orlin '13]	$O(n \log n)$ if $k = 3$

Our strategy: Extend Kaplan/Nussbaum's $O(n \log n)$-time algorithm for single source/sink

Preliminaries

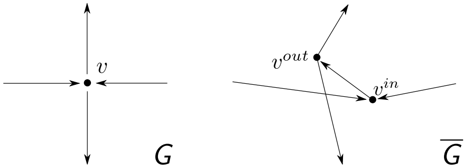
- Assume for this talk that max degree $= 4$
- A *saddle* is a vertex where incident arcs change direction ≥ 4 times



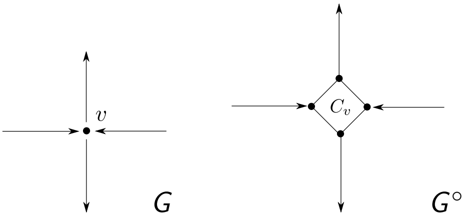
- A vertex is *bad* if its capacity is being violated
- The *excess* of a bad vertex is the amount by which its capacity is being violated
- The *excess* of a flow is the maximum excess of a vertex

Preliminaries

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



G°: replace each v with undirected cycle of capacity $c(v)/2$



Algorithm for single source/sink

Theorem (Kaplan & Nussbaum '11)

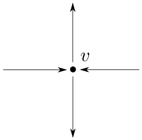
Can find max flow in directed planar graphs with vertex capacities and a single source/sink in $O(n \log n)$ time.

Idea: Find a maximum flow f° in G°, "project" this flow back into G to get a flow f

Lemma (Khuller & Naor '94)

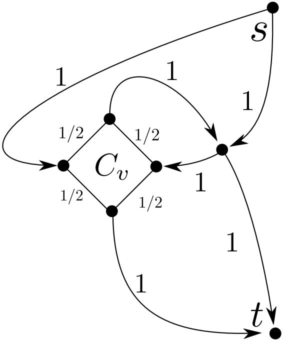
If single source/sink, then G and G° have the same maximum flow value

Problem: f may violate vertex capacities at its saddles

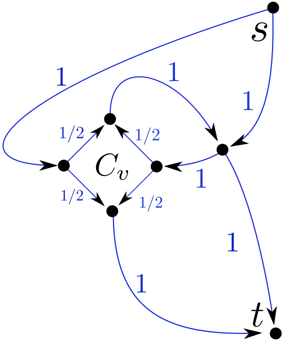


Algorithm for single source/sink

Problem: f may violate vertex capacities at its saddles



G°

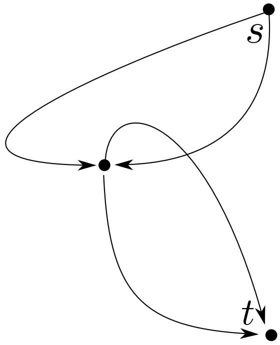


f° in G°

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° is a max flow in G° 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°, essentially by cancelling flow-cycles:

Lemma (Kaplan & Nussbaum '11)

In $O(n)$ time, we can convert any flow f° in G° to another flow in G° of the same value as f° whose projection to G is acyclic.

Algorithm for single source/sink

Kaplan and Nussbaum's $O(n \log n)$-time algorithm for single source and sink:

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

Multiple source/sink case

Problems:

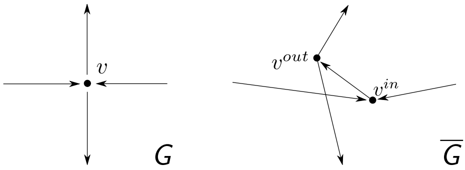
- Value of max flow in G° might be larger than value of max flow in G
- We only know how to find a max flow f° in G° whose projection f in G violates at most $k-2$ vertex capacities

On the other hand, if a vertex v is bad, then excess of v is at most $c(v)$

Unit capacity case

1. Find a maximum flow f° in G° 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 $\overline{f'}$, find max flow $\overline{f''}$ using Ford-Fulkerson algorithm ($O(kn)$ time)
4. Return $f' + f''$

Total: $O(n \log^3 n + kn)$ time



Integer capacity case

Let λ^* be the value of max flow in G.

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

Integer capacity case

Let λ^* be the value of max flow in G.

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

Running time analysis

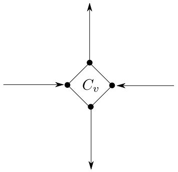
- While-loop takes $O(k^4 n \log^3 n)$ time per iteration, $O(k \log U)$ iterations.
- Binary search for λ^* contributes $\log(nU)$ factor.
- Step 4 takes $O(k^2 n)$ time
- Total time $O(k^5 n \log^5(nU))$.

Improvement phase strategy

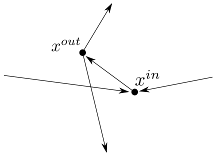
- Let $x_1, \ldots, x_{k-2}$ be bad vertices.
- First we want circulations $\phi_1, \ldots, \phi_{k-2}$ where ϕ_i eliminates excess flow through x_i without increasing flow through other bad vertices
- To compute ϕ_i, find a flow in a modified residual graph H_i
- H_i is a graph that can become planar after removing $O(k)$ vertices (i.e., an $O(k)$-apex graph), so computing ϕ_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°

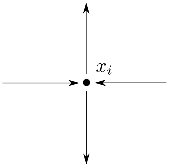


good vertex v becomes cycle

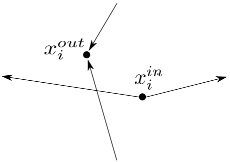


bad vertex x becomes arc

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}

Improvement phase strategy (cont.)

- 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: $\mathit{excess}(f + \gamma/k)$ is at most $\frac{k-1}{k}\mathit{excess}(f)$
- Convert $(f + \gamma/k)^\circ$ to a flow of the same value whose projection to G is acyclic; projection is desired improved f

Conclusion

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