Maximum Flow in Directed Planar Graphs with Vertex Capacities*

Haim Kaplan and Yahav Nussbaum

The Blavatnik School of Computer Science,
Tel Aviv University, 69978 Tel Aviv, Israel
{haimk,yahav.nussbaum}@cs.tau.ac.il

Abstract. In this paper we present an $O(n \log n)$ algorithm for finding a maximum flow in a directed planar graph, where the vertices are subject to capacity constraints, in addition to the arcs. If the source and the sink are on the same face, then our algorithm can be implemented in $O(n)$ time.

For general (not planar) graphs, vertex capacities do not make the maximum flow problem more difficult, as there is a simple reduction that eliminates vertex capacities. However, this reduction does not preserve the planarity of the graph. The essence of our algorithm is a different reduction that does preserve the planarity, and can be implemented in linear time. For the special case of undirected planar graph, an algorithm with the same time complexity was recently claimed, but we show that it has a flaw.

1 Introduction

The problem of finding a maximum flow in a graph, or in a network, is a well-studied problem with applications in many fields, see the book of Ahuja, Magnanti and Orlin [1] for a survey. The maximum flow problem is also interesting if we restrict it to *planar* graphs, which are graphs that have an embedding in the plane without crossing edges. The case of planar graphs appears in many applications of the problem, for example road traffic or VLSI design. The special structure of planar graphs allows us to get simpler and more efficient algorithms for the maximum flow and related problems.

In the maximum flow problem, usually the arcs of the graph have capacities which limit the amount of flow that may go through each arc. We study a version of the problem in which the vertices of the graph also have capacities, which limit the amount of flow that may enter each vertex. This version appears for example when computing vertex disjoint paths in graphs, and in other problems where the vertices model objects which have a capacity.

Ford and Fulkerson [3, Chapter I.11] studied this version of the problem. They suggested the following simple reduction to eliminate vertex capacities.

* This research was partially supported by the United States - Israel Binational Science Foundation, project number 2006204 and by the Laura Schwarz-Kipp Institute of Computer Networks.

A. Fiat and P. Sanders (Eds.): ESA 2009, LNCS 5757, pp. 397–407, 2009.

We replace every vertex v with a finite capacity c by two vertices v' and v''. The arcs that were directed into v now enter v', and the arcs that were directed out of v now leave v''. We also add a new arc with capacity c from v' to v''. Unfortunately, this reduction does not preserve the planarity of the graph [7]. Consider for example the complete graph of four vertices, where one of the vertices has finite capacity. This graph is planar. If we apply the construction of Ford and Fulkerson we get a graph whose underlying undirected graph is the compete graph with 5 vertices. This graph is not planar by Kuratowski's Theorem.

The most efficient algorithm for maximum flow in directed planar graphs without vertex capacities, to date, was given by Borradaile and Klein [2] (Weihe [11] gave an algorithm with the same time bound but assuming a certain connectivity condition on the graph). Their paper also contains a survey of the history of the maximum flow problem on planar graphs. The time bound of the algorithm of [2] is $O(n \log n)$ where n is the number of vertices in the input graph. Borradaile and Klein ask whether their algorithm can be generalized to the case where the flow is subject to vertex capacities.

A planar graph is a *st-planar* graph if the source and the sink are on the same face. Hassin [4] gave an algorithm for the maximum flow problem in directed *st*-planar graphs without vertex capacities. The bottleneck of the algorithm is the computation of single-source shortest-path distances, which takes $O(n)$ time in a planar graph, using the algorithm of Henzinger et al. [5].

Khuller and Naor [7] were the first to study the problem of maximum flow with vertex capacities in planar graphs. They gave various results, including an $O(n\sqrt{\log n})$ time algorithm for finding the value of the maximum flow in *st*-planar graphs (which can be improved to $O(n)$ time using the algorithm of [5]), an $O(n \log n)$ time algorithm for finding the maximum flow in *st*-planar graphs, an $O(n \log n)$ time algorithm for finding the value of the maximum flow in undirected planar graphs, and an $O(n^{1.5} \log n)$ time algorithm for the same problem on directed planar graphs. If all vertices have unit capacities, then we get the *vertex-disjoint paths problem*. Ripphausen-Lipa et al. [10] solved this problem in $O(n)$ time for undirected planar graphs.

Recently, Zhang, Liang and Chen [13], used a construction similar to the one of [7] to obtain a maximum flow for undirected planar graphs with vertex capacities that runs in $O(n \log n)$ time. Their algorithm first constructs a planar graph without vertex capacities, and then uses the algorithm of [2] to find a maximum flow in it, which is modified in $O(n \log n)$ time to a flow in the original graph with vertex capacities. They also gave a different $O(n)$ time algorithm for finding a maximum flow in undirected *st*-planar graphs. Zhang et al. also ask in their paper if there is an algorithm that solves the problem for directed planar graphs.

In this paper we answer [2] and [13], and show a linear time reduction of the problem of finding maximum flow in directed planar graphs with arc and vertex capacities, to the problem of finding maximum flow in directed graphs with only arc capacities. This problem is more general than the one for undirected planar

graphs, since an undirected planar graph can be viewed as a special case of a directed planar graph, in which there are two opposite arcs between any pair of adjacent vertices.

We show how to apply the constructions of [7] and [13] to directed planar graphs. Given directed planar graph, we construct another directed planar graph without vertex capacities, such that we can transform a maximum flow in the new graph back to a maximum flow in the original graph. Since the new graph does not have vertex capacities we can find a maximum flow in it using the algorithm of [2] (or of [4] if it is an st-planar graph). The time bound of our reduction is linear in the size of the graph, therefore we show that vertex capacities do not increase the time complexity of the maximum flow problem also for planar graphs.

In addition, we show that the algorithm of [13] unfortunately has a flaw. We give an undirected graph in which this algorithm does not find a correct maximum flow. Therefore, in fact our algorithm is also the first to solve the problem for undirected graphs.

The outline of the paper is as follows: In the next section we give some background and terminology. In Sect. 3 we describe the construction of [7] that we use, and in Sect. 4 we describe the one of [13]. In Sect. 5 we characterize when a maximum flow in the constructed graph induces a maximum flow in the original graph and show how to efficiently find such a flow in the constructed graph. Finally, in the last section we combine all the pieces together to get our algorithm.

2 Preliminaries

We consider a simple directed planar graph $G = (V, E)$, where V is the set of vertices and E is the set of arcs, with a given planar embedding. The planar embedding of the graph G is represented combinatorially, see [9] for survey on planar graphs. An arc $e = (u, v) \in E$ is directed from $u \in V$ to $v \in V$. We denote the number of vertices by n, since the graph is planar we have $|E| = O(n)$.

A *path* $P = (e_0, e_1, \ldots, e_{k-1})$ is a sequence of arcs $e_i = (u_i, v_i)$ such that for $0 \leq i < k-1$ we have $v_i = u_{i+1}$. If in addition $v_{k-1} = u_0$ then P is a *cycle*. We say that a path P *contains* a vertex v, if either (u, v) or (v, u) is in P, for some vertex u. The path $P = (e_0, e_1, \ldots, e_{k-1})$ *starts* at u_0 and *ends* at v_{k-1}. For $v \in V$, $in(v) = \{(u, v) \mid (u, v) \in E\}$ is the set of incoming arcs and $out(v) = \{(v, u) \mid (v, u) \in E\}$ is the set of outgoing arcs.

The graph G has two distinguished vertices, $s \in V$ is the *source* and $t \in V$ is the *sink*. The source s has no incoming arcs, and the sink t has no outgoing arcs. Every arc $e \in E$, has a capacity $c(e) \geq 0$, and in addition every vertex $v \in V \setminus \{s, t\}$ has a capacity $c(v) \geq 0$. A capacity might be ∞. We assume that the source and the sink have no capacities, if we wish to allow them to have capacities, we can add a vertex s' that will be the source instead of s, and an arc (s', s) with the desired capacity, and similarly add a new sink t', and an arc (t, t') with the desired capacity. Note that this transformation keeps the graph planar, and even st-planar if it was so. It is easy to extend the given embedding

to accommodate s', t', and the arcs (s', s) and (t, t'). A graph without vertex capacities can be viewed as a special case in which $c(v) = \infty$ for every vertex.

A function $f : E \to R$ is a *flow function* if and only if it satisfies the following three constraints:

$$0 \leq f(e) \leq c(e) \quad \forall e \in E \ , \tag{1}$$

$$\sum_{e \in in(v)} f(e) \leq c(v) \quad \forall v \in V \setminus \{s, t\} \ , \tag{2}$$

$$\sum_{e \in in(v)} f(e) = \sum_{e \in out(v)} f(e) \quad \forall v \in V \setminus \{s, t\} \ . \tag{3}$$

Constraints (1) are the *arc capacity constraints*, Constraints (2) are the *vertex capacity constraints* and Constraints (3) are the *flow conservation constraints*.

We say that $e \in in(v)$ *carries flow into* v if $f(e) > 0$, and that $e' \in out(v)$ *carries flow out of* v if $f(e') > 0$.

The *value* of a flow f is $\sum_{e \in in(t)} f(e)$, the amount of flow which enters the sink. If the value of f is 0 then f is a *circulation*. Our goal, in the *maximum flow problem*, is to find a flow function of maximum value.

For a flow function f we define a cycle C to be a *flow-cycle* if $f(e) > 0$ for every arc in C. We extend this definition to every function $f : E \to R$, even if it is not a flow. If a function f has no flow-cycles we say that f is *acyclic*. An *acyclic flow* is a flow function which is acyclic.

Let $e = (u, v)$ we denote $rev(e) = (v, u)$. For a flow function f, we may assume that f does not contain an arc e such that both $f(e) > 0$ and $f(rev(e)) > 0$, because otherwise the flows in both directions can cancel each other. For a path $P = (e_0, e_1, \ldots, e_{k-1})$ we let $rev(P) = (rev(e_{k-1}), rev(e_{k-2}), \ldots, rev(e_0))$.

The planar embedding of G partitions the plane into connected regions called *faces*. For a simpler description of our algorithm, we fix an embedding of G such that t is on the boundary of the infinite face. It is easy to convert any given embedding to such an embedding [9].

The *dual graph* G^* of G has a vertex $D(h)$ for every face h of G, and an arc $D(e)$ for every arc e of G. The arc $D(e)$ connects the two vertices corresponding to the faces incident to e. The arc $D(e)$ is directed from the vertex that corresponds to the face on the left side of e to the one of the face on the right side of e. Intuitively, G^* is obtained from G by turning the arcs clockwise. The dual graph G^* is planar, but it is may have loops or parallel arcs. Every face h of G^* corresponds to a vertex v in G, such that the arcs that bound h are dual to the arcs that are incident to v. See Fig. 1. The capacity of $e \in E$, $c(e)$, is interpreted in G^* as the *length* of $D(e)$.

In the construction we present below we add undirected edges to directed graphs. Each such undirected edge uv can be represented by two antiparallel directed arcs (u, v) and (v, u), with the same capacity. If e is an undirected edge, then $D(e)$ is also undirected.

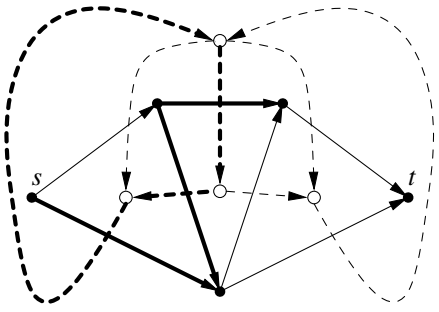

Fig. 1. A planar graph and its dual graph. The vertices of G are dots, and its arcs are solid. The vertices of G^* are circles, and its arcs are dashed. The bold arcs are an arc-cut in G and a cut-cycle in G^*. Capacities are not shown in this figure.

2.1 Residual Cycles

Let f be a flow in G. The *residual capacity* of an arc e with respect to f is defined as $c_r(e) = c(e) - f(e) + f(rev(e))$. In other words, the residual capacity of e is the amount of flow that we can add to e, or reduce from $rev(e)$. The *residual graph* of G with respect to f has the same vertex set and arc set as G, and the capacity for each edge e is $c_r(e)$. A *residual arc* with respect to f is an arc e with a positive residual capacity. A *residual path* is a path made of residual arcs. A *residual cycle* is a cycle made of residual arcs.

Khuller, Naor and Klein [8] presented an algorithm that finds a circulation such that there are no clockwise residual cycles with respect to this circulation, in a directed planar graph. The bottleneck of the algorithm of [8] is the computation of single-source shortest-path distances. Henzinger et al. [5] showed how to find these distances in a planar graph in $O(n)$ time, so the algorithm of [8] can be implemented in the same time bound. The complete details of the algorithm can be found also in [2]. We present an extension of this algorithm that changes a given flow into another flow, with the same value, without clockwise residual cycles with respect to it. We use this algorithm later to get the linear time bound for our reduction. The algorithm of [8] is for directed planar graphs without vertex capacities, so for the rest of this section assume that the graph G does not have vertex capacities.

In this section we also assume that if $e \in E$ then also $rev(e) \in E$. This assumption can be satisfied, without changing the problem, by adding the arc $rev(e)$ with capacity 0 for every arc e such that $rev(e)$ is not in E.

Given a flow f in G, we wish to find a flow f' with the same value, such that there are no clockwise residual cycles with respect to f'.

Let G' be the residual graph of G with respect to f. We find a circulation f_r in G', such that G' does not have clockwise residual cycles with respect to f_r, using the algorithm of [8]. Define f' to be the sum of f and f_r, that is $f'(e) = \max\{0, f(e) + f_r(e) - [f(rev(e)) + f_r(rev(e))]\}$. In other words, we add $f(e)$ and $f_r(e)$, and let the flows on e and $rev(e)$ cancel each other.

The function f' satisfies the two constraints of a flow without vertex capacities. The capacity of an arc e in G' is $c(e) - f(e) + f(rev(e))$ and therefore $f_r(e)$ is smaller than this capacity, therefore $f(e) + f_r(e) - [f(rev(e)) + f_r(rev(e))] \leq c(e)$, so $f'(e) \leq c(e)$ and the arc capacity constraints are satisfied in f'. The conservation constraints are satisfied, because these constraints are satisfied for f and for f_r, and f' is the sum of these two flows.

The value of the flow f' is the sum of the values of f and f_r. Since f_r is a circulation, its value is 0, and so the value of f' is the same as the value of f.

The flow f' has the desired property that G has no clockwise residual cycles with respect to f'. To show that, we show that if C is a clockwise residual cycle in G with respect to f', then C is also a residual cycle in G' with respect to f_r, contrary to the way we find f_r. Let e be an arc of C, and assume for contradiction that e is not residual in G' with respect to f_r. From our assumption $f_r(e) = c_r(e) = c(e) - f(e) + f(rev(e))$ and $f_r(rev(e)) = 0$. Therefore, $f'(e) = c(e)$ and e is not residual in G with respect to f', contradicting the fact that e is a member of C.

Lemma 1. *Let G be a directed planar graph without vertex capacities, and let f be a flow in G. We can find a flow f' in G, with the same value as f, such that there are no clockwise residual cycles with respect to f', in $O(n)$ time.*

3 Minimum Cut

In a graph without vertex capacities, a *cut* S is a minimal subset of E such that every path from s to t contains an arc in S. To avoid ambiguity later, when we introduce cuts that may contain vertices, we call such a cut an *arc-cut*. See Fig. 1. The *value* of an arc-cut S is $\sum_{e \in S} c(e)$. The *minimum cut problem* asks to find an arc-cut of minimum value. The fundamental connection between maximum flow and the minimum cut problems was given by Ford and Fulkerson [3] in the Max-Flow Min-Cut Theorem:

Theorem 2. *[3] The value of the maximum flow (in a graph without vertex capacities) is equal to the value of the minimum arc-cut in the same graph.*

Let C be a cycle in G^*. We say that C is a *cut-cycle* if it separates the faces corresponding to s and t, and goes counterclockwise around s (or equivalently, clockwise around t). See Fig. 1. The length of C is the sum of the lengths of its arcs. Johnson [6] showed the following relation between the value of minimum arc-cut and the value of shortest cut-cycle:

Lemma 3. *[6] Let G be a directed planar graph without vertex capacities. Then the value of the minimum arc-cut of G, is the same as the length of the shortest cut-cycle in G^*.*

Ford and Fulkerson [3, Chapter I.11] extended the definition of cuts to graphs with vertex capacities. In such a graph, a cut S is a minimal subset of $E \cup V$ such that every path from s to t contains an arc or a vertex in S. The value of a cut S is similarly defined as $\sum_{x \in S} c(x)$. Ford and Fulkerson also presented a

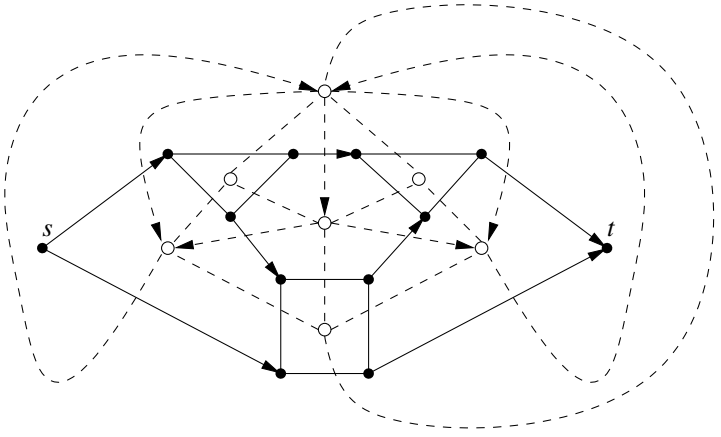

Fig. 2. Construction of G_C and G_E for the graph in Fig. 1. The graph G_C is presented as the dual graph of G_E. The newly added (undirected) edges are without arrowheads. Capacities are not shown in this figure.

version of the Max-Flow Min-Cut Theorem for graphs with vertex capacities, in this case the value of maximal flow (subject to both arc and vertex capacities) is equal to the value of the minimum cut (which contains both arcs and vertices).

Khuller and Naor [7] extended Lemma 3 using a supergraph G_C of G^* which they construct as follows. Let h be a face of G^* that corresponds to a vertex v of G with finite capacity. We add a new vertex v_h inside h and connect it by an (undirected) edge of length $c(v)/2$ to every vertex on the boundary of h. See Fig. 2.

Lemma 4. *[7] The values of the maximum flow and minimum cut in G are equal to the length of the shortest cut-cycle in G_C.*

4 The Extended Graph

Zhang, Liang and Jiang [12] and Zhang, Liang and Chen [13] construct the *extended graph* for an undirected graph with vertex capacities, based on the construction of Khuller and Naor [7]. We use the same construction for directed planar graphs with vertex capacities. The extended graph is defined as follows. We replace every vertex $v \in V$ which has a finite capacity with d vertices $v_0, \cdots, v_{d-1}$, where $d = |in(v)| + |out(v)|$ is the degree of v. We connect every v_i to $v_{(i+1) \bmod d}$ with an (undirected) edge of capacity of $c(v)/2$. We make every arc that was adjacent to v, adjacent to some vertex v_i instead, such that each arc is connected to a different vertex v_i, and the clockwise order of the arcs is preserved. We identify the new arc (u, v_i) or (v_i, u) with the original arc (u, v) or (v, u). We denote the resulting graph by G_E, and the cycle that replaces $v \in V$ in G_E by C_v. The graph G_E is a simple directed planar graph without vertex capacities. The arc set of G_E contains the arc set of G. See Fig. 2.

From the construction of G_E and G_C follows that G_C is the dual of G_E. Let v be a vertex with finite capacity and let h be the corresponding face in G^*. Then, in G_E we replaced v with C_v, and in G_C we placed v_h inside h. The edges which connects v_h to the boundary of h are dual to the edges of C_v.

From Theorem 2 we get that the value of the maximum flow in G_E is the same as the value of minimum arc-cut in G_E. By Lemma 3 this value is the same as the value of the shortest cut-cycle in G_C. Lemma 4 implies that this value equals to the value of the maximum flow in G, so the next lemma follows.

Lemma 5. *The value of the maximum flow of G is equal to the value of the maximum flow of G_E.*

5 Reduction from the Extended Graph to the Original Graph

We denote by f_E a flow function in G_E, and by f the *restriction* of f_E to the arcs of G, that is for every arc e of G, $f(e) = f_E(e)$. The value of f is the same as the value of f_E. The next lemma generalizes the result of Zhang et al. [13, Theorem 3], and its proof is similar.

Lemma 6. *Let f_E be a flow function in G_E. If f is acyclic then f is a flow function in G.*

In order to use Lemma 6 we must find a maximum flow f_E in G_E such that f is acyclic. In this section we show how to do that.

The algorithm of Zhang et al. [13, Section 3] for undirected planar graphs finds a flow f_E in G_E and than cancels flow-cycles in f in an arbitrary order. They call the resulting flow f_a and claim that this flow satisfies vertex capacities constraints. This approach is flawed. Fig. 3 shows an example on which the algorithm of [13] fails. After we cancel flow-cycles in f in an arbitrary order it is possible that there is no flow f'_E in G_E such that the restriction f' of f'_E to G is f_a, and therefore Lemma 6 does not apply.

As the example in Fig. 3 shows, it is not enough to cancel arbitrary flow-cycles in f. We can cancel a flow-cycle in f only if there is a cycle C in G_E that contains it, such that we can reduce flow along the cycle C. In this case the cycle $rev(C)$ in G_E is a residual cycle with respect to f_E. Therefore, in order to cancel a flow-cycle in G with respect to f we must cancel a residual cycle in G_E with respect to f_E. Canceling a arbitrary residual cycle is not enough, since we always want to reduce the flow that f assigns to arcs, and never increase it.

Let f_E be a flow in G_E. We define a new capacity function c' on the arcs of G_E which guarantees that the flow in an arc e of G never increases beyond the value of $f(e)$. For $e \in E$ we let $c'(e) = f(e)$. The arcs of G_E which are not in G are arcs of C_v for some vertex v, for these arcs we do not have to limit the flow to the amount in f_E, so we set $c'(e) = c(e) = c(v)/2$. The flow function f_E is also a flow function in G_E with the new capacity function c', by the way we defined c'. Since $c'(e) \leq c(e)$ for every arc e, every flow in G_E with the capacity function c' is also a flow in G_E with the original capacity function c.

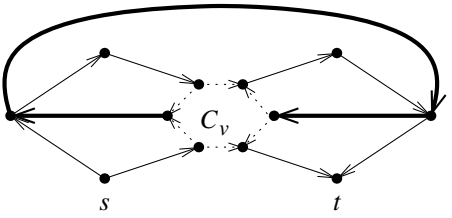

Fig. 3. A counterexample to the algorithm of [13]. The edges of the original undirected graph G are solid. The vertex v has capacity 1, the edges of C_v are dotted. The flow in every solid edge is 1, the flow in every dotted edges is 1/2, in the specified direction (the edges are undirected). The bold edges form a flow-cycle in G, after we cancel it we remain with an acyclic flow in G, but the amount of flow that enters v is 2. The correct solution is to cancel the flow in the two internal flow-cycles.

Instead of canceling the residual cycles one by one, we apply to G_E and c' the algorithm in Sect. 2.1 and find a new flow f'_E with the same value as f_E, such that there are no clockwise residual cycles in G_E with respect to f'_E and c'. The following lemma shows the crucial property of f'_E.

Lemma 7. *The restriction f' of f'_E to G does not contain counterclockwise flow-cycles.*

Proof. Assume, for a contradiction, that there is a counterclockwise flow-cycle C with respect to f' in G. We choose C such that C does not contain any other counterclockwise flow-cycle inside its embedding in the plane. We show that we can extend $rev(C)$ to a clockwise residual cycle with respect to f'_E in the graph G_E with capacity function c', in contradiction to the way we constructed f'_E.

For every arc e of C, $f'_E(e) > 0$, and therefore $rev(e)$ is a residual arc with respect to f'_E and c'. If C does not contain a vertex $v \in V$ with $c(v) \neq \infty$ then $rev(C)$ is a clockwise residual cycle with respect to f'_E and c', and we obtain a contradiction.

Let v be a vertex in C with $c(v) \neq \infty$. Let (u, v) and (v, u') be the arcs of C which are incident to v. These arcs correspond to arcs (u, v_i) and (v_j, u') in G_E, where v_i and v_j are in C_v. Let P be the path from v_j to v_i which goes counterclockwise around C_v (recall that C_v is undirected in G_E). To show a complete residual cycle in G_E, we argue that P is a residual path in G_E with respect to f'_E and c', so we can use it to fill the gap between v_j and v_i in $rev(C)$. An arc e of P is not residual if and only if $f'_E(e) = c'(e) = c(v)/2$. Without loss of generality we assume that the vertex v_k in the path P is followed by v_{k+1}.

Let $e_i = (v_{i-1}, v_i)$ be the last arc in the path P from v_j to v_i. Assume for contradiction that the arc e_i is not residual with respect to f'_E and c'. Then $f'_E(e_i) = c(v)/2$ and so the total flow into v_i in f'_E is $\sum_{e \in in(v_i)} f'_E(v_i) \geq f'_E((u, v_i)) + f'_E(e_i) > c(v)/2$. The only remaining arc that can carry flow out of v_i is $e_{i+1} = (v_i, v_{i+1})$. Because f'_E satisfies flow conservation constraints $f'_E(e_{i+1}) > c(v)/2$. But this is impossible since the capacity of the arc e_{i+1} is $c(v)/2$. Therefore, $f'_E(e_i) < c(v)/2$ and e_i is residual with respect to f'_E and c'.

We now proceed by induction. Assume by induction that we already know that the arc $e_{k+1} = (v_k, v_{k+1})$ on the path P from v_j to v_i is residual with

respect to f'_E and c'. If $k = j$ then we are done. Otherwise, we prove that $e_k = (v_{k-1}, v_k) \in P$ is also residual with respect to f'_E and c'. Since e_{k+1} is residual it follows that $f'_E(e_{k+1}) < c(v)/2$. Let e' be the single arc of E incident to v_k. If e' is directed out of v_k and $f'_E(e') > 0$ then since C is a cycle and e' is inside C in the embedding of G, there must be a path carrying flow that starts with e' and continues to another vertex on C. (Recall that t is on the boundary of the outer face.) This implies that there is a counterclockwise flow-cycle with respect to f' and G inside the embedding of C, in contradiction to the choice of C. Therefore e' does not carry flow of f'_E out of v_k in G_E. This implies, by the conservation constraint on v_k, that $f'_E(e_k) \leq f'_E(e_{k+1}) < c(v)/2$, so e_k is indeed residual with respect to f'_E and c'.

We showed that there is a residual path from v_j to v_i. Since v was an arbitrary vertex with $c(v) > 0$ on C it follows that we can extend $rev(C)$ to a residual cycle in G_E with respect to f'_E and c'. Since C is a counterclockwise cycle, the residual cycle we get from $rev(C)$ is a clockwise cycle. This contradicts the definition of f'_E, and therefore a counterclockwise flow-cycle C with respect to f' and G does not exist. □

We repeat the previous procedure symmetrically, by defining a new capacity c'' which restricts the flow in G to the flow in f', and applying a symmetric version of the algorithm of Sect. 2.1. This way we get from f'_E a flow f''_E of the same value, such that f'' does not contain clockwise flow-cycles in G. For every $e \in E$ we changed the flow such that $f''(e) \leq f'(e) \leq f(e)$, so we did not create any new flow-cycles. Therefore we have the following lemma.

Lemma 8. *The flow function f''_E has the same value as the flow function f_E. The restriction f'' of f''_E to G is acyclic.*

6 The Algorithm

Combining together the results of the previous sections we get an algorithm for finding maximum flow in a directed planar graph with vertex capacities.

First, we construct G_E from G by replacing each vertex that has a finite capacity with C_v as defined in Sect. 4. Next, we find a maximum flow f_E in G_E, which is a directed planar graph without vertex capacities. Last, we change f_E to another flow f''_E as in Sect. 5.

According to Lemma 8, the flow f''_E is a maximum flow in G_E, and its restriction f'' is acyclic. By Lemma 6, the function f'' is a flow in G. Since the value of f'' is the same as the value of f_E, Lemma 5 implies that f'' is a maximum flow.

The construction of G_E from G takes $O(n)$ time. The computation of f''_E from f_E also takes $O(n)$ time by Lemma 1. Therefore, the only bottleneck of our algorithm is finding f_E, a maximum flow in a directed planar graph without vertex capacities.

Theorem 9. *The maximum flow in a directed planar graph with both arc capacities and vertex capacities can be computed within the same time bound as the maximum flow in a directed planar graph with arc capacities only.*

The algorithm of Borradaile and Klein [2] finds a maximum flow in directed planar graph with arcs capacities in $O(n \log n)$ time. If G is a st-planar graph, then G_E preserves this property. In this case the algorithm of Hassin [4], using the algorithm of [5] for single-source shortest-path distances, finds a maximum flow in $O(n)$ time.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms and Applications. Prentice-Hall, New Jersey (1993)
2. Borradaile, G., Klein, P.: An $O(n \log n)$ algorithm for maximum st-flow in a directed planar graph. J. ACM 56 (2009)
3. Ford, L.R., Fulkerson, D.R.: Flows in Networks. Princeton University Press, New Jersey (1962)
4. Hassin, R.: Maximum flow in (s, t) planar networks. Information Processing Letters 13, 107 (1981)
5. Henzinger, M.R., Klein, P., Rao, S., Subramania, S.: Faster shortest-path algorithms for planar graphs. J. Comput. Syst. Sci. 55, 3–23 (1997)
6. Johnson, D.B.: Parallel algorithms for minimum cuts and maximum flows in planar networks. J. ACM 34, 950–967 (1987)
7. Khuller, S., Naor, J.: Flow in planar graphs with vertex capacities. Algorithmica 11, 200–225 (1994)
8. Khuller, S., Naor, J., Klein, P.: The lattice structure of flow in planar graphs. SIAM J. Disc. Math. 63, 477–490 (1993)
9. Nishizwki, T., Chiba, N.: Planar Graphs: Theory and Algorithms. Ann. Discrete Math, vol. 32. North-Holland, Amsterdam (1988)
10. Ripphausen-Lipa, H., Wagner, D., Weihe, K.: The vertex-disjoint Menger problem in planar graphs. SIAM J. Comput. 26, 331–349 (1997)
11. Weihe, K.: Maximum (s, t)-flows in planar networks in $O(|V| \log |V|)$-time. J. Comput. Syst. Sci. 55, 454–476 (1997)
12. Zhang, X., Liang, W., Jiang, H.: Flow equivalent trees in node-edge-capacitated undirected planar graphs. Information Processing Letters 100, 100–115 (2006)
13. Zhang, X., Liang, W., Chen, G.: Computing maximum flows in undirected planar networks with both edge and vertex capacities. In: Hu, X., Wang, J. (eds.) COCOON 2008. LNCS, vol. 5092, pp. 577–586. Springer, Heidelberg (2008)