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

Yipu Wang*

Abstract

We consider the maximum flow problem in directed planar graphs with capacities on both vertices and arcs and with multiple sources and sinks. We present three algorithms when the capacities are integers. The first algorithm runs in $O(\min\{k^2 n \log n, n\log^3 n + kn\})$ time when all capacities are bounded by a constant, where n is the number of vertices in the graph and k is the number of terminals. This algorithm is the first to solve the vertex-disjoint paths problem in near-linear time when k is fixed but larger than 2. The second algorithm runs in $O(k^5 n \text{ polylog}(nU))$ time, where U is the largest finite capacity of a single vertex. Finally, when $k = 3$, we present an algorithm that runs in $O(n \log n)$ time; this algorithm works even when the capacities are arbitrary reals. Our algorithms improve on the fastest previously known algorithms when k is fixed and U is bounded by a polynomial in n. Prior to this result, the fastest algorithms ran in $O(n^2/\log n)$ time for real capacities, $O(n^{3/2}\log n \log U)$ for integer capacities, and $\tilde{O}(n^{10/7})$ for unit capacities, even when $k = 3$.

1 Introduction

Finding a maximum flow in directed graphs is a well-studied problem with applications in many fields. The problem remains interesting even in directed planar graphs, which are graphs that can be embedded in the plane without crossing arcs. Such graphs arise in, for example, rail traffic models and VLSI design.

Typically, the maximum flow problem asks us to route some commodity along arcs with capacities, which limit the amount of commodity that can go through the arc. In this paper we are concerned with the case where vertices of the graph also have capacities, which limit the amount of commodity that can go through that vertex. When all the arc and vertex capacities are unit, we get the *vertex-disjoint paths problem*.

In general graphs, adding capacities to the vertices does not make the problem any harder because of a reduction first suggested by Ford and Fulkerson [5]. For each vertex v with finite capacity c, we do the following. Replace v with two vertices v^{in} and v^{out}, and add an arc of capacity c directed from v^{in} to v^{out}. All arcs that were directed into v are directed into v^{in} instead, and all arcs that were directed out of v are directed out of v^{out} instead. Unfortunately, this reduction does not preserve planarity. Consider the complete directed graph on four vertices. This graph is planar, but if we apply the reduction of Ford and Fulkerson on any single vertex, we get a graph whose underlying undirected graph is K_5, which is not planar by Kuratowski's Theorem.

Prior work has only been able to deal with the cases where there is a single source and sink or when the number of vertices with capacities is fixed. Khuller and Naor [12] were the first to consider the case where there is a single source and sink. Currently, the best known algorithm for this case is due to Kaplan and Nussbaum [10], who described an algorithm for maximum flow in directed planar graphs with vertex capacities that runs in $O(n \log n)$ time. (Here n is the number of vertices in the input graph). In doing so, they fixed a flaw in a paper of Zhang, Liang and Chen [18]. They also give an algorithm that runs in $O(n)$ time when all vertex and arc capacities are unit, solving the vertex-disjoint paths problem in directed planar graphs with a single source and sink. Zhang, Liang, and Chen [18] described an algorithm that finds maximum flows in undirected st-planar graphs with vertex capacities in $O(n)$ time. (A planar graph is st-planar if the source and the sink are on the same face.)

In the case of multiple sources and sinks, Borradaile et al. give an algorithm that runs in $O(\alpha^3 n \log^3 n)$ time, where α is the number of vertex capacities [1]. For arbitrary numbers of terminals and vertex capacities, the best-known algorithm prior to this paper uses the Ford-Fulkerson reduction described earlier, connects a super-source to all sources, connects all sinks to a supersink, and then in the resulting graph applies either Madry's algorithm [15] for finding maximum flows in unit-capacity networks, Goldberg and Rao's algorithm [6] for finding maximum flows in integer-capacity networks, or Orlin's algorithm [16] for finding maximum flows in sparse real-capacity networks. The resulting algorithm runs in $\tilde{O}(n^{10/7})$ time for unit

*University of Illinois at Urbana-Champaign, ywang298@illinois.edu

capacities, $O(n^{3/2} \log n \log U)$ time for integer capacities where U is the largest capacity, and $O(n^2/\log n)$ time for real capacities.

Maximum flows in directed planar graphs *without* vertex capacities can be computed in near-linear time [1], and one expects to be able to achieve the same time bound even when the graph has arbitrarily many vertex capacities. However, doing so seems to be difficult. The techniques for computing flows in planar graphs without vertex capacities rely heavily on the use of residual graphs, which do not exist when there are vertex capacities. The only tool we have for dealing with vertex capacities in planar graphs is Kaplan and Nussbaum's reduction, but that reduction fails when there are multiple sources and sinks because of *saddles*, which we explain in section 3. Even for the case where there are two sources and one sink and all vertex capacities are unit, no near-linear-time algorithms were previously known.

In this paper, we extend Kaplan and Nussbaum's algorithm to directed planar graphs with integer capacities and a fixed number of sources and sinks. The key observation is that when there are multiple sources and sinks, applying their algorithm results in a flow that is infeasible at only $k-2$ vertices, where k is the number of terminals. For each of these infeasible vertices, we define the excess of the vertex to be the amount by which it is infeasible, and we show that the sum of the excesses of all the infeasible vertices is at most $(k-2)U$. This means that when U is small, the flow returned by Kaplan and Nussbaum's algorithm is close to feasible. We exploit this observation to obtain the following:

THEOREM 1.1. *Let G be a directed planar graph with k terminals and with integer capacities on both arcs and vertices. If all capacities are bounded by a constant, then we can find a maximum flow in G in $O(\min\{k^2 n \log n, n \log^3 n + kn\})$ time.*

Thus when k is fixed, we can solve the vertex-disjoint paths problem in directed planar graphs with multiple sources and sinks in near-linear time.

Our second algorithm deals with the case where U may be unbounded. The basic idea is a scaling algorithm. First we guess the value of the maximum flow using binary search; this increases the running time of the algorithm by a factor of $O(\log(nU))$. Starting with a flow with $k-2$ infeasible vertices, we find a way to improve the flow that decreases the maximum excess of the vertices by some factor that depends only on k. The improved flow has the same value as the original flow. We show that after $O(k \log(kU))$ improvement phases, each infeasible vertex has $O(k)$ excess. As in the first algorithm, we exploit this observation to obtain the following:

THEOREM 1.2. *Let G be a directed planar graph with k terminals and with integer capacities on both arcs and vertices. If U is the maximum capacity of a single vertex or arc, then we can find a maximum flow in G in $O(k^5 n \ \mathit{polylog}(nU))$ time.*

Our third algorithm deals with the special case where $k = 3$. In this case, the fact that there is only one infeasible vertex considerably simplifies the problem, since we can just focus on decreasing the excess of this one vertex without worrying about trade-offs. (Roughly speaking, if there is more than one infeasible vertex, we have to consider that decreasing the excess of one vertex could increase the excess of another vertex.) We show that we can modify our second algorithm such that only one improvement phase is necessary. This third algorithm works even if the capacities are arbitrary real numbers instead of integers.

THEOREM 1.3. *Given a directed planar graph G with three terminals and with capacities on both arcs and vertices, we can find a maximum flow in G in $O(n \log n)$ time.*

The outline of this paper is as follows. In section 2, we give some basic definitions and describe some basic graph constructions that will be used in the paper. In section 3, we prove the structural properties that show that Kaplan and Nussbaum's algorithm almost works when there are multiple sources and sinks. In section 4, we describe the algorithm for the case where capacities are integers bounded by a constant. In section 5, we use this algorithm to solve the case of arbitrary integer capacities. In section 6, we describe the modifications to the algorithms that are necessary for the case when $k = 3$ and the capacities are arbitrary reals.

2 Preliminaries

In this paper, G is a simple directed plane graph with vertex set $V(G)$, arc set $E(G)$, and face set $F(G)$. Let n be the number of vertices in G; it is well known that Euler's formula implies $|E(G)| = O(n)$. For any vertex $v \in V(G)$, let $\deg_G(v)$ denote the degree of v in G. If G is a graph and $W \subseteq V(G)$, then $G \setminus W$ is the induced subgraph of G with vertex set $V(G) \setminus W$. For any integer N, let $[N] = \{1, \ldots, N\}$.

We use (u, v) to denote an arc or directed edge that is directed from u to v. A *path* is a sequence of arcs $((u_1, v_1), \ldots, (u_p, v_p))$ such that $v_i = u_{i+1}$ for all $i \in [1, p-1]$. Such a path *starts* at u_1 and *ends* at v_p. If in addition $v_p = u_1$ then P is a *cycle*. A path P contains a vertex v if one of the arcs of P has v as an endpoint. Thus we will sometimes view

paths and cycles as sets of vertices or as sets of arcs instead of as sequences of arcs. For any $v \in V$, let $in(v) = \{(u,v) \mid (u,v) \in E(G)\}$ be the set of *incoming arcs* of v, and let $out(v) = \{(v,u) \mid (v,u) \in E(G)\}$ be the set of *outgoing arcs* of v. Similarly, if W is a set of vertices, then $in(W) = \{(u,v) \in E(G) \mid u \notin W, v \in W\}$ and $out(W) = \{(u,v) \in E(G) \mid u \in W, v \notin W\}$.

The *reversal* of any arc (u,v), denoted $rev((u,v))$, is (v,u). We may assume without loss of generality that if $e \in E(G)$, then $rev(e) \in E(G)$, and both e and $rev(e)$ are embedded together. If P is a path $(e_1, \ldots, e_p)$, then the *reversal* of P, denoted $rev(P)$, is $(rev(e_p), \ldots, rev(e_1))$.

Two disjoint subsets of $V(G)$ are special: S is a set of *sources* and T is a set of *sinks*. Vertices that are in either S or T are called *terminals*. Let k be the number of terminals.

Each arc e has a non-negative capacity $c(e)$ and each non-terminal vertex v has a positive capacity $c(v)$. Capacities may be infinite, and we can assume without loss of generality that terminals have infinite capacity: if a source s has finite capacity c, then we can add a node s', an arc (s', s) of capacity c, replace s with s' in S, and let s' have infinite capacity, all while preserving planarity. A similar reduction eliminates finite capacities on the sinks.

Flows. A *flow network* is a directed graph that has a capacity on each arc and vertex, a set of sources, and a set of sinks. Suppose G is a flow network with capacity function $c : E(G) \cup V(G) \to [0, \infty)$, source set S, and sink set T. Let $f : E(G) \to [0, \infty)$. To lighten notation, in this paper we will write $f(u,v)$ instead of $f((u,v))$ for any arc (u,v). For each vertex v, let

$$f^{in}(v) = \sum_{e \in in(v)} f(e) \quad \text{and} \quad f^{out}(v) = \sum_{e \in out(v)} f(e).$$

Similarly, if W is a set of vertices, then let

$$f^{in}(W) = \sum_{e \in in(W)} f(e) \quad \text{and} \quad f^{out}(W) = \sum_{e \in out(W)} f(e).$$

The function f is a *flow in* G if it satisfies the following *flow conservation constraints*:

$$f^{in}(v) = f^{out}(v) \quad \forall v \in V(G) \setminus (S \cup T)$$

A flow is *feasible* if in addition it satisfies the following two types of constraints:

$$\begin{array}{ll} 0 \le f(e) \le c(e) & \forall e \in E(G) \\ f^{in}(v) \le c(v) & \forall v \in V(G) \setminus (S \cup T) \end{array}$$

Constraints of the first type are *arc capacity constraints* and those of the second type are *vertex capacity constraints*. A flow f *routes* $f(e)$ units of flow through the arc e. An arc $e \in in(v)$ *carries flow into* v if $f(e) > 0$, and an arc $e' \in out(v)$ *carries flow out of* v if $f(e') > 0$. We assume that $\min\{f(e), f(rev(e)\} = 0$ for every arc e.

In the *maximum flow problem*, we are trying to find a feasible flow f with maximum *value*, where the value $|f|$ of a flow f is defined as

$$|f| = \sum_{s \in S} (f^{out}(s) - f^{in}(s)).$$

When all the vertex and arc capacities are 1, the maximum flow problem becomes the *vertex-disjoint paths problem*.

Let $\mathrm{val}(G)$ be the value of the maximum flow in a flow network G (which may have vertex capacities). A *circulation* is a flow of value 0. A circulation g is *simple* if $g^{in}(v) = g^{out}(v)$ for every terminal v. Non-simple circulations only exist if there are more than two terminals. A flow f has a *flow cycle* C if C is a cycle and $f(e) > 0$ for every arc e in C, and f is *acyclic* if it has no flow cycles. A flow cycle C of a flow f is *unit* if $f(e) = 1$ for every arc e in C. A flow f *saturates* an arc e if $f(e) = c(e)$. A flow is a *path-flow* if its support is a path.

We will often add two flows f and g together to obtain a flow $f+g$, or multiply a flow f by some constant c to get a flow cf. These operations are defined in the obvious way: for every arc e, we have

$$\begin{aligned} (f+g)(e) &= \max\{0, f(e) + g(e) - f(rev(e)) \\ &\qquad - g(rev(e))\} \\ (cf)(e) &= c \cdot f(e) \end{aligned}$$

k-apex graphs and G_{st}. A graph G is a *k-apex graph* if there are k vertices whose removal from the graph would make G planar. These k vertices are called *apices*.

Given a flow network with multiple sources and sinks, we can reduce the maximum flow problem to the single-source, single-sink case by adding a supersource s, supersink t, infinite-capacity arcs (s, s_i) for every $s_i \in S$, and infinite-capacity arcs (t_i, t) for every $t_i \in T$. Call the resulting flow network G_{st}. Finding a maximum flow in the original network G is equivalent to finding a maximum flow from s to t in G_{st}. The graph G_{st} is not necessarily planar but is a 2-apex graph. In this paper, we will work in G_{st} instead of G when we want circulations to be unions of flow cycles; in G, circulations can consist of source-to-source or sink-to-sink paths.

The flow graph f_G. Given a flow f in a flow network G, the *flow graph* of f is a graph f_G that contains all the vertices of G but only contains the arcs of G that carry non-zero flow; furthermore, each arc e

in f_G has weight $f(e)$. Depending on the context, we will interpret these arc weights as either capacities or flow.

The extended graph G°. Given a flow network G with vertex capacities, Kaplan and Nussbaum [10] defined the *extended graph* G° based on constructions of Khuller and Naor [12], Zhang, Liang, and Jiang [19], and Zhang, Liang, and Chen [18]. Starting with G_{st}, we replace each finitely capacitated vertex $v \in V(G_{st})$ with an undirected cycle of d vertices $v_1, \ldots, v_d$, where $d = |in(v)| + |out(v)|$ is the degree of v. Each edge in the cycle has capacity $c(v)/2$. (An undirected edge e with capacity $c(e)$ can be viewed as two arcs e and $rev(e)$, each with capacity $c(e)$, so G° can be viewed as a directed flow network.) We make every arc that was incident to v incident to some vertex v_i instead, such that each arc is connected to a different vertex v_i, the clockwise order of the arcs is preserved, and the graph remains planar. We also identify the new arc (u, v_i) or (v_i, u) with the old arc (u, v) or (v, u) and denote the cycle replacing v by C_v. The resulting graph G° has $O(n)$ vertices and arcs. See Figure 1.

This idea of eliminating vertex capacities in planar graphs by replacing each vertex with a cycle has also been used in the context of finding shortest vertex-disjoint paths in planar graphs [3].

The graph $\overline{G}$. Given a flow network G with vertex capacities, let $\overline{G}$ be the flow network obtained as follows: Starting with G_{st}, replace each capacitated vertex v with two vertices v^{in} and v^{out}, and add an arc of capacity $c(v)$ directed from v^{in} to v^{out}. All arcs that were directed into v are directed into v^{in} instead, and all arcs that were directed out of v are directed out of v^{out} instead. See Figure 1. It is well known that every feasible flow in G_{st} corresponds to a feasible flow in $\overline{G}$ of the same value, and vice versa. The graph $\overline{G}$ has $O(n)$ vertices and arcs.

Restrictions and extensions. Suppose G and H are flow networks such that every arc in G is also an arc in H. If f' is a flow in H, then the *restriction* of f' to G is the flow f in G defined by $f(e) = f'(e)$ for all arcs $e \in E(G)$. Conversely, if f is a flow in G, then an *extension* of f to H is any flow f' in H such that $f(e) = f'(e)$ for every arc $e \in E(G)$.

Every arc in G or G_{st} is an arc in both $\overline{G}$ and G°. Every feasible flow in $\overline{G}$ has a feasible restriction in G. Conversely, every feasible flow f in G has a feasible extension $\overline{f}$ in $\overline{G}$, by defining $\overline{f}(v^{in}, v^{out}) = f^{in}(v)$. Every feasible flow in G° has a restriction in G; this restriction is a flow but is not necessarily feasible. On the other hand, we have the following lemma:

LEMMA 2.1. *Every feasible flow f in G has an extension f° that is feasible in G°. Furthermore, we can find f° in $O(n \log^3 n)$ time.*

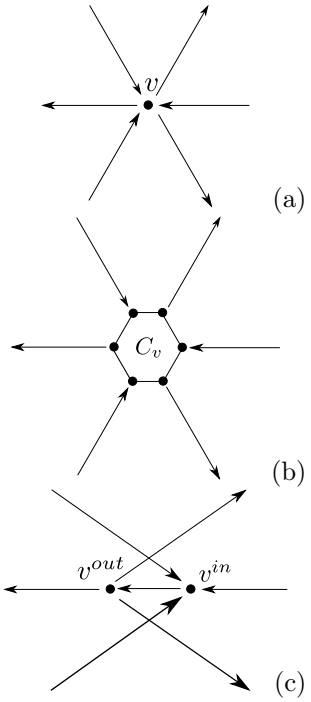

Figure 1: (a) capacitated vertex $v \in G$ with capacity $c(v)$ (b) corresponding cycle C_v in G°; each edge in C_v has capacity $c(v)/2$ (c) corresponding arc (v^{in}, v^{out}) in $\overline{G}$ with capacity $c(v)$

Proof. To show that f° exists, we use the well-known flow decomposition theorem, which states that any flow f in G can be decomposed into a sum of flows $f_1, \ldots, f_m$ such that for each i, the support of f_i is either a cycle or a path from a source to a sink. For each $i \in [m]$, let p_i be the support of f_i and let $u_i = |f_i|$.

For each capacitated vertex $w \in G$, we define f° on the cycle C_w in G° as follows: for each $i \in [m]$, if some arc in p_i carries u_i units of flow into a vertex x on C_w and another arc in p_i carries u_i units of flow out of a vertex x' on C_w, then we route $u_i/2$ units of flow clockwise along C_w from x to x' and $u_i/2$ units of flow counter-clockwise along C_w from x to x'. It is easy to see that f° satisfies conservation constraints. Since $f^{in}(C_w) \le c(w)$, no arc on C_w carries more than $c(w)/2$ units of flow, so f° is feasible.

We now describe how to find f°. We must define $f^\circ(e) = f(e)$ for all arcs $e \in E(G)$. We reduce the problem of finding f° on all other arcs to finding a flow in a planar flow network H. Let H be the subgraph of G° consisting of all cycles C_v where v is a capacitated vertex in G; it suffices to define f° on the arcs of H. Recall that for all $v \in V(G)$, the vertices in C_v are

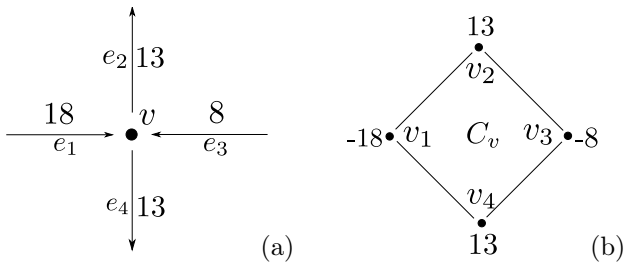

Figure 2: Extending a flow from G to G°. (a) An example of f at v; arcs are labeled with their flow values (b) H at C_v with terminals labeled with their demand values; v_1 and v_3 are sources; v_2 and v_4 are sinks

$v_1, \ldots, v_d$ in clockwise order, where $d = \deg_G(v)$. For each vertex v_i in H, let $e_{i,v}$ be the unique arc in G incident to v_i. When it is clear what vertex v is, we will write e_i instead of $e_{i,v}$. For each $v \in V(G)$ and $i \in [\deg_G(v)]$, let

$$demand(v_i) = \begin{cases} -f(e_i) & \text{if } e_i \in in(v_i) \\ f(e_i) & \text{if } e_i \in out(v_i) \end{cases}$$

That is, $demand(v_i)$ is the net amount of flow that f° carries out of v_i using only arcs in $E(G)$. For each vertex v_i such that $demand(v_i)$ is negative, let v_i be a source in H; similarly, if $demand(v_i)$ is positive, let v_i be a sink in H. For each $v \in V(G)$, $\sum_{i=1}^{\deg_G(v)} demand(v_i) = 0$. See Figure 2.

Since f° exists, there exists a flow f_H in H such that $f_H^{out}(v_i) = -demand(v_i)$ for every source v_i and $f_H^{in}(v_i) = demand(v_i)$ for every sink v_i. To actually find f_H, we do the following. For each source v_i in H, we add a vertex v_i' that will be a source instead of v_i, and we add an arc (v_i', v_i) with capacity $-demand(v_i)$; similarly, for each sink v_j in H, we add a vertex v_j' that will be a sink instead of v_j, and we add an arc (v_j, v_j') with capacity $demand(v_j)$. Then f_H is an acyclic maximum flow in the resulting network. The restriction of f_H to H is exactly f° on the arcs of H. Finding f_H requires finding a maximum flow in a union of disjoint "suns" with multiple sources and sinks, where a sun is a cycle in which each vertex has a pendant arc appended to it. This can be done in, say, $O(n \log^3 n)$ time using the algorithm of Borradaile et al. [1]. Simpler and more intuitive algorithms exist but would take longer to describe.

The residual graph. If f is a flow in a flow network G with capacity function c and without vertex capacities, then the *residual capacity* of an arc e with respect to f and c, denoted $c_f(e)$, is $c(e) - f(e) + f(rev(e))$. The *residual graph of G with respect to f and c* (or just the *residual graph of G with respect to f* when c is the capacity function given as input) has the same vertices and arcs as G, but each arc e has capacity $c_f(e)$. A *residual arc* of G with respect to f is an arc with positive residual capacity, a *residual path* is a path made up of residual arcs, and a *residual cycle* is a cycle made up of residual arcs. It is well known that a flow f is a maximum flow in a graph G if the residual graph of G with respect to f does not have any residual paths from a source to a sink.

Fractional and integer flows. A flow f° in G° is an *integer flow* if $f^\circ(e)$ is an integer for every arc e in G; otherwise, f° is *fractional*. A flow in $\overline{G}$ is *integer* if it is integer-valued on all arcs of $\overline{G}$ and is *fractional* otherwise. We now describe how to convert a fractional flow f° in G° to an integer flow f_1° in G° of the same value, assuming that $|f^\circ|$ is an integer and G° has integer arc capacities. Furthermore, for each $v \in V(G)$, we will have

$$(f_1^\circ)^{in}(C_v) \le \left\lceil (f^\circ)^{in}(C_v) \right\rceil$$

First let f be the restriction of f° to G, and let $\overline{f}$ be the extension of f to $\overline{G}$. Since $\overline{G}$ has $O(n)$ arcs, results by Lee et al. [13] and by Kang and Payor [9] imply the following.

LEMMA 2.2. *Let $\overline{f}$ be a fractional flow in $\overline{G}$ such that $|\overline{f}|$ is an integer. Then there exists an integer flow $\overline{f_1}$ in $\overline{G}$ of the same value as $\overline{f}$ such that $\overline{f_1}(e) \le \lceil \overline{f}(e) \rceil$ for every arc e in $\overline{G}$. Furthermore, we can find $\overline{f_1}$ in $O(n \log n)$ time.*

Compute $\overline{f_1}$ as in Lemma 2.2 and let f_1 be the restriction of $\overline{f_1}$ to G. Finally, we define f_1° to be an extension of f_1 to G°; by Lemma 2.1, we can do this in $O(n \log^3 n)$ time.

We need to show that f_1° is the desired integer flow. Since $\overline{f_1}$ is an integer flow in $\overline{G}$, f_1° is an integer flow in G°. Also, we have $|f_1^\circ| = |f_1| = |\overline{f_1}| = |\overline{f}| = |f| = |f^\circ|$. Lemma 2.2 implies that for every $v \in V(G)$, we have

$$\begin{aligned} &\overline{f_1}(v^{in}, v^{out}) \le \left\lceil \overline{f}(v^{in}, v^{out}) \right\rceil \\ &\implies f_1^{in}(v) \le \lceil f^{in}(v) \rceil \\ &\implies (f_1^\circ)^{in}(C_v) \le \left\lceil (f^\circ)^{in}(C_v) \right\rceil . \end{aligned}$$

We have proved the following lemma:

LEMMA 2.3. *Let f° be a fractional flow in G° such that $|f^\circ|$ is an integer. Then there exists an integer flow f_1° in G° of the same value as f° such that $(f_1^\circ)^{in}(C_v) \le \left\lceil (f^\circ)^{in}(C_v) \right\rceil$ for every vertex $v \in V(G)$. Furthermore, we can find f_1° in $O(n \log^3 n)$ time.*

Subroutines. Our algorithm uses several algorithms that compute maximum flows or circulations in

graphs *without* vertex capacities. First, we use an algorithm of Borradaile and Klein [2] [4] for finding maximum flows in directed planar graphs with a single source and sink in $O(n \log n)$ time. Second, we use Miller and Naor's observation [14] that maximum flow with multiple sources and sinks can be computed with a simple pair of nested for-loops: For all sources s_i, for all sinks t_j, compute the maximum flow from s_i to t_j in the current residual graph. Combining this with the algorithm of Borradaile and Klein [2] yields an algorithm for maximum flows in directed planar graphs with multiple sources and sinks that runs in $O(k^2 n \log n)$ time. Third, we use an algorithm of Borradaile et al. [1] for finding maximum flows in directed planar graphs with multiple sources and sinks in $O(n \log^3 n)$ time. Equivalently, the algorithm finds maximum flows in directed 2-apex graphs with a single source and sink if the source and sink are the only apices. Fourth, we use another algorithm by Borradaile et al. [1] that finds maximum flows in k-apex graphs with multiple sources and sinks in $O(k^3 n \log^3 n)$ time. Fifth, we use the classical Ford-Fulkerson augmenting-path algorithm that computes maximum flows in general directed graphs with integer capacities in $O(mU^*)$ time, where m is the number of arcs in the flow network and U^* is the value of the maximum flow. Finally, we implicitly use two algorithms that allow us to assume without loss of generality that certain flows are acyclic. The first is by Kaplan and Nussbaum [10]:

LEMMA 2.4. *Given a feasible flow f° in G°, we can compute in $O(n)$ time another feasible flow of the same value as f° whose restriction to G is feasible and acyclic by canceling flow-cycles.*

We describe this algorithm in more detail in Appendix A. Using this algorithm, we can assume that whenever we compute a flow in G°, the restriction of that flow to G is acyclic.

The second algorithm that we use implicitly is by Sleator and Tarjan [17]:

LEMMA 2.5. *Given a flow in a flow network with $O(n)$ vertices and arcs, we can compute another flow of the same value that is acyclic in $O(n \log n)$ time by canceling flow-cycles.*

Using this algorithm, we may assume that whenever we compute a flow in a graph, the computed flow is acyclic.

3 Saddles and excess

Suppose f° is a feasible flow in G° whose restriction f to G is acyclic. It is easy to see that f satisfies conservation and arc capacity constraints. In this section, we show that f violates at most $|S| + |T| - 2$ vertex capacity constraints.

Let f_G be the flow graph of f. For any vertex v in f_G, the *alternation number* of v, denoted by $\alpha(v)$, is the number of direction changes (i.e., from in to out or vice versa) of the arcs incident to v as we examine them in clockwise order. Thus $\alpha(u) = 0$ for all terminals u, and the alternation number of any vertex is even. A vertex v is a *saddle in f* if $\alpha(v) \geq 4$. We let $index(v)$ denote the *index* of v and define it by $index(v) = \alpha(v)/2 - 1$.

Since f_G is a plane directed acyclic graph, a result of Guattery and Miller [7] implies the following:

LEMMA 3.1. *If f_G has k_1 sources and k_2 sinks, then the sum of the indices of the saddles in f_G is at most $k_1 + k_2 - 2$.*

In particular, a vertex in f_G is a saddle if and only if it has positive index, so f_G has at most $k - 2$ saddles. The proof of Lemma 3.1 is simple and can be found in Appendix B.

A vertex $v \in V(G)$ is *infeasible* under the flow f if $f^{in}(v) > c(v)$ and is *feasible* otherwise. For any vertex $v \in V(G)$, let $\mathrm{ex}(f^\circ, v)$ and $\mathrm{ex}(f, v)$ denote the *excess* of the vertex v under f° or f:

$$\mathrm{ex}(f^\circ, v) = \mathrm{ex}(f, v) = \max\{0, f^{in}(v) - c(v)\}$$

The excess of a vertex is positive if and only if the vertex is infeasible. We also define $\mathrm{ex}(f^\circ) = \mathrm{ex}(f) = \max_{v \in V(G)} \mathrm{ex}(f, v)$. We say that f has excess $\mathrm{ex}(f, v)$ on v.

LEMMA 3.2. *Let $index(v)$ be defined for each vertex v in G using the flow graph f_G of f. For each vertex v in f_G, we have $ex(f, v) \leq index(v)c(v)$.*

Proof. Let f_G° be the flow graph of f°. We have $\alpha(v) = 2 \cdot index(v) + 2$. Thus, if we examine the arcs in f_G incident to v in clockwise order, there are $index(v) + 1$ groups of consecutive incoming arcs. Consider such a group of consecutive incoming arcs $(u_i, v), \ldots, (u_j, v)$ in f_G. We can view these as arcs $(u_i, v_i), \ldots, (u_j, v_j)$ in f_G°, where $v_i, \ldots, v_j$ are consecutive vertices in C_v. In f_G°, the only two arcs in $out(\{v_i, \ldots, v_j\})$ are (v_i, v_{i-1}) and (v_j, v_{j+1}), which have total capacity $c(v)$. Thus, for each vertex v in f_G, each group of consecutive incoming arcs in f_G has total weight at most $c(v)$. This shows that $f^{in}(v) \leq (index(v) + 1)c(v)$ for any vertex v, from which the lemma follows.

Combining Lemmas 3.1 and 3.2, we see that the sum of the excesses of all vertices under f is at most $(k-2)U$. Lemma 3.2 implies that f is only infeasible at saddles of f_G.

4 Bounded integer capacity case

Suppose that all vertex and arc capacities are integers less than some constant U. Let f° be an integral maximum flow in G°, and let f be its restriction to G. By Lemma 2.4 we may assume without loss of generality that f is acyclic. The flow f may be infeasible at up to $k-2$ vertices $x_1, \ldots, x_{k-2}$. By Lemma 3.1 and 3.2, the sum of the excesses of the infeasible vertices is at most $(k-2)U$. Computing f takes $O(n \log^3 n)$ time using the algorithm of Borradaile et al. [1] or $O(k^2 n \log n)$ time using $O(k^2)$ invocations of the algorithm of Borradaile and Klein [2]. After finding f, the algorithm has two steps.

Step 1. In this step, we remove $\mathrm{ex}(f, x_i)$ units of flow through each infeasible vertex x_i to get a feasible flow f_1 in G. The flow f_1 will have lower value than f. To do this, let f_G be the flow graph of f. The graph f_G is a directed acyclic graph; let $v_1, \ldots, v_n$ be a topological ordering of f_G. To remove all of the excess flow through an infeasible vertex x_i, we do the following:

- Push $\mathrm{ex}(f, x_i)$ units of flow back from x_i to the sources of f_G, as follows. Process the vertices in f_G in the order $v_n, \ldots, v_1$. To process a vertex v_j, check whether there is too much flow going through v_j (either because $v_j = x_i$ or because flow conservation is violated at v_j). If so, then decrease the flow on incoming arcs of v_j in f_G until there is no longer too much flow going into v_j (i.e., until the flow going into v_j is at most $c(v_j)$ if $v_j = x_i$, or until flow is conserved at v_j if $v_j \neq x_i$).

- Pull $\mathrm{ex}(f, x_i)$ units of flow back to x_i from the sinks of f_G, using a similar algorithm as in the previous bullet point.

Let f_1 be the resulting feasible flow. Removing excess flow through a vertex x_i takes $O(n)$ time, so step 1 takes $O(kn)$ time.

Step 2. Let $\overline{f_1}$ be the extension of f_1 to $\overline{G}$. In this step, we do the following:

- Compute a maximum flow $\overline{f_2}$ in the residual graph of $\overline{G}$ with respect to $\overline{f_1}$ using the classical Ford-Fulkerson algorithm.

- Return the restriction of $\overline{f_1} + \overline{f_2}$ to G.

Since $\overline{f_2}$ is a maximum flow in the residual graph of $\overline{G}$ with respect to $\overline{f_1}$, we see that $\overline{f_1} + \overline{f_2}$ is a maximum flow in $\overline{G}$. It follows that the restriction of $\overline{f_1} + \overline{f_2}$ to G is a maximum flow in G, as desired.

We have $val(\overline{G}) \leq val(G^\circ) = |f| \leq |f_1| + (k-2)U$. Thus the value of $\overline{f_2}$ is at most $(k-2)U$, so computing $\overline{f_2}$ takes $O(knU)$ time. Hence step 2 takes $O(knU)$ time. Thus if U is bounded by a constant, the entire algorithm runs in $O(\min\{k^2 n \log n, n \log^3 n + kn\})$ time.

5 Integer capacities and $k > 2$

Suppose all vertex and arc capacities are integers. Let $\lambda^* = val(G)$. The basic structure of the algorithm is as follows:

- Guess λ^* via binary search.

    1. Suppose we guess the value of the maximum flow of G to be λ. Find a flow f° in G° of value λ. By Lemma 2.4, we may assume that the restriction f of f° to G is acyclic.
    2. While $\mathrm{ex}(f) > 2k$, improve f.
    3. Fix f using the algorithm from section 4.

One can see that the algorithm has three main steps which we call *phases*. In phase 2, improving f means that we find a flow f_2 of the same value as f such that

$$\mathrm{ex}(f_2) \leq \left\lceil \frac{k-1}{k} ex(f) \right\rceil.$$

We then set f to be the new flow f_2. We will eventually show that a single improvement of f can be done in $O(k^4 n \log^3 n)$ time. In phase 3, fixing f means that we remove $\mathrm{ex}(f, x)$ units of flow through each infeasible vertex x to get flow f', extend f' to a flow $\overline{f'}$ in $\overline{G}$, and then use the Ford-Fulkerson algorithm to find a maximum flow $\overline{f''}$ in the residual graph of $\overline{G}$ with respect to $\overline{f'}$; we then set f to be the restriction of $\overline{f'} + \overline{f''}$ to G. To do the binary search, we use the fact that $\lambda \leq \lambda^*$ if the result of phase 3 is a feasible flow of value λ, and $\lambda > \lambda^*$ if either phase 2 fails or if the flow that results from phase 3 has value less than λ.

Before we describe how phase 2 is implemented, let us analyze the running time of the algorithm. If U is the maximum capacity of a single vertex, then $\lambda^* \leq nU$, so the binary search for λ^* only requires $O(\log(nU))$ guesses. Computing f° in phase 1 takes $O(n \log^3 n)$ time using the algorithm of Borradaile et al. [1]. By Lemma 3.2, at the beginning of phase 2, $\mathrm{ex}(f) \leq (k-2)U$. The following lemma shows that phase 2 takes $O(k^5 n \log^3 n \log(kU))$ time:

LEMMA 5.1. *After $O(k \log(kU))$ iterations of the while-loop in phase 2, $ex(f, x) \leq 2k$ for every vertex $x \in V(G)$.*

Proof. After each iteration, $\mathrm{ex}(f)$ decreases roughly by a factor $1 + 1/(k-1) \geq 1 + 1/k$. Thus we only require $O(\log_{1+1/k}(kU)) = O(\frac{\ln(kU)}{\ln(1+1/k)})$ iterations. For $k \geq 1$

we have

$$\begin{aligned} & e^{1/2} < (1+1/k)^k < e \\ \implies & 1/2 < k\ln(1+1/k) < 1 \\ \implies & \frac{1}{2k} < \ln(1+1/k) < \frac{1}{k}. \end{aligned}$$

This means that $O(k \log kU)$ iterations suffice.

In phase 3, the same reasoning as in Section 4 shows that computing $\overline{f'} + \overline{f''}$ takes $O(k^2 n)$ time. The total running time of the algorithm is thus

$$\begin{aligned} & O(\log(nU)[n\log^3 n + k^5 n \log^3 n \log(kU) + k^2 n]) \\ & = O(k^5 n \log^3 n \log kU \log nU) \\ & = O(k^5 n \text{ polylog}(nU)). \end{aligned}$$

The rest of this section describes one iteration of the while-loop in phase 2. Specifically, given a feasible flow f° whose restriction f to G has at most $k-2$ infeasible vertices, we compute a flow f_2° in G° whose restriction f_2 to G has at most $k-2$ infeasible vertices, each of which has excess at most $\left\lceil \frac{k-1}{k} ex(f) \right\rceil$. Let X be the set of infeasible vertices under f, and for each $x \in X$, define $ex_x = \text{ex}(f, x)$. The procedure that finds f_2° has three stages. In stage 1, we find a circulation g° such that $f^\circ + g^\circ$ is feasible in G° and $(f^\circ + g^\circ)^{in}(C_x) \leq c(x)$ for every $x \in X$. However, the restriction of $f^\circ + g^\circ$ to G may have large excesses on vertices not in X. To fix this, in stage 2 we use g° to compute a feasible flow f_1° in G° satisfying $\text{ex}(f_1^\circ) \leq \left\lceil \frac{k-1}{k} \text{ex}(f) \right\rceil$. Intuitively, f_1° approximates $f^\circ + g^\circ/k$ while being an integer circulation. In stage 3, we use Lemma 2.4 on f_1° to get a flow f_2° of the same value whose restriction to G is acyclic and has at most $k-2$ infeasible vertices. If $\lambda > \lambda^*$, then g° may not exist and stage 1 may fail; if $\lambda \leq \lambda^*$, then g° exists and all three stages will work.

5.1 Stage 1. To get g°, we first convert f° to a feasible flow $f^\times$ of the same value in a flow network $G^\times$ such that the restrictions of f° and $f^\times$ to G are equal. Then, we find a circulation $g^\times$ in $G^\times$ such that the restriction of $f^\times + g^\times$ to G has no excesses on the vertices of X. Finally, we convert $f^\times + g^\times$ to a feasible flow $f^\circ + g^\circ$ in G°, from which we get g°.

We construct $G^\times$ as follows. Starting with G°, we do the following for each vertex $x \in X$:

- Replace C_x with an arc (x^{in}, x^{out}) of capacity $c(x)$.
- Every arc of a capacity c going from a vertex u to a vertex in the cycle C_x is now an arc (u, x^{in}) of capacity c.

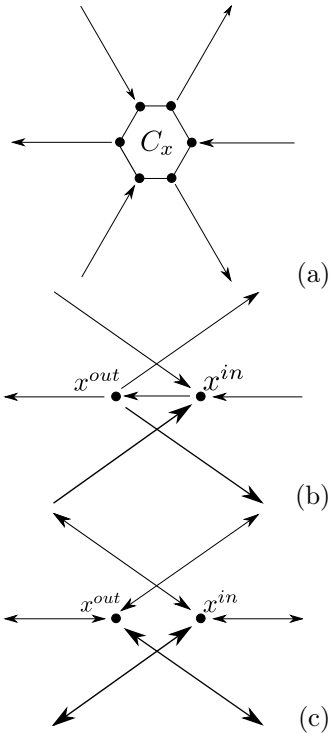

Figure 3: (a) C_x where $x \in X$ in G° (b) x^{in} and x^{out} in $G^\times$ (c) source x^{in} and sink x^{out} in H_i if $x = x_i$

- Every arc of a capacity c going from a vertex in the cycle C_x to a vertex x is now an arc (x^{out}, u) of capacity c.

In a slight abuse of terminology, we say that a flow in G° is an extension of a flow in $G^\times$ if the two flows have the same restriction to G. Similarly, a flow in $G^\times$ is a restriction of a flow in G° if the two flows have the same restriction to G. See Figure 3. To define $f^\times$, let $f^\times(u,v) = f^\circ(u,v)$ for all arcs $(u,v) \in E(G^\times) \cap E(G^\circ)$, and let $f^\times(x^{in}, x^{out}) = (f^\circ)^{in}(C_x)$ for all $x \in X$. It is easy to see that $f^\times$ is a flow from s to t whose only infeasible arcs are (x^{in}, x^{out}) for all $x \in X$. Furthermore, $(f^\times)^{out}(x^{out}) = (f^\times)^{in}(x^{in}) = f^{in}(x)$ for all $x \in X$. We have the following lemma:

LEMMA 5.2. *For each $x \in X$, let $\omega_x \geq 0$. The following two statements are equivalent:*

1. *There exists a feasible circulation g° in the residual graph of G° with respect to f° such that*

   $$(f^\circ + g^\circ)^{in}(C_x) = (f^\circ + g^\circ)^{out}(C_x) = f^{in}(x) - \omega_x$$

   for all $x \in X$.

2. *There exists a circulation $g^\times$ in $G^\times$ such that $f^\times + g^\times$ has a feasible extension in G°, $f^\times + g^\times$*

is feasible in $G^\times$ except at arcs (x^{in}, x^{out}) for all $x \in X$, and

$$(f^\times + g^\times)^{in}(x^{in}) = (f^\times + g^\times)(x^{in}, x^{out}) = (f^\times + g^\times)^{out}(x^{out}) = f^{in}(x) - \omega_x$$

for all $x \in X$.

Proof. (1) ⇒ (2) : Suppose (1) holds. Let $g^\times$ be the circulation in $G^\times$ defined by $g^\times(e) = g^\circ(e)$ for each $e \in E(G^\circ) \cap E(G^\times)$ and $g^\times(x^{in}, x^{out}) = (g^\circ)^{in}(C_x)$ for all $x \in X$. That is, $g^\times$ is the restriction of g° to $G^\times$. The circulation $g^\times$ satisfies conservation constraints at x^{out} because $(g^\times)^{in}(x^{out}) = g^\times(x^{in}, x^{out}) = (g^\circ)^{in}(C_x) = (g^\circ)^{out}(C_x) = (g^\times)^{out}(x^{out})$; a similar argument shows that $g^\times$ satisfies conservation constraints at x^{in}. Also, $g^\times$ satisfies conservation constraints at all other vertices because g° does.

Since $(f^\circ)^{in}(C_x) = (f^\times)^{in}(x^{in})$ and $(g^\circ)^{in}(C_x) = (g^\times)^{in}(x^{in})$, we have $(f^\circ+g^\circ)^{in}(C_x) = (f^\times+g^\times)^{in}(x^{in})$. A symmetric argument shows that $(f^\circ + g^\circ)^{out}(C_x) = (f^\times + g^\times)^{out}(x^{out})$. Flow conservation implies $(f^\times + g^\times)^{in}(x^{in}) = (f^\times + g^\times)(x^{in}, x^{out})$. The flow $f^\times + g^\times$ is feasible at all arcs in $E(G^\times) \cap E(G^\circ)$ because $f^\circ + g^\circ$ is.

(2) ⇒ (1) : Suppose (2) holds. There is a feasible extension h° of $f^\times + g^\times$ to G°. Let g° be the circulation in G° such that $f^\circ + g^\circ = h^\circ$. Since $f^\circ + g^\circ$ is feasible in G°, g° is feasible in the residual graph of G° with respect to f°. It is easy to see that g° is an extension of $g^\times$.

Since $(f^\circ)^{in}(C_x) = (f^\times)^{in}(x^{in})$ and $(g^\circ)^{in}(C_x) = (g^\times)^{in}(x^{in})$, we have $(f^\circ+g^\circ)^{in}(C_x) = (f^\times+g^\times)^{in}(x^{in})$. A symmetric argument shows that $(f^\circ + g^\circ)^{out}(C_x) = (f^\times + g^\times)^{out}(x^{out})$.

If $\lambda \le \lambda^*$, then there exists a feasible flow f_λ in G of value λ that can be extended to feasible flows $f_\lambda^\times$ in $G^\times$ and f_λ° in G°. Thus statement (1) of Lemma 5.2 holds for the circulation $f_\lambda^\circ - f^\circ$ in G° and for some choices of ω_x where $\omega_x \ge ex_x$ for all $x \in X$. Lemma 5.2 then implies that there exists a circulation $g^\times$ in $G^\times$ such that $(f^\times + g^\times)^{in}(x^{in}) = (f^\times + g^\times)(x^{in}, x^{out}) = (f^\times + g^\times)^{out}(x^{out}) \le c(x)$ for all $x \in X$, meaning that $f^\times + g^\times$ is feasible in $G^\times$. If $\lambda > \lambda^*$, then $g^\times$ may not exist and its computation may fail. Let g be the restriction of $g^\times$ to G.

We will compute the circulation $g^\times$ as the sum of $k-2$ circulations $\phi_1^\times, \ldots, \phi_{k-2}^\times$. Let $x_1, \ldots, x_{k-2}$ be an arbitrary ordering of the vertices in X. For all $i \in [k-2]$, let $\gamma_i^\times = \phi_1^\times + \cdots + \phi_i^\times$, and let γ_i be the restriction of $\gamma_i^\times$ to G. In particular, $\gamma_0^\times$ is the zero flow and $\gamma_{k-2}^\times = g^\circ$. We will find the circulations $\phi_1^\times, \ldots, \phi_{k-2}^\times$ one by one, and we will choose $\phi_i^\times$ to satisfy the following property:

LEMMA 5.3. *For all $i \in [k-2]$, $f^\times + \gamma_i^\times$ is a flow in $G^\times$ that is feasible except at some arcs (x_j^{in}, x_j^{out}) where $j > i$. Furthermore, the restriction of $f^\times + \gamma_i^\times$ to G has at most $ex(f)$ excess on $x_{i+1}, \ldots, x_{k-2}$ and at most $i \cdot ex(f)$ excess on vertices in $V(G) \setminus X$.*

Intuitively, the lemma states that $\phi_i^\times$ gets rid of the excess on x_i without increasing any of the excesses on $x_1, \ldots, x_{i-1}$ above 0, without increasing any of the excesses on $x_{i+1}, \ldots, x_{k-2}$ above $\mathrm{ex}(f)$, and without increasing any other excesses by more than $\mathrm{ex}(f)$.

We now describe how to find $\phi_i^\times$ inductively. Suppose $h^\times = f^\times + \gamma_{i-1}^\times$ is a feasible flow in $G^\times$ whose restriction h to G has no excess on $x_1, \ldots, x_{i-1}$, at most $\mathrm{ex}(f)$ excess on $x_i, \ldots, x_{k-2}$, and at most $(i-1) \cdot \mathrm{ex}(f)$ excess on all vertices in $V(G) \setminus X$. Our goal is to find a circulation $\phi_i^\times$ in $G^\times$ such that $h^\times + \phi_i^\times$ is a feasible flow in $G^\times$ satisfying Lemma 5.3. Finding $\phi_i^\times$ reduces to finding a flow $\phi_{i,H}$ in an $O(k)$-apex graph H_i, and we construct H_i as follows: Starting with the residual graph of $G^\times$ with respect to $h^\times$, delete arcs (x_i^{in}, x_i^{out}) and (x_i^{out}, x_i^{in}). Let the source be x_i^{in} and the sink be x_i^{out}. For all $j > i$, the arc (x_j^{in}, x_j^{out}) has capacity $c(x_j) + \mathrm{ex}(f) - h^\times(x_j^{in}, x_j^{out})$ and the arc (x_j^{out}, x_j^{in}) has capacity $h^\times(x_j^{in}, x_j^{out})$. See Figure 3. We have the following lemma:

LEMMA 5.4. *Let $\omega \ge 0$. For all $i \in [k-2]$, the following two statements are equivalent:*

1. *There exists a circulation $\phi_i^\times$ in $G^\times$ such that $h^\times + \phi_i^\times$ is feasible in $G^\times$ except possibly at arcs (x_j^{in}, x_j^{out}) for all $j > i$, where $(h^\times + \phi_i^\times)(x_j^{in}, x_j^{out}) \le c(x_j) + ex(f)$. Also,*

$$(h^\times + \phi_i^\times)^{in}(x_i^{in}) = (h^\times + \phi_i^\times)(x_i^{in}, x_i^{out}) = (h^\times + \phi_i^\times)^{out}(x_i^{out}) = h^{in}(x_i) - \omega.$$

2. *There exists a feasible flow $\phi_{i,H}$ in H_i of value ω.*

Proof. (1) ⇒ (2) : Suppose (1) holds. Let $\phi_{i,H}$ be the restriction of $\phi_i^\times$ to H_i. The flow $\phi_{i,H}$ is feasible in H_i by the definition of H_i.

Since $(h^\times)(x_i^{in}, x_i^{out}) = (h)^{in}(x_i)$ and $(h^\times + \phi_i^\times)(x_i^{in}, x_i^{out}) = h^{in}(x_i) - \omega$, we have $\phi_i^\times(x_i^{out}, x_i^{in}) = \omega$. Since x_i^{in} is not a source in $G^\times$, flow conservation at x_i^{in} implies $(\phi_i^\times)^{out}(x^{in}) = \omega$. This means that $|\phi_{i,H}| = \phi_{i,H}^{out}(x_i^{in}) = \omega$.

(2) ⇒ (1) : Suppose (2) holds. Define an extension $\phi_i^\times$ of $\phi_{i,H}$ to a circulation in $G^\times$ by setting $\phi_i^\times(x_i^{out}, x_i^{in}) = \omega$. It is easy to see that $g^\times$ satisfies conservation constraints. The arc capacities in H_i ensure $h^\times + \phi_i^\times$ is feasible in $G^\times$ except possibly at arcs

(x_j^{in}, x_j^{out}) with $j > i$, where $(h^\times + \phi_i^\times)(x_j^{in}, x_j^{out}) \leq c(x_j) + \mathrm{ex}(f)$.

Since $h^\times(x_i^{in}, x_i^{out}) = h^{in}(x_i)$ and $\phi_i^\times(x_i^{out}, x_i^{in}) = \omega$, we have $(h^\times + \phi_i^\times)(x^{in}, x^{out}) = h^{in}(x_i) - \omega$. On the other hand, x_i^{in} and x_i^{out} are not terminals in $G^\times$, so flow conservation implies $(h^\times + \phi_i^\times)^{in}(x_i^{in}) = (h^\times + \phi_i^\times)(x_i^{in}, x_i^{out}) = (h^\times + \phi_i^\times)^{in}(x_i^{out})$.

By the existence of $g^\times$, we know that there exists a circulation $\phi_i^\times$ such that $h^\times + \phi_i^\times$ is feasible in $G^\times$, so statement (1) in Lemma 5.4 holds for some $\omega \geq \mathrm{ex}(h, x_i)$. By Lemma 5.4, there must exist a flow $\phi_{i,H}$ of value $\mathrm{ex}(h, x_i)$ in H_i. We compute $\phi_{i,H}$ as follows: Starting with H_i, we add a vertex x^s that will be the source instead of x_i^{in}, and we add an arc (x^s, x_i^{in}) with capacity $\mathrm{ex}(h, x_i)$; similarly, we add a vertex x^t that will be the sink instead of x_i^{in}, and an arc (x_i^{out}, x^t) with capacity $\mathrm{ex}(h, x_i)$. The resulting graph has an acyclic maximum flow that saturates every arc incident to a terminal and so has value $\mathrm{ex}(h, x_i)$, and the restriction of this flow to H_i is $\phi_{i,H}$. We have assumed (by induction) that $\mathrm{ex}(h^\times, x_i) \leq \mathrm{ex}(f)$, so $|\phi_{i,H}| \leq \mathrm{ex}(f)$.

We need to show that our choice of $\phi_i^\times$ satisfies Lemma 5.3. By Lemma 5.4, the flow $\phi_{i,H}$ corresponds to a circulation $\phi_i^\times$ in $G^\times$ such that $h^\times + \phi_i^\times$ has no excess on $x_1, \ldots, x_i$ and is feasible in $G^\times$ except possibly at arcs (x_j^{in}, x_j^{out}) for all $j > i$, where $(h^\times + \phi_i^\times)(x_j^{in}, x_j^{out}) \leq c(x_j) + \mathrm{ex}(f)$. The restriction of $h^\times + \phi_i^\times$ to G is thus feasible at $x_1, \ldots, x_i$ and has at most $\mathrm{ex}(f)$ excess at $x_{i+1}, \ldots, x_{k-2}$. If $\lambda > \lambda^*$, then $\phi_{i,H}$ may not exist and might have value strictly less than $\mathrm{ex}(h, x_i)$ when we try to compute it. If this happens, then the restriction of $h^\times + \gamma_i^\times$ to G will have positive excess on x_i.

Let ϕ_i be the restriction of $\phi_i^\times$ to G. Let v be any vertex in $V(G) \setminus X$. Since $\phi_{i,H}$ is acyclic, ϕ_i sends at most $|\phi_{i,H}|$ units of flow through v. Thus for all $v \in V(G) \setminus X$, we have

$$\begin{aligned}\mathrm{ex}(h + \phi_i, v) &\leq \mathrm{ex}(h, v) + |\phi_{i,H}| \\ &\leq (i-1) \cdot \mathrm{ex}(f) + \mathrm{ex}(f) \\ &\leq i \cdot \mathrm{ex}(f).\end{aligned}$$

This proves Lemma 5.3.

When $i = k-2$, we get that $f^\times + \gamma_i^\times = f^\times + g^\times$ is a feasible flow in $G^\times$ where $\mathrm{ex}(f+g, v) \leq (k-2)\mathrm{ex}(f)$ for all $v \in V(G) \setminus X$. The flow $f^\times + g^\times$ has no excess on the vertices of X, so the proof of Lemma 2.1 implies that $f^\times + g^\times$ has an extension in G°. Lemma 5.2 then implies that $g^\times$ corresponds to a circulation g° in G° such that $\mathrm{ex}(f^\circ + g^\circ, x) = 0$ for all $x \in X$ and $\mathrm{ex}(f^\circ + g^\circ, v) \leq (k-2)\mathrm{ex}(f)$ for all $v \in V(G) \setminus X$; we can compute g° in $O(n \log^3 n)$ time. We have proved the following lemma:

LEMMA 5.5. *For any vertex $x \in X$, $ex(f^\circ + g^\circ, x) = 0$. For any vertex $v \in V(G) \setminus X$, $ex(f^\circ + g^\circ, v) \leq (k-2)ex(f)$.*

Computing $\phi_i^\times$ requires us to compute a maximum flow in a graph with $O(k)$ apices (these are s, t, and x^{in} and x^{out} for all $x \in X$), which takes $O(k^3 n \log^3 n)$ time using the algorithm of Borradaile et al. [1]. Since we need to compute $k-2$ such flows, computing g° takes $O(k^4 n \log^3 n)$ time.

5.2 Stage 2. Having found an integer circulation g° in G°, we construct the fractional circulation g°/k in G°. Then, using the algorithm of Lemma 2.3, we let f_1° be an integer flow in G° such that

$$\begin{aligned}&(f_1^\circ)^{in}(C_v) \leq \left\lceil (f^\circ + g^\circ/k)^{in}(C_v) \right\rceil \\ &\implies \mathrm{ex}(f_1^\circ, v) \leq \left\lceil \mathrm{ex}(f^\circ + g^\circ/k, v) \right\rceil.\end{aligned}$$

for every vertex $v \in V(G)$.

LEMMA 5.6. *For any vertex $v \in V(G)$, $ex(f_1^\circ, v) \leq \left\lceil \frac{k-1}{k} ex(f) \right\rceil$.*

Proof. If $x \in X$, then we have $\mathrm{ex}(f^\circ, x) \leq ex(f)$ and $\mathrm{ex}(f^\circ + g^\circ, x) = 0$ by Lemma 5.5. Thus

$$\mathrm{ex}(f_1^\circ, x) \leq \left\lceil \mathrm{ex}(f^\circ + g^\circ/k, x) \right\rceil \leq \left\lceil \frac{k-1}{k} ex(f) \right\rceil.$$

If $v \in V(G) \setminus X$, we have $\mathrm{ex}(f^\circ, v) = 0$ and $\mathrm{ex}(f^\circ + g^\circ, v) \leq (k-2)ex(f)$ by Lemma 5.5. Thus

$$\mathrm{ex}(f_1^\circ, v) \leq \left\lceil \mathrm{ex}(f^\circ + g^\circ/k, v) \right\rceil \leq \left\lceil \frac{k-2}{k} ex(f) \right\rceil.$$

Using the algorithm of Lemma 2.3, computing f_1° takes $O(n \log^3 n)$ time.

5.3 Stage 3. In this stage, we finally get f_2°. Using Lemma 2.4, we find a flow f_2° of the same value as f_1° such that the restriction f_2 of f_2° to G is acyclic. By Lemma 3.1, f_2° has at most $k-2$ infeasible vertices. Since $f_2^\circ(e) \leq f_1^\circ(e)$ for all arcs e, we still have $\mathrm{ex}(f_2^\circ, v) \leq \mathrm{ex}(f_1^\circ, v) \leq \left\lceil \frac{k-1}{k} \mathrm{ex}(f) \right\rceil$ for all vertices $v \in V(G)$. Stage 3 takes $O(n)$ time, so the total running time of stages 1-3 is $O(k^4 n \log^3 n)$.

6 The case $k = 3$

In the case of three terminals, we can find a maximum flow in $O(n \log n)$ time even if G has arbitrary real capacities. Without loss of generality, we may assume that there are two sources and one sink. Let f° be a maximum flow in G°. We can compute f° in $O(n \log n)$ time by using the algorithm of Borradaile and Klein [2]:

first find a maximum flow f_1° from s_1 to t, and then find a maximum flow f_2° from s_2 to t in the residual graph of G° with respect to f_1°. The desired flow f° is just $f_1^\circ + f_2^\circ$. By Lemma 2.4 we may assume without loss of generality that the restriction f of f° to G is acyclic. By Lemma 3.1, the flow graph f_G of f has at most one saddle x, and has index 1. If f is feasible at x, then f is the maximum flow in G and we are done, so assume f is infeasible at x.

6.1 Almost-feasible flows. Let $\delta = val(G^\circ) - val(G)$. Suppose f_δ° is a maximum flow in G° whose restriction f_δ to G is acyclic and has a single infeasible vertex x_δ. Lemmas 3.1 and 3.2 guarantee that x_δ has excess at most $c(x_\delta)$, but this excess might still be greater than δ. If it turns out that $\mathrm{ex}(f_\delta, x_\delta) = \delta$, then f_δ° and f_δ are *almost feasible*. Given an almost-feasible flow f_δ in G, we can remove δ units of flow through x_δ to get a maximum flow in G. This can be done in $O(n)$ time using an algorithm similar to that of Step 1 in Section 4. In this subsection, we show that almost-feasible flows exist.

LEMMA 6.1. *There exists a maximum flow f_δ° in G° such that the restriction f_δ of f_δ° to G is acyclic and has a single infeasible vertex x with $ex(f_\delta, x) = \delta$.*

Proof. Let f_{max} be a maximum flow in G, and let f_{max}° be an extension of f_{max} to G°. In the residual graph of G° with respect to f_{max}°, find an acyclic maximum flow g°. Let $(f')^\circ = f_{max}^\circ + g^\circ$ and let f' be the restriction of $(f')^\circ$ to G. Since g° is acyclic with a single sink and has value δ, it can be decomposed into arc-disjoint paths whose total flow value is δ. Therefore, for every vertex v in G, the flow through v in f' is larger than the flow through v in f_{max} by at most δ. Hence, the excess of every vertex under f' is at most δ. By Lemma 2.4, there is a flow f_δ° with the same value as $(f')^\circ$ whose restriction f_δ to G does not contain flow-cycles.

Since g° is a maximum flow in the residual graph of G° with respect to f°, $(f')^\circ$ and f_δ° are maximum flows in G°. Since $f_\delta(e) \le f'(e)$ for every arc e, we have $\mathrm{ex}(f_\delta) \le \delta$. Since f_δ is acyclic, Lemma 2.4 implies that f_δ has at most one infeasible vertex x.

If $\mathrm{ex}(f_\delta, x) < \delta$, then, starting with f_δ, we can remove $\mathrm{ex}(f_\delta, x)$ units of flow through x to get a feasible flow in G with value strictly higher than $|f_\delta| - \delta = val(G)$, a contradiction. Thus $\mathrm{ex}(f_\delta, x) = \delta$.

6.2 Getting an almost-feasible flow. It remains to show how to compute an almost-feasible flow. We will describe an algorithm that finds a circulation g° in G° such that the restriction of $f^\circ + g^\circ$ to G is almost feasible. Let $ex_x = \mathrm{ex}(f)$.

Construct the flow network $G^\times$ the same way as in Section 5.1, and construct H in the same way as H_i is constructed in section 5.1 for $i = 1$. In other words, H is constructed as follows. Starting with the residual graph of G° with respect to f°, we do the following:

- Replace C_x with vertices x^{in} and x^{out}.
- Every arc of capacity c going from a vertex u to a vertex in the cycle C_x is now an arc from (u, x^{in}) of capacity c
- Every arc of capacity c going from a vertex in the cycle C_x to a vertex u is now an arc (x^{out}, u) of capacity c.
- Let x^{in} be the source instead of s and x^{out} be the sink instead of t. (Recall from the definition of G_{st} in section 2 that s is a supersource connected to the original two sources s_1 and s_2.)

Lemmas 5.2 and 5.4 both apply for $H = H_1$. Thus our goal is now to find a maximum flow g_H in H that can be extended to a circulation in the residual graph of G° with respect to f°.

We will now show that we can make two simplifications to H. The goal of these simplifications is to eliminate the apices s, x^{in}, and x^{out} so that H becomes planar. First, since we are ultimately trying to find a flow in H from x^{in} to x^{out}, we may assume without loss of generality that arcs of the form (u, x^{in}) and (x^{out}, v) do not exist in H. As a result, the only arcs in H that are incident to x^{in} are arcs of the form (x^{in}, u) where $f(u, x) > 0$. If we consider these arcs as arcs in G, then, since x has index 1, these arcs form two intervals in the cyclic order around x. Therefore, we can replace x^{in} with two sources x_1^{in} and x_2^{in}, replacing arcs (x^{in}, u) in the first interval with (x_1^{in}, u) and arcs (x^{in}, u) in the second interval with arcs (x_2^{in}, u). A similar simplification eliminates x^{out}. See Figure 4. (We could not perform this simplification in Section 5.1 because the desired flow in H could send flow from x^{in} to some $(x')^{in}$ to x^{out}.) One effect of this simplification is that every flow g_H in H automatically extends to a circulation g° in the residual graph of G° with respect to f°. This is because f has an extension f° in G° and $(f + g_H)(e) \le f(e)$ for any arc e incident to x in G, so $f + g_H$ has an extension to G°.

Second, we show that we can delete the arcs (s, s_1) and (s, s_2). This eliminates the apex s.

LEMMA 6.2. *If there is an augmenting path in H (i.e., a path from a source to a sink in H) containing s, then there is an augmenting path in H not containing s.*

Proof. See Figure 5. In this proof, we say that an arc e carries flow if $f(e) > 0$, and a path carries flow if all

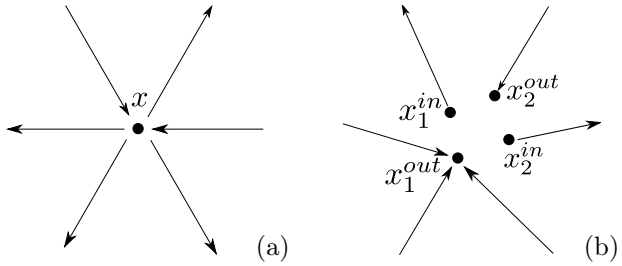

Figure 4: (a) The flow graph of f at the unique infeasible vertex (b) sources and sinks of H after eliminating apices

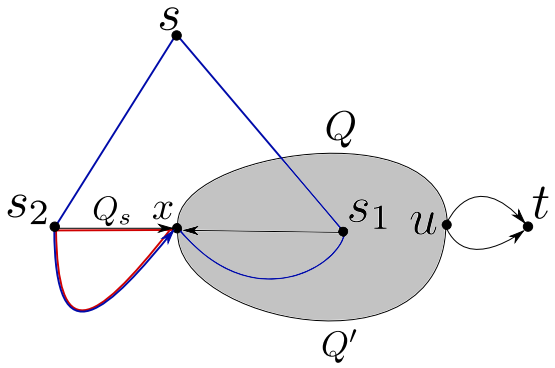

Figure 5: H in the proof of Lemma 6.2 with all terminals merged into the vertex x. The blue path is π. The red path is π'. The shaded region is R.

of its arcs carry flow. Consider two arcs e, e' carrying flow out of x such that as we cyclically traverse the arcs incident to x in clockwise order, some arc between e and e' carries flow into x, and some arc between e' and e carries flow into x. There must be a path P from x to t starting with e that carries flow. Similarly, there must be a path P' from x to t starting with e' that carries flow. Without loss of generality, assume P and P' do not cross. Let u be the first vertex on P after x that also appears on P'. The vertex u must also be the first vertex on P' after x that also appears on P, because otherwise f has flow-cycles. Let Q be the prefix of P that ends at the arc of P that goes into u, and let Q' be the prefix of P' that ends at the arc of P' that goes into u. These prefixes are well-defined because f is acyclic. Since both Q and Q' go from x to u, their union partitions the plane into two regions. Denote the inner region by R and the outer region by R'.

Since there are arcs in both R and R' carrying flow into x and f is acyclic, one source must be in R and the other must be in R'. Furthermore, there is some path Q_s from s_2 to x carrying flow, and there is some path from s_1 to x carrying flow.

Without loss of generality, suppose the augmenting path π in H starts in x^{in}, goes to $s_1 \in R$, uses arcs (s_1, s) and (s, s_2), and ends by going from $s_2 \in R'$ to x^{out}. We can replace it by an augmenting path π' that starts at x^{in}, follows $rev(Q_s)$ to s_2, and then follows π from s_2 to x^{out}. The augmenting path π' does not contain s.

Let g_H be the maximum flow in H and let g° be its extension to G°. We apply Lemma 2.4 to find a flow f_3° with the same value as $f^\circ + g^\circ$ whose restriction f_3 has no flow-cycles.

By Lemma 3.1, the flow f_3° is infeasible at a single vertex y. If $y = x$, then f_3° must be almost feasible. This is because Lemma 5.2 implies that if g_H is a maximum flow in H, then $f^\circ + g^\circ$ is a maximum flow in G° that minimizes the excess of x. Furthermore, $f_3(e) \leq (f^\circ + g^\circ)(e)$, so f_3 is a maximum flow in G° that minimizes the excess of x.

If $y \neq x$, then we define a function $F : E(G) \times [0, 1] \to \mathbb{R}$ for each arc $e \in G$. $F(e, \beta)$ is defined as follows. We apply Lemma 2.4 to $f^\circ + \beta g^\circ$ to get a flow f_β° whose restriction f_β to G is acyclic. We then define $F(e, \beta) = f_\beta(e)$. For all arcs $e \in E(G)$, we have $F(e, 0) = f(e)$ and $F(e, 1) = f_3(e)$.

Clearly, $F(\cdot, \beta)$ has an extension that is feasible in G° for all β, and $F(e, \cdot)$ is continuous for any arc $e \in E(G)$. Consider how $F(\cdot, \beta)$ changes as β increases from 0 to 1. We start with excess on x and no other vertices, and end with excess on y but no other vertices. Moreover, no matter what β is, there is at most one infeasible vertex. Thus, at some point, say when $\beta = \beta_0$, we must have no infeasible vertices. Since $F(\cdot, \beta_0)$ is a maximum flow in G°, it must be a maximum flow in G.

To compute β_0, we need the following lemma.

LEMMA 6.3. *For every fixed arc $e \in E(G)$, $\frac{\partial F(e,\beta)}{\partial \beta}$ is constant.*

Proof. The proof requires understanding the details of the algorithm of Lemma 2.4, which can be found in Appendix A. Here we summarize how the flow $F(\cdot, \beta)$ is computed:

1. Compute $f_\beta^\circ = f^\circ + \beta g^\circ$. Define a capacity function c' by $c'(e) = f_\beta^\circ(e)$ for all $e \in E(G)$ and $c'(e) = c(e)$ for all $e \notin E(G)$. Construct the residual graph G_β° of G° with respect to f_β° and c'. Let h_∞ be the infinite face of $G^\circ \setminus \{s\}$. For each face h of $G_\beta^\circ \setminus \{s\}$, let $\Phi(h)$ be the distance of h^* from h_∞^* in $(G_\beta^\circ \setminus \{s\})^*$. For each arc e in $G^\circ \setminus \{s\}$, let h_ℓ be the face on the left of e and let h_ℓ be the face on the right. Let $g_\beta(e) = \Phi(h_r) - \Phi(h_\ell)$ for each arc e in $G^\circ \setminus \{s\}$; g_β is a simple circulation. Finally, let $f_\gamma^\circ = f_\beta^\circ + g_\beta$, and let f_γ be the restriction of f_γ°

to G. The flow f_γ has no counter-clockwise flow-cycles.

2. Define a new capacity function $c''(e) = f^\circ_\gamma(e)$ for $e \in E(G)$ and $c''(e) = c(e)$ for $e \notin E(G)$. Construct the residual graph G°_γ of G° with respect to c'' and f°_γ. For each face h of $G^\circ_\gamma \setminus \{s\}$, let $\Phi(h)$ be he distance of h^* from h^*_∞ in $(G^\circ_\gamma \setminus \{s\})^*$. Let $g_\gamma(e) = \Phi(h_\ell) - \Phi(h_r)$. Finally, $F(\cdot, \beta)$ is the restriction of $f^\circ_\gamma + g_\gamma$ to G.

It suffices to show that the shortest path trees T_β in $(G^\circ_\beta \setminus \{s\})^*$ and T_γ in $(G^\circ_\gamma \setminus \{s\})^*$ rooted at h^*_∞ do not change as β increases. Suppose for the sake of argument that T_β changes as β increases. Then, there exist vertices u^* and v^* in $(G^\circ_\beta \setminus \{s\})^*$ and two internally disjoint paths P^*_1 and P^*_2 from u^* to v^* in $(G^\circ_\alpha \setminus \{s\})^*$ whose lengths are changing at different rates as β increases. Let H be the region bounded by P^*_1 and P^*_2, and suppose that $P_1 \circ rev(P_2)$ is a clockwise cycle. The change in the length of P^*_1 in $(G^\circ_\beta \setminus \{s\})^*$ is the change in the capacity of the cut P_1 in $G^\circ_\beta \setminus \{s\}$, which is the change in the amount of flow $f^\circ + \beta g^\circ$ sends out of H through the arcs of P_1. Similarly, the change in the length of P^*_2 is the change in the amount of flow $f^\circ + \beta g^\circ$ sends into H through the arcs of P_2. This means that the net amount of flow that $f^\circ + \beta g^\circ$ carries into H is changing as β increases, but this is impossible, since g° is a simple circulation. We conclude that T_β does not increase as β increases. A similar argument shows that since g_β is a simple circulation and $f^\circ_\gamma = f^\circ + \beta g^\circ + g_\beta$, T_γ does not change as β increases.

Lemma 6.3 implies that $\frac{d}{d\beta} ex(F(\cdot, \beta), x)$ is constant, and we can find it because

$$\begin{aligned}\frac{d}{d\beta} ex(F(\cdot, \beta), x) &= \operatorname{ex}(F(\cdot, 1), x) - \operatorname{ex}(F(\cdot, 0), x) \\ &= \operatorname{ex}(f_3, x) - \operatorname{ex}(f, x),\end{aligned}$$

We then let

$$\beta_0 = -\frac{\operatorname{ex}(F(\cdot, 0), x)}{\frac{d}{d\beta}\operatorname{ex}(F(\cdot, \beta), x)}.$$

and $F(\cdot, \beta_0)$ is a maximum flow in G.

The algorithm takes $O(n \log n)$ time to compute f°. It takes $O(n \log n)$ time to compute g_H, from which we can obtain g°, f°_3, and f_3 in linear time. If $y = x$, then we have an almost-feasible flow that can be turned into a maximum flow in G in linear time. If $y \neq x$, then we can compute β_0 and $F(\cdot, \beta_0)$ in linear time. The entire algorithm takes $O(n \log n)$ time.

6.3 Discussion. One natural question is what happens when $k = 4$. Here, we can define a maximum flow in G° as being almost feasible if we can remove δ units of flow to get a feasible flow in G. We can also prove that almost-feasible flows always exist. The main problem seems to be that there is no easy way of characterizing or getting almost-feasible flow. For example, minimizing the sum of the excesses of the two infeasible vertices x and x' does not necessarily work. Suppose there is one flow where the infeasible vertices both have excesses of 10, and another flow where the excesses are both 7. If $\delta = 10$, then it could be the case that the first flow is almost feasible because removing a unit of flow through x may simultaneously remove a unit of flow through x' (i.e., we can decompose the first flow into paths and cycles such that some paths pass through both x and x'), while the second flow is not almost feasible because removing a unit of flow through x does not simultaneously remove a unit of flow through x', and vice versa.

Acknowledgments. I would like to thank Jeff Erickson for helpful discussions and for comments on an earlier draft of this paper.

References

[1] Glencora Borradaile, Philip Klein, Shay Mozes, Yahav Nussbaum, and Christian Wulff-Nilsen. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. *SIAM Journal on Computing (SICOMP)*, 46(4):1280-1303, 2017.

[2] Glencora Borradaile and Philip N. Klein. An $O(n \log n)$ algorithm for maximum st-flow in a directed planar graph. *Journal of the ACM*, 56(2): 9:1-30, 2009.

[3] Éric Colin de Verdière and Alexander Schrijver. Shortest vertex-disjoint two-face paths in planar graphs. *ACM Trans. Algorithms*, 7(2):19:1-19:12, 2011.

[4] Jeff Erickson. Maximum flows and parametric shortest paths in planar graphs. In *Proc. 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 794-804, 2010.

[5] L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

[6] Andrew V. Goldberg and Satish Rao. Beyond the flow decomposition barrier. *Journal of the ACM*, 45:783-797, 1998.

[7] Stephen Guattery and Gary L. Miller. A contraction procedure for planar directed graphs. In *Proc. 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 431-441,1992.

[8] Monika R. Henzinger, Philip Klein, Satish Rao, and Sairam Subramanian. Faster shortest-path algorithms for planar graphs. *J. Comput. Syst. Sci.*, 55:3-23, 1997.

[9] Donggu Kang and James Payor. Flow Rounding. `arXiv:1507.08139`, July 2015.

[10] Haim Kaplan and Yahav Nussbaum. Maximum flow in directed planar graphs with vertex capacities. *Algorithmica*, 61(1):174-189, 2011.

[11] Samir Khuller, Joseph Naor, and Philip Klein. The lattice structure of flow in planar graphs. *SIAM J. Discrete Math*, 63:477-490, 1993.

[12] Samir Khuller and Joseph (Seffi) Naor. Flow in planar graphs with vertex capacities. Algorithmica, 11(3):200-225, 1994

[13] Yin Tat Lee, Satish Rao, and Nikhil Srivastava. A new approach to computing maximum flows using electrical flows. In *Proc. 45th Symposium on Theory of Computing - STOC '13*, pages 755-764, 2013.

[14] Gary L. Miller and Joseph Naor. Flow in planar graphs with multiple sources and sinks. *SIAM J. Comput.*, 24(5):1002-1017, 1995.

[15] Aleksander Mądry. Navigating central path with electrical flows: from flows to matchings, and back. In *Proc. 54th Annual IEEE Symposium on Foundations of Computer Science - FOCS '13*, pages 253-262, 2013.

[16] James B. Orlin. Max flows in $O(nm)$ time, or better. In *Proc. 45th Annual ACM Symposium on the Theory of Computing - STOC '13*, pages 765-774, 2013.

[17] Daniel D. Sleator and Robert E. Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.*, 26:362-391, 1983.

[18] Xianchao Zhang, Weifa Liang, and Guoliang Chen. Computing maximum flows in undirected planar networks with both edge and vertex capacities. In *Proc. 24th International Computing and Combinatorics Conference*, pages 577-586, 2008.

[19] Xianchao Zhang, Weifa Liang, and He Jiang. Flow equivalent trees in node-edge-capacitated undirected planar graphs. *Inf. Process. Lett.*, 100:100-115, 2006.

Appendices

A Proof sketch of Lemma 2.4.

The purpose of this section is to describe the algorithm of Lemma 2.4. This is needed for the proof of Lemma 6.3. We will not prove the correctness of the algorithm, as that has been done elsewhere [10] [11].

A.1 Duality. First, we need a few standard definitions. If G is a planar graph, the *dual graph* G^* of G has a vertex h^* for every face h of G, and an arc e^* for every arc e of G. The arc e^* is directed from the vertex of G^* corresponding to the face in G on the left side of e, to the vertex of G^* corresponding to the face in G on the right side of e. If e is undirected, then so is e^*. Any undirected edge $\{u, v\}$ can be represented by two directed arcs (u, v) and (v, u), each with the same weight as $\{u, v\}$. We put lengths $\ell(e^*)$ on the arcs e^* of G^* as follows: $\ell(e^*) = c(e)$ for every $e \in E(G)$.

A.2 Algorithm description. The algorithm has three steps and is based on an algorithm of Khuller, Naor, and Klein [11] that finds a circulation without clockwise residual cycles in a directed planar graph in $O(n)$ time.

Finding a circulation without clockwise residual cycles. We describe the algorithm of Khuller, Naor, and Klein that finds a circulation g in G° without clockwise residual cycles [11].

The graph $G^\circ \setminus \{s, t\}$ is planar. Let h_∞ be the infinite face of $G^\circ \setminus \{s, t\}$, and let h^*_∞ be its dual vertex. Using the algorithm of Henzinger et al. [8], compute the shortest path tree rooted at h^*_∞ in $(G^\circ \setminus \{s, t\})^*$ in $O(n)$ time. For every face h of G, let $\Phi(h)$ be the distance in $(G^\circ \setminus \{s, t\})^*$ from h^*_∞ to h^*. For any arc $e \in E(G^\circ \setminus \{s, t\})$, we define $g(e)$ as follows. Let h_ℓ be the face on the left of e and h_r be the face on the right of e. If $\Phi(h_r) \geq \Phi(h_\ell)$, then set $g(e) = \Phi(h_r) - \Phi(h_\ell)$. Otherwise, set $g(e) = 0$ (and $g(rev(e))$ will automatically be set to $\Phi(h_\ell) - \Phi(h_r)$). Khuller, Naor, and Klein [11] proved that the resulting flow function g is a simple circulation in G° such that G° has no clockwise residual cycles with respect to g.

Finding a flow without clockwise residual cycles. Let f° be a feasible flow in G°. We describe an algorithm due to Kaplan and Nussbaum [10] that computes a flow f°_1 in G° with the same value as f° and without clockwise residual cycles. A symmetric algorithm can then compute a flow in G° with the same value as f° and without counterclockwise residual cycles.

Let G°_f be the residual graph of G° with respect to f°. Using the algorithm of step 1, find a circulation g in G°_f such that G°_f does not have clockwise residual cycles with respect to g. Now define $f^\circ_1 = f^\circ + g$. Computing f°_1 takes $O(n)$ time. Kaplan and Nussbaum showed that f°_1 is a feasible flow in G° with the same value as f° and without clockwise residual cycles [10].

Finding an acyclic flow. Finally, let f° be a feasible flow in G°. We describe the algorithm due to Kaplan and Nussbaum [10] that computes a flow of the same value as f° whose restriction to G is acyclic. We will do this by first eliminating counterclockwise flow-cycles to get a flow f°_1; a symmetric algorithm then eliminates clockwise flow-cycles.

Define a new capacity function c_1 on the arcs of G° by first setting $c_1(e) = f^\circ(e)$ for $e \in E(G)$. This will ensure that we do not increase the flow along any arc of G. All other arcs in G° are in C_v for some vertex v; for these arcs e we set $c_1(e) = c(e) = c(v)/2$. Now we apply the previous algorithm to G° and c_1 to find a flow f°_1 with the same value as f° such that there are no clockwise residual cycles in G° with respect to

f_1° and c_1. Kaplan and Nussbaum [10] showed that the restriction of f_1° to G does not contain counterclockwise flow-cycles.

We now repeat the previous procedure symmetrically, by defining a new capacity c_2 that restricts the flow on every arc e of G to be at most $f_1^\circ(e)$, and finding a circulation in G° without counterclockwise residual cycles. This way we get from f_1° a flow f_2° of the same value whose restriction to G does not contain clockwise flow-cycles in G. For every $e \in E(G)$, we have $f_2^\circ(e) \leq f_1^\circ(e) \leq f^\circ(e)$, so we did not create any new flow-cycles when going from f° to f_1° to f_2°. Thus f_2° is a feasible flow in G° with the same value as f° whose restriction to G is feasible and acyclic.

B Proof of Lemma 3.1

In this section, we prove Lemma 3.1.

First we need a few definitions. For any face ϕ in f_G, let $\alpha(\phi)$ denote the alternation number of ϕ; $\alpha(\phi)$ is the number of times the arcs on the boundary of ϕ change direction as we traverse this boundary. Thus $\alpha(\phi) = 0$ if the arcs on the boundary of ϕ form a directed cycle. We use $index(\phi)$ to denote the index of a face ϕ, which is defined by $index(\phi) = \alpha(\phi)/2 - 1$.

Now we can proceed with the proof. See Figure 6. If at each vertex v in f_G we cycle through its incident arcs in order according to the embedding of f_G, each transition from one arc e to the next arc e' results in exactly one alternation either for v or for the face on whose boundary the two arcs e and e' lie. Thus

$$\begin{aligned}
2E &= \sum_{v \in V(f_G)} \alpha(v) + \sum_{\phi \in F(f_G)} \alpha(\phi) \\
\implies E &= \sum_{v \in V(f_G)} (index(v) + 1) \\
&\quad + \sum_{\phi \in F(f_G)} (index(\phi) + 1) \\
\implies E &= V + F + \sum_{v \in V(f_G)} index(v) \\
&\quad + \sum_{\phi \in F(f_G)} index(\phi) \\
\implies -2 &= \sum_{v \in V(f_G)} index(v) + \sum_{\phi \in F(f_G)} index(\phi)
\end{aligned}$$

where in the last line we have used Euler's formula $V(f_G) - E(f_G) + F(f_G) = 2$. Since f_G is acyclic, $index(\phi) \geq 0$ for each face ϕ, so $-2 \geq \sum_{v \in V(f_G)} index(v)$. Finally, $index(v) = -1$ for each terminal v, so

$$k_1 + k_2 - 2 \geq \sum_{v : index(v) \geq 1} index(v).$$

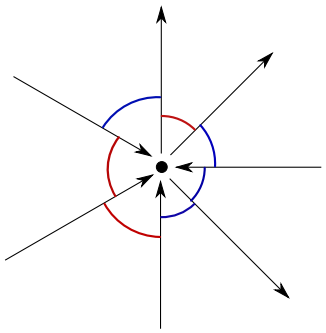

Figure 6: Proof of Lemma 3.1. Blue transitions contribute one alternation to a vertex; red transitions contribute one alternation to a face.

A vertex v is a saddle if and only if $index(v) \geq 1$, so this shows that the sum of the indices of the saddles in f_G is at most $k_1 + k_2 - 2$.