

An $O(n \log n)$ algorithm for maximum st -flow in a directed planar graph

Glencora Borradaile^{*†} Philip Klein[†]

Abstract

We give the first correct $O(n \log n)$ algorithm for finding a maximum st -flow in a directed planar graph. After a preprocessing step that consists in finding single-source shortest-path distances in the dual, the algorithm consists of repeatedly saturating the leftmost residual s -to- t path.

1 Introduction

The study of maximum st -flow in planar graphs has a long history. In 1956, Ford and Fulkerson [6] described the *uppermost-path algorithm*, which solves the problem in the special case where the source s and the sink t are adjacent in the planar graph. (Such a graph is called an *st-planar* graph.) The algorithm repeatedly pushes flow along the uppermost residual s -to- t path. This algorithm has the property that no flow is ever removed from an arc. Since each augmentation makes at least one arc nonresidual, the algorithm requires at most m augmentations, where m is the number of arcs.

In 1979, Itai and Shiloach [13] showed that each iteration of the uppermost path algorithm could be implemented in $O(\log n)$ time, where n is the number of vertices. Consequently, the algorithm can be carried out in $O(n \log n)$ time (using the fact that a simple planar graph with n vertices has at most $3n$ arcs).

In 1991, Hassin demonstrated that a maximum st -flow in an st -planar graph G could be derived from shortest-path distances in the planar dual G^* of G where lengths in G^* correspond to capacities in G . With this insight, it can be seen that the uppermost-path algorithm can be interpreted in the planar dual as Dijkstra's algorithm. The fact that the uppermost path algorithm can be implemented to run in $O(n \log n)$ time corresponds to the observation, due to Johnson [14], that Dijkstra's algorithm could be implemented to run in $O(n \log n)$ time by using a priority queue.

Frederickson showed later that shortest-path distances in a planar graph with nonnegative lengths could be computed in $O(n\sqrt{\log n})$ time, and Henzinger et al. [12] showed subsequently that the same problem could be solved in $O(n)$

^{*}Partially supported by the Rosh foundation and a PGS-D fellowship from NSERC.

[†]Department of Computer Science, Brown University

time; combining this with Hassin’s result yields an $O(n)$ -time algorithm for maximum st -flow in st -planar graphs.

There remained, however, the more general and more natural problem of st -flow in a planar graph in which s and t need not be adjacent. In 1983, Reif [20] showed that the value of the maximum st -flow could be found in $O(n \log^2 n)$ time for the special case of *undirected* planar graphs. In 1985, Hassin and Johnson [10] draw on Reif’s technique to show that the flow assignment could also be found within the same time bound, again for *undirected* planar graphs. The result of Henzinger et al. can be used to reimplement these algorithms in $O(n \log n)$ time.

Still the general problem of st -flow in a planar directed graph remained open.¹ In 1982, Johnson and Venkatesan gave a divide-and-conquer algorithm that takes $O(n\sqrt{n} \log n)$ [15]. In 1989, Miller and Naor [19] showed that the problem could be reduced to computing shortest-path distances in a graph with positive and negatives lengths. In 1994, Henzinger et al. gave an algorithm for this problem that yielded a bound of $O(n^{4/3} \log n \log C)$ for max st -flow, where C is the sum of capacities.²

1.1 Weihe’s algorithm

In a significant step forward, Weihe [25, 26] published an $O(n \log n)$ algorithm for planar directed maximum st -flow. The proof of correctness is quite complicated and subtle. The algorithm, though clearly inspired by the uppermost-path algorithm, is also quite complicated. There is a preprocessing step in which the input graph is transformed into one of a special form, satisfying the following three requirements.

1. Each vertex but the source and sink has degree exactly three;
2. there are no clockwise cycles; and
3. for each arc uv , there is a simple s -to- v path and a simple u -to- t path, both using the arc uv .

Satisfying Requirement 1 involves: splicing together every two successive arcs sharing an endpoint of degree one; and replacing each vertex of high degree by a cycle, increasing the number of vertices to $2m$, which is at most $6n$. Requirement 2 can be satisfied in $O(m \log n)$ time by using a reduction of Khuller, Naor, and Klein [17] to computing shortest-path distances in the dual.³

Requirement 3 is problematic. As Weihe points out, any arc not satisfying this requirement is *useless* for st -flow, and can therefore be deleted. However,

¹The problem of maximum st -flow in an undirected graph can be reduced to the same problem in a directed graph.

²Subsequent to the work of Weihe, described below, Fakcharoenphol and Rao [5] presented a sub-quadratic algorithm for computing shortest-path distances in a graph with positive and negatives lengths. Combining this with Miller and Naor’s reduction implies an $O(n \log^4 n)$ algorithm for st -flow in a planar directed graph.

³The bound of $O(m \log n)$ comes from Dijkstra’s algorithm. The bound can be improved to $O(m)$ time using the algorithm of Henzinger et al. [12].

as pointed out by Biedl, Brejová, and Vinař [3], there is no known $O(m \log n)$ -time algorithm to delete all such arcs. They give an $O(m \log n)$ -time algorithm to find the set of arcs for which Requirement 3 fails, but, as they point out, deleting all such arcs does not achieve Requirement 3 since deleting one useless arc may make another arc useless. In fact, they show that $\Omega(n)$ phases of deletion can be necessary. It appears that the preprocessing step of Weihe’s algorithm therefore requires $\Omega(n^2 \log n)$ time using known methods. To our knowledge, the dependence of Weihe’s proof of correctness on Requirement 3 has not been resolved.⁴

1.2 The new result

In this paper, we give an $O(n \log n)$ -time algorithm for max st -flow in planar directed graphs. The only relevant requirement is Requirement 2. The max-flow algorithm itself is conceptually quite simple, and is arguably the right generalization of the uppermost-path algorithm. An abstract description of our algorithm (after preprocessing) is:

repeatedly augment the flow by saturating the leftmost residual s -to- t path until no such path remains.

The correctness of this algorithm follows directly from traditional maximum-flow theory (an st -flow is maximum if there remains no residual s -to- t path). We show that there are at most $3m$ iterations. We show, furthermore, that each iteration can be implemented in $O(\log n)$ time.

In Section 2, we give the notation and terminology used throughout the paper. In Section 3, we describe the algorithm in more detail and prove some invariants. In Section 4, we state a key theorem (Theorem 4.1) and show that it implies the bound on the number of iterations. In Section 5, we prove Theorem 4.1.

2 Notation and terminology

2.1 Graphs

We are concerned with directed graphs $G = \langle V, A \rangle$. For each arc $a \in A$, we define two oppositely directed *darts*, one in the same orientation as a (which we sometimes identify with a) and one in the opposite orientation. We define $rev(\cdot)$ to be the function that takes each dart to the corresponding dart in the opposite direction. Formally, the dart set is $A \times \{\pm 1\}$, and $rev(\langle a, i \rangle) = \langle a, -i \rangle$. The head and tail of a dart d in a graph G (written $head_G(d)$ and $tail_G(d)$) are such that the dart is oriented from tail to head. We may omit the subscript when doing so introduces no ambiguity.

A *path* of darts is a sequence $d_1 \dots d_k$ such that no dart appears twice and, for $i = 2, \dots, k$, $head_G(d_{i-1}) = tail_G(d_i)$. If additionally $head_G(d_k) = tail_G(d_1)$

⁴Weihe claims that his algorithm can be corrected; however, this claim has not been verified.

then the path is a cycle. A path/cycle of darts is *simple* if no vertex occurs twice as the head of a dart in the path/cycle.

If $P = d_1 \dots d_k$ and $Q = e_1 \dots e_\ell$ are paths such that $\text{end}(P) = \text{start}(Q)$, we use $P \circ Q$ to denote the path $d_1 \dots d_k e_1 \dots e_\ell$.

If $P = d_1 \dots d_k$ is a path, we use $\text{start}(P)$ to denote $\text{tail}(d_1)$ and we use $\text{end}(P)$ to denote $\text{head}(d_k)$. If x and y are vertices such that $x = \text{tail}(d_i)$, $y = \text{head}(d_j)$ and $i < j$ (or, for a trivial subpath, $x = y$), we use $P[x, y]$ to denote the subpath P' such that $\text{start}(P') = x$ and $\text{end}(P') = y$.⁵ If P is a cycle, and $i > j$, $P[x, y] = P[x, \cdot] \circ P[\cdot, y]$ denotes a subpath of the cycle. We use $P[x, y)$ to denote the path obtained from $P[x, y]$ by deleting the last dart; $P(x, y)$ and $P(x, y)$ are defined similarly. $P[\cdot, y]$ denotes the subpath P' with $\text{start}(P') = \text{start}(P)$ and $\text{end}(P') = y$. $P[x, \cdot]$ is similarly defined. The reverse of P , $\text{rev}(P)$ is defined as the sequence $\text{rev}(d_k), \text{rev}(d_{k-1}), \dots, \text{rev}(d_1)$.

A (*root-*)*directed spanning tree* of G is a set T of darts such that (i) every vertex but one (the *root*) is the head of exactly one dart, and (ii) there are no cycles. For a vertex v , $T[v]$ denotes the unique path of darts in T whose start vertex is v and whose end vertex is the root. For vertices u and v , $T[u, v]$ denotes the unique simple u -to- v path through T (possibly using the reverses of arcs of T).

For a set S of vertices, $\Gamma^+(S)$ is the set of darts whose tails are in S and whose heads are not. Such a set of darts is called a *cut*. The cut is *simple* if S is connected and the complement of S connected.⁶

2.2 Planar Graphs

According to the geometric definition,⁷ a planar embedding of a graph is a drawing of the graph on the plane or on the surface of a sphere so that vertices are mapped to distinct points and arcs are mapped to nonintersecting sets of points. A planar graph is a graph for which there exists a planar embedding. The set of points in the plane/sphere that are not in the image of the vertices or arcs decomposes into connected components, called *faces*. That is, faces are the maximal regions bounded by the embeddings of the vertices and edges.

For an embedding on the plane, there is one *infinite face*. For an embedding on the sphere, an arbitrary face can be designated as the infinite face.

Corresponding to every connected planar embedded graph G there is another connected planar embedded graph denoted G^* . The faces of G are the vertices of G^* , and the arcs (and hence darts) of G correspond one-to-one with those of G^* . If d is a dart of G , the tail of the corresponding dart of G^* is the face to the left of d , and the head is the face to the right of d . Thus intuitively the geometric orientation in G^* of the dart corresponding to d is obtained by

⁵Since nonsimple paths visit vertices multiple times, when we use this notation with nonsimple paths, we intend x and y to refer to specific occurrences of vertices within the path.

⁶A simple cut is known as a *bond*.

⁷One can alternatively define embeddings and planar graphs combinatorially, without reference to topology. Such a formulation makes proofs and algorithms somewhat easier to formulate, but for this paper we use the more familiar geometric definition.

rotating the embedding of d clockwise roughly 90 degrees. It is notationally convenient to equate the darts of G with the darts of G^* .

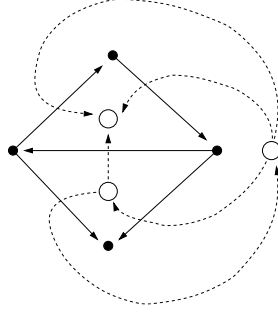


Figure 1: A planar graph and its dual: the primal is given by solid vertices and solid arcs and the dual is given by open vertices and dotted arcs.

Primal and dual spanning trees A classical result on planar graphs is as follows: for an undirected spanning tree T of G , the set of arcs *not* in T form an undirected spanning tree of the dual G^* [4, 23].

Simple cycles and cuts Another classical result on planar graphs is that a set of darts forms a simple directed cycle in the primal iff it forms a simple directed cut in the dual [27].

2.3 Vector spaces

The *arc space* of a graph $G = \langle V, A \rangle$ is the vector space \mathbf{R}^A . That is, a vector δ in arc space assigns a real number $\delta[a]$ to each arc $a \in A$. It is notationally convenient to interpret a vector δ in arc space as assigning real numbers to all darts. For any dart $\langle a, i \rangle$ ($i = \pm 1$), we define

$$\delta[\langle a, i \rangle] = i \cdot \delta[a]$$

For each arc a , we define $\delta(a)$ to be the vector in arc space that assigns 1 to a and zero to all other arcs. That is, for any arc a' ,

$$\delta(a)[a'] = \begin{cases} 1 & \text{if } a' = a \\ 0 & \text{otherwise} \end{cases}$$

For a multiset S of darts, we define $\delta(S) = \sum_{d \in S} \delta(d)$. For a graph structure H (e.g. path or cycle), we define $\delta(H) = \delta(S)$ where S is the multiset of darts comprising H .

We equate a vertex v with the set of darts whose tails are v , so $\delta(v)$ is $\sum_{\text{tail}(d)=v} \delta(d)$. Similarly, we equate a face f with the set of darts forming the

counterclockwise boundary of the face, so $\delta(f)$ is the sum of $\delta(d)$ over such darts.

A vector η in arc space specifies a set of darts of G , namely the set of darts assigned positive values by η . We say, for example, that a dart d is *in* η if $\eta[d] > 0$. We can similarly say that η contains a path or a cycle.

2.4 Flow

Given a graph G and vertices s and t , an *st-flow* is a vector in arc-space \mathbf{f} that satisfies $\mathbf{f} \cdot \delta(v) = 0$ for every vertex v except s and t .⁸ in arc space such The *value* of the *st-flow* \mathbf{f} is $\mathbf{f} \cdot \delta(s)$. A *capacity function* for a graph G is a function $c(\cdot)$ mapping the set of darts (not the set of arcs⁹) of G to the real numbers. We say that an *st-flow* \mathbf{f} is *feasible with respect to* c if $\mathbf{f}[d] \leq c(d)$ for every dart d . A *maximum st-flow* for a graph G and capacity function c is a feasible *st-flow* of maximum value.

A dart d is *residual* with respect to \mathbf{f} and c if $\mathbf{f}[d] < c(d)$. Otherwise, d is nonresidual. A path/cycle is residual if all its darts are residual. It is well-known that an *st-flow* \mathbf{f} that is feasible with respect to c is maximum if there is no residual *s-to-t* path with respect to \mathbf{f} and c .

Augmenting an *st-flow* \mathbf{f} along a residual *s-to-t* path P means increasing $\mathbf{f}[d]$ by the same amount for each dart d in P . Suppose that f is feasible with respect to c . If the amount of the increase is no more than

$$\min_{d \in P} c(d) - \mathbf{f}[d]$$

then after augmentation the *st-flow* \mathbf{f} is still feasible. If the increase is exactly this amount then we say the augmentation *saturates* the path P . In this case, at least one dart of P becomes nonresidual.

2.5 Circulations

The *cycle space* of G is the subspace of the arc space spanned by

$$\{\delta(C) : C \text{ a cycle of darts in } G\}.$$

We refer to a vector in the cycle space of G as a *circulation*. *Warning:* Except in a preprocessing step in the algorithm, circulations have nothing to do with flow in this paper.

In a planar graph, for any face f_0 the set of vectors

$$\{\delta(f) : f \text{ a face of } G, f \neq f_0\}$$

⁸As a consequence of our convention expressing the value of an arc-space vector at darts (not just arcs), we have $\mathbf{f}[d] = -\mathbf{f}[\text{rev}(d)]$, which coincides with the convention of *antisymmetry* introduced by Goldberg [8].

⁹If a capacity function c is given only in terms of the arcs, then we define a capacity function c' on the darts as $c'((a, 1)) = c(a)$ and $c'((a, -1)) = 0$ for each arc a .

is a basis for the cycle space of G . (We take f_0 to be the infinite face f_∞ .) Therefore any circulation $\boldsymbol{\eta}$ can be represented as a linear combination of these basis vectors. We use $\boldsymbol{\phi}$ to denote the vector of coefficients for this linear combination, so

$$\boldsymbol{\eta} = \sum_{f \neq f_\infty} \phi[f] \boldsymbol{\delta}(f)$$

We call $\boldsymbol{\phi}$ a *potential assignment*, and we refer to $\phi[f]$ as the *potential* of face f .¹⁰ We adopt the convention that $\phi[f_\infty]$ is defined to be zero.

External and internal We say a face f is *external* to the circulation corresponding to a potential assignment $\boldsymbol{\phi}$ if $\phi[f] = 0$, and is *internal* otherwise. We say that a dart d is external if the faces to d 's left and right are external, and we say d is internal if the faces to d 's left and right are internal. A dart can be neither internal nor external. In this case, the dart is contained by the circulation. A dart and its reverse have the same property (external, internal, or contained) with respect to a circulation. We say a vertex v is external if every dart incident to v is external, and is internal if every dart incident to v is internal.

Encloses For a cycle C , we say C *encloses* a face (dart or vertex, resp.) if the face (dart or vertex, resp.) is internal to $\boldsymbol{\delta}(C)$.

Counterclockwise and clockwise A circulation is *counterclockwise* (abbreviated *c.c.w.*) if the potential of every face is nonnegative [17]. A circulation is *clockwise* (abbreviated *c.w.*) if the potential of every face is nonpositive. A cycle P is clockwise if $\boldsymbol{\delta}(P)$ is clockwise.

2.6 The Left/Right Relation

Based on [17], Weihe defined the left-right relation for flows [26] and Klein specialized this to paths [18]: An x -to- y path A is *left of* an x -to- y path B if $\boldsymbol{\delta}(A) - \boldsymbol{\delta}(B)$ is a clockwise circulation. Likewise A is *right of* B if $\boldsymbol{\delta}(A) - \boldsymbol{\delta}(B)$ is a counterclockwise circulation. *Left of* and *right of* are transitive, reflexive, antisymmetric relations. An x -to- y path A is the leftmost x -to- y path in a graph if, for every x -to- y path B , A is left of B . If A is a leftmost path then every subpath of A is a leftmost path.

For two st -flows of equal value \mathbf{f}_1 and \mathbf{f}_2 , \mathbf{f}_1 is left (resp. right) of \mathbf{f}_2 if $\mathbf{f}_1 - \mathbf{f}_2$ is a clockwise (resp. c.c.w.) circulation. A flow \mathbf{f} is a leftmost flow of its value if for every other flow \mathbf{f}' of the same value, \mathbf{f} is left of \mathbf{f}' .

¹⁰This use of potentials was introduced by Hassin [9] for st -planar graphs and by Miller and Naor [19] for general planar graphs.

Crossings The definition of “crosses” and “enters/leaves on the right/left” are illustrated in Figure 2a.

If A and B do not cross for any vertex, then they are noncrossing. A path is *non-self-crossing* if for every pair of subpaths P and Q of the path, P does not cross Q . See Figure 2 for examples of these situations.

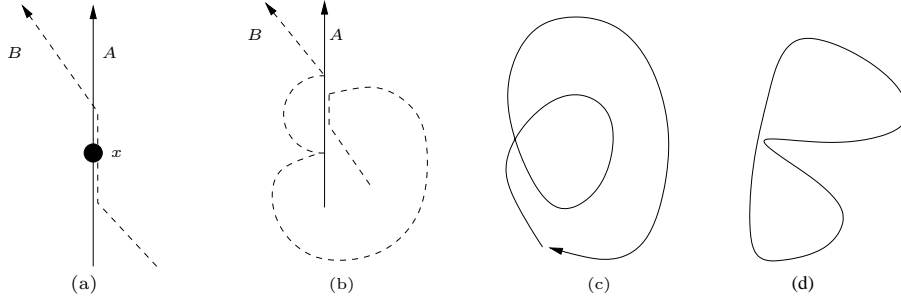


Figure 2: (a) A crosses B : $B[\cdot, x]$ enters A on the right and $B[x, \cdot]$ leaves A on the left. (b) A and B are noncrossing. (c) This is a self-crossing cycle. (d) This is a non-self-crossing but nonsimple cycle.

Lemma 2.1 (Composition Lemma). *Let C be a non-self-crossing cycle and let A be a non-self-crossing path with endpoints on C and in no part enclosed by C . Then $A \circ C[\text{end}(A), \text{start}(A)]$ is a non-self-crossing cycle.*

Proof. Since A is in no part enclosed by C , A does not cross C . It follows that $A \circ C[\text{end}(A), \text{start}(A)]$ is non-self-crossing. \square

Theorem 2.2 (Noncrossing Theorem). *If P and Q are non-self-crossing x -to- y paths that do not cross each other, P is either right of or left of Q .*

Proof. Let $C = Q \circ \text{rev}(P)$: C is a non-self-crossing cycle. Let G_C be the graph induced on C . Since G_C is a connected graph, each face of G_C has a connected boundary. Recall that we equate a face with the set of darts forming the c.c.w. boundary of the face.

First we show that each face of G_C uses either darts of C or darts of $\text{rev}(C)$ (but not both). Suppose for a contradiction that f is a face that is bounded by subpaths A and $\text{rev}(B)$ of C such that $\text{end}(A) = \text{end}(B)$: let x be this common vertex. We can write C as $A \circ A' \circ B \circ B'$. Since f is a face, C does not cross the boundary of f . The boundary of f is oriented c.c.w. and $A \circ A'$ leaves the boundary of f so it must leave the boundary of f from the right. Likewise $B \circ B'$ leaves the boundary of f from the right. Both $A \circ A'$ and $B \circ B'$ leave f from the right at x . This implies that $A \circ A'$ crosses $B \circ B'$ at x . This contradicts that C is a non-self-crossing cycle.

Consider the following face assignment for the faces of G_C :

$$\phi[f] = \begin{cases} 1 & \text{if } d \text{ is on the boundary of } f \text{ and } \text{rev}(d) \text{ is not} \\ 0 & \text{otherwise} \end{cases}$$

Since each face of G_C uses either darts of C or darts of $rev(C)$, the above assignment is consistent. Consider the circulation $\boldsymbol{\eta} = \boldsymbol{\delta}C$. If a dart d is contained by $\boldsymbol{\eta}$ then the face to the left of d is assigned 1 and the face to the right is assigned 0 and $\boldsymbol{\eta}[d] = 1$ as expected.

If $\phi[f_\infty] = 0$, then ϕ is a valid potential assignment and $\boldsymbol{\eta}$ is c.c.w. so Q is right of P . If $\phi[f_\infty] = 1$ then $\phi - \mathbf{1}$ (where $\mathbf{1}$ is the all-ones vector) is a valid potential assignment. Then $\boldsymbol{\eta}$ is c.w. so Q is left of P . \square

3 Algorithm

Our algorithm for finding the maximum st -flow in a planar embedded graph G_{input} with positive capacities c_{input} is as follows:

Designate some face f_∞ with t on its boundary as the infinite face.
 $(G_0, c_0) = \text{CLOCKWISE-CYCLE-SATURATING-0-FLOW}(G_{\text{input}}, c_{\text{input}}, f_\infty)$.
 Return $\text{MAX-FLOW}(G_0, c_0, s, t)$.

The second step finds a 0-value flow¹¹ in G_{input} with respect to which the residual graph has no clockwise cycles. This can be found as follows (expanded from [17]):

$\text{CLOCKWISE-CYCLE-SATURATING-0-FLOW}(G_{\text{input}}, c_{\text{input}}, f_\infty)$
 interpret capacities $c_{\text{input}}(d)$ as lengths of darts in the dual graph G_{input}^*
 let $\phi[f] = \text{dist}_{G_{\text{input}}^*}(f_\infty, f)$ for every face f
 let $\boldsymbol{\eta}[d] = \phi[\text{head}_{G_{\text{input}}^*}(d)] - \phi[\text{tail}_{G_{\text{input}}^*}(d)]$ for every dart d
 let G_0 be the residual graph with respect to $\boldsymbol{\eta}$
 $c_0(d) = c_{\text{input}}(d) - \boldsymbol{\eta}[d]$ for every dart d
 return (G_0, c_0)

We chose the arc set of G_0 to be such that every arc has nonzero capacity: for each arc a of G_{input} , if $\boldsymbol{\eta}[a] = c(a)$ then interpret $rev(a)$ as an arc of G_0 , else interpret a as an arc. Also note that since $c_{\text{input}}(d) \geq 0$ for each dart d , $c_0(d) \geq 0$ for each dart d . In what follows, *arc* always refers to an arc of G_0 , *anti-arc* refers to a dart whose reverse is an arc.

Lemma 3.1. *Every clockwise cycle of darts in G_0 is nonresidual.*

Proof. Let C be a clockwise cycle of darts in G_0 and let T be the tree representing the shortest path distances computed in $\text{CLOCKWISE-CYCLE-SATURATING-0-FLOW}$. There is a path in T from f_∞ to every face enclosed by C , so at least one dart of C is in the shortest path tree. Let d be such a dart:

$$\begin{aligned} c(d) &= c_{\text{input}}(d) - \boldsymbol{\eta}[d] \\ &= c_{\text{input}}(d) - \left(\text{dist}_{G_{\text{input}}^*}(f_\infty, \text{tail}_{G_{\text{input}}^*}(d)) - \text{dist}_{G_{\text{input}}^*}(f_\infty, \text{head}_{G_{\text{input}}^*}(d)) \right) \\ &= c_{\text{input}}(d) - c_{\text{input}}(d) \quad \text{since } d \text{ is in the shortest path tree} \\ &= 0 \end{aligned} \quad \square$$

¹¹equivalently, a circulation

The procedure MAX-FLOW takes a planar graph G_0 with no clockwise cycle of arcs such that t is on the boundary of the infinite face. We start with an abstract description of MAX-FLOW:

Abstract version of MAX-FLOW(G_0, c, s, t):
 initialize \mathbf{f} to be the st -flow of value zero.
 while there is a residual s -to- t path w.r.t. \mathbf{f} ,
 saturate the leftmost such path, modifying \mathbf{f} .
 return \mathbf{f} .

The correctness of the algorithm follows from a well-known max-flow result, that an st -flow \mathbf{f} is maximum iff there is no s -to- t path that is residual with respect to \mathbf{f} . For the sake of bounding the running time, we next give a implementation of MAX-FLOW(G_0, c, s, t) as a kind of network-simplex algorithm. Later in this section we show that the implementation indeed implements the abstract version. The algorithm maintains a directed spanning tree T of the graph (rooted at the sink t) and the corresponding dual spanning tree T^* (rooted at the infinite face f_∞) which will also turn out to be directed.

Implementation of MAX-FLOW(G_0, c, s, t):
 1 initialize \mathbf{f} to be the st -flow of value zero.
 2 initialize T to be the right-first search tree searching backwards from t .
 3 let G be the graph obtained from G_0 by deleting all vertices not in T .
 4 initialize T^* to consist of the darts of G whose arcs are not in T .
 5 repeat
 6 if $T[s]$ is residual, saturate $T[s]$, modifying \mathbf{f} .
 7 let d be the rootmost nonresidual dart in $T[s]$.
 8 if $tail_{G^*}(d)$ is a descendent in T^* of $head_{G^*}(d)$ then return \mathbf{f} .
 9 let e be the parent dart in T^* of $head_{G^*}(d)$.
 10 eject e from T^* and insert $rev(d)$ into T^* .
 11 eject d from T and insert e into T .
 12 replace the darts in $T[tail_G(e), tail_G(d)]$ with their reverses.

Right-first search [21] in Step 2 constructs a tree T spanning every vertex v that can reach t in G_0 , and the path $T[v]$ is the leftmost v -to- t path in G_0 . The primal tree T is represented using a *dynamic tree* data structure [1, 2, 7, 22, 24], enabling Steps 6, 7, 11 and 12 to be implemented to run in $O(\log n)$ time. The dual tree T^* is represented by an Euler-tour tree data structure [11], so Steps 8, 9 and 10 can also be implemented in $O(\log n)$ time.

We refer to an iteration as a *pivot step*. This step is illustrated in Figure 3. To show that MAX-FLOW(G_0, c, s, t) takes $O(m \log n)$ time, we show that there are at most $3m$ pivot steps. It therefore follows that the algorithm runs in $O(m \log n)$ time.

Invariant 3.2. *For every arc a , exactly one of $\langle a, 1 \rangle$ and $\langle a, -1 \rangle$ is in exactly one of T and T^* .*

Proof that the algorithm maintains the invariant. Steps 2 and 4 establish the invariant, and Steps 10 and 11 preserve it. \square

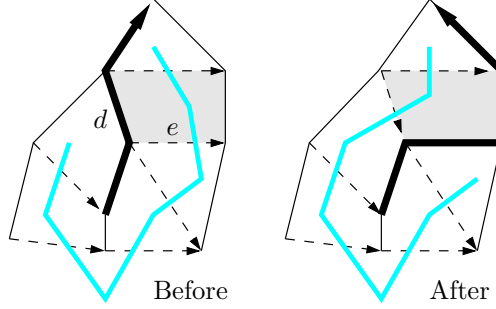


Figure 3: The edges of T are solid, nontree edges are dashed. T^* is represented by light edges. In an iteration of MAX-FLOW, the s -to- t path (bold) is saturated and d is the rootmost nonresidual edge. The shaded face is the face whose parent in T^* changes, $head_{G^*}(d)$. The parent dart e is removed from T^* and inserted into T , and $rev(d)$ is inserted into T^* and becomes the new parent dart.

Invariant 3.3. T^* is a rooted tree whose darts are all oriented towards the root f_∞ .

Proof that the algorithm maintains the invariant. First we show that the invariant holds initially. Since T^* is composed of edges not in T , T^* is a spanning tree. Now we consider orientations. Let a be any arc not in T . By construction of T , the path of arcs $a \circ T[head_G(a)]$ is right of $T[tail_G(a)]$. Let $C = a \circ T[head_G(a), tail_G(a)]$: C is a simple c.c.w. cycle. The face to the left of a is enclosed by C and the face to the right is not. In G^* , a is directed out of C . Since a is the only arc on the boundary of C that is not in T and f_∞ is not enclosed by C , $\langle a, 1 \rangle$ points towards f_∞ in T^* .

Next, note that, in each nonterminating pivot step, $tail_{G^*}(d)$ is not a descendant in T^* of $head_{G^*}(d)$ and so $head_{G^*}(rev(d))$ is not a descendant in T^* of $tail_{G^*}(rev(d))$. Dart e , the parent of $head_{G^*}(d) = tail_{G^*}(rev(d))$, is removed from T^* . The component of $T^* - \{e\}$ that contains f_∞ contains $head_{G^*}(rev(d))$ and not $tail_{G^*}(rev(d))$ and so inserting $rev(d)$ into T^* maintains the invariant. See Figure 3 for an illustration of this. \square

We show that Steps 11 and 12 maintain T as a directed tree.

Invariant 3.4. T is a directed spanning tree.

Proof that the algorithm maintains the invariant. The invariant holds initially by construction. Let d be the rootmost nonresidual dart in $T[s]$ and let e be the dart chosen by the pivot step.

Let $h = tail_{G^*}(e)$ (note that $z = head_{G^*}(d)$ also) and let z be the least common ancestor of $head_G(e)$ and $tail_G(e)$ in T . The cycle $C = e \circ T[head_G(e), z] \circ rev(T[z, tail_G(e)])$ is simple. Since e is the only nontree dart of C and e is directed in T^* towards f_∞ , C must enclose h since C cannot enclose f_∞ . It

follows that $d \in T[\text{head}_G(e), z]$ (rather than $T[z, \text{tail}_G(e)]$) and $T[z, \text{tail}_G(e)]$ does not change during the pivot. So e is directed towards t in T after the pivot. The darts of $T[\text{head}_G(e), \text{tail}_G(d)]$ are now directed away from t in T , but Step 12 corrects this. \square

Invariant 3.5. *Darts in T^* are residual and their reverses are nonresidual.*

Proof that the algorithm maintains the invariant. By construction of G_0 , all arc capacities are strictly positive. Initially f is the all-zero st -flow, so arcs are residual and anti-arcs are nonresidual. Since T^* initially consists of darts in the same direction as their corresponding arcs, this shows the invariant holds initially.

In Step 6, the dart d is nonresidual so $\text{rev}(d)$ is residual. Therefore the insertion of $\text{rev}(d)$ into T^* in Step 9 preserves the invariant. \square

We say that a dart d is a *nontree dart* if $d \notin T$ and $\text{rev}(d) \notin T$.

Lemma 3.6. *There is no clockwise simple cycle whose nontree darts are all residual.*

Proof. Suppose for a contradiction that C was such a cycle. Let S be the set of nontree darts in C . By Invariant 3.2, for every dart $d \in S$, the tree T^* contains either d or $\text{rev}(d)$. Since every dart in S is residual, Invariant 3.5 implies that T^* contains every dart in S . Since C is clockwise, $\text{head}_{G^*}(d)$ is enclosed by C for every dart d in C . Since T is a tree, C contains at least one nontree dart $d \in S$. The directed path $T^*[\text{head}_{G^*}(d)]$ is completely enclosed by C , implying that C also encloses $f_\infty = \text{end}(T^*[\text{head}_{G^*}(d)])$, a contradiction. \square

We say that a flow \mathbf{f} contains a cycle of flow if there is a cycle C such that $\mathbf{f}[d] > 0$ for each dart $d \in C$. Otherwise \mathbf{f} is acyclic.

Corollary 3.7. *The flow is acyclic.*

Proof. If there is a clockwise flow cycle, the residual graph has a clockwise cycle of arcs, contradicting the definition of G_0 (Lemma 3.1). If there is a counterclockwise flow cycle, the reverse of the cycle is a clockwise residual cycle, contradicting Lemma 3.6. \square

We show in the next two corollaries that the network-simplex version of MAX-FLOW implements the abstract version. Corollary 3.8 shows that the leftmost residual s -to- t path is augmented in each iteration and Corollary 3.9 shows that the algorithm is correct.

Corollary 3.8. *For every vertex v , there is no residual path strictly left of $T[v]$.*

Proof. Suppose for a contradiction that there is a residual path strictly left of $T[v]$. Then the leftmost residual v -to- t path, P must be strictly left of $T[v]$. Let Q be a subpath of P such that the endpoints of Q are on $T[v]$ but Q and $\text{rev}(Q)$ are both edge disjoint from $T[v]$. Since P is leftmost, Q is left of $T[\text{end}(Q), \text{start}(Q)]$. So, $Q \circ \text{rev}(T[\text{end}(Q), \text{start}(Q)])$ is a simple c.w. cycle whose nontree darts are residual, contradicting Lemma 3.6. \square

Corollary 3.9. *The st -flow \mathbf{f} returned by the algorithm is maximum.*

Proof. When the algorithm terminates in Step 7, $tail_{G^*}(d)$ is a descendent in T^* of $head_{G^*}(d)$. Let C be the simple cycle $d \circ T^*[head_{G^*}(d), tail_{G^*}(d)]$. In the primal G , the darts of C form a directed cut $\Gamma_G^+(S)$. Every dart in C except d is a nontree dart, so the $head_G(d)$ -to- t path in T does not use any dart in C or the reverse of any dart in C . Since t is on the infinite face, C does not enclose t and so S does not contain t . Likewise the s -to- $tail_G(d)$ path in T does not use and dart in C or the reverse of any dart in C . Since d crosses C , S contains s . Since every dart comprising the cut is nonresidual, there is no residual s -to- t path. \square

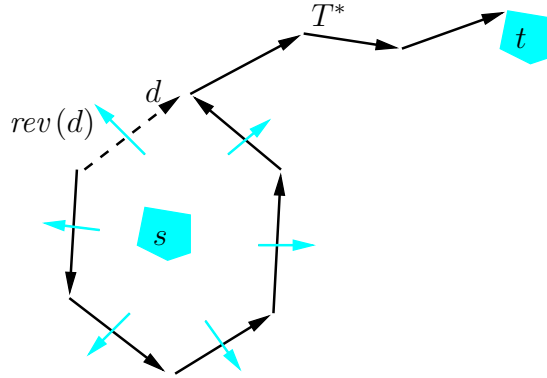


Figure 4: An illustration of Corollary 3.9. Darts of the dual tree are dark, with darts of T^* solid. In the dual, s and t are faces, shown shaded. Upon termination, $tail_{G^*}(d)$ is a descendent in T^* of $head_{G^*}(d)$. Since $rev(d)$ is nonresidual and the reverses of dual tree darts are nonresidual, the cycle shown is a saturated cut separating s from t . The darts of this cut (in the primal) are light.

4 Analysis

We now show that there are at most $3m$ pivot steps in the MAX-FLOW algorithm. Let d be a dart. We have the following facts with regards to the MAX-FLOW algorithm:

- Fact 1 If d is residual at time i and nonresidual at time j ($i < j$), there was an augmentation including d at some time between i and j .
- Fact 2 If d is nonresidual at time i and residual at time j ($i < j$), there was an augmentation including $rev(d)$ at some time between i and j .
- Fact 3 When it is inserted into T , d is residual by Invariant 3.3.

Fact 4 When it is ejected from T , d is nonresidual.

In Section 5 we will prove the following theorem:

Theorem 4.1 (Unusability Theorem). *Let a be an arc of G_0 . If dart $\langle a, 1 \rangle$ belongs to an augmentation at time i and dart $\langle a, -1 \rangle$ belongs to an augmentation at time j ($j > i$), then dart $\langle a, 1 \rangle$ cannot belong to an augmentation at any time after j .*

Claim 4.2. *A dart $\langle a, 1 \rangle$ where a is an arc of G_0 is ejected at most once.*

Proof. Let a be an arc and let $d = \langle a, 1 \rangle$. Suppose for a contradiction that a is ejected at time i_1 and at time i_2 ($i_1 < i_2$).

To be ejected at time i_1 , d must be nonresidual by Fact 4. Fact 1 implies that there was an augmentation including d at some time k_0 where $0 < k_0 < i_1$.

To be ejected at time i_2 , d must have been inserted at some time j_1 where $i_1 < j_1 < i_2$. At time j_1 , d is residual by Fact 3. By Fact 2, there was an augmentation including $rev(d)$ at some time k_1 where $i_1 < k_1 < j_1$.

Since there was an augmentation including d at time k_0 and there was an augmentation including $rev(d)$ at time $k_1 > k_0$, d cannot be augmented after time k_1 by the Unusability Theorem.

Finally, to be ejected at time i_2 , d must be nonresidual by Fact 4. By Fact 2, there was an augmentation including d at some time k_2 where $j_1 < k_2 < i_2$. But d cannot be augmented after time k_1 . This is a contradiction. \square

Corollary 4.3. *A dart $\langle a, -1 \rangle$ where a is an arc of G_0 is ejected at most twice.*

Proof. Let a be an arc and let $d = \langle a, -1 \rangle$.

Suppose d is ejected at times i_1 and i_2 . Then d must be inserted at time $i_1 < j_1 < i_2$. By Fact 4, d is nonresidual at time i_1 and by Fact 3, d is residual at time j_1 . By Fact 2, $rev(d)$ must be part of an augmentation at some time k_1 where $i_1 < k_1 < j_1$.

Likewise, by Fact 4, d is nonresidual at time i_2 and by Fact 1 d must be augmented at time k_2 where $j_1 < k_2 < i_2$.

At time i_2 , d is out of the tree and nonresidual. Since $rev(d)$ cannot be augmented after time k_2 by Claim 4.2, d can never become residual again and so cannot be inserted or ejected again. \square

As a consequence of the above, we have the following theorem:

Theorem 4.4. *There are at most $3m$ pivot steps in the MAX-FLOW algorithm.*

5 Proof of the Unusability Theorem

In this section we will prove the Unusability Theorem. The structure of the proof is as follows. We define what it means for an arc to be *unusable*. We show that a leftmost augmenting path contains no unusable arcs. We show that an

arc a becomes unusable once $rev(a)$ is part of an augmentation. Finally, we show that an unusable arc remains unusable.

Since the augmentations of the algorithm are always along leftmost residual paths (Corollary 3.8), we have a number of properties which we give in the following lemma and which will be used in the remainder of this section.

Lemma 5.1 (Prohibited augmentations). *The following situations are not permitted if A is a leftmost augmentation and the given vertex indices are well-defined:*

1. $A[x, y]$ is right of a residual path $R[x, y]$.
2. $A[x, y]$ makes a clockwise cycle with residual path $R[y, x]$.
3. A has a dart that enters a simple t -to- s residual path R from the right.

Proof. We prove each part separately.

1. $A[s, x] \circ R[x, y] \circ A[y, t]$ is left of A . This contradicts the requirement that A is the leftmost residual path.
2. This contradicts Lemma 3.6.
3. Suppose uv is a dart of a leftmost augmentation path and suppose uv enters a t -to- s residual path R from the right. As such, $uv \notin R$ and $rev(uv) \notin R$. $A[s, u]$ must intersect R at some vertex: let x be the last intersection of $A[s, u]$ with R . If $x \in R(v, s]$, then $A[x, v] \circ R[v, x]$ is a clockwise residual cycle, contradicting Lemma 3.6. If $x \in R(t, v)$ then $R[x, v]$ is a residual path that is left of $A[x, v]$, which is a prohibited augmentation of the second kind. \square

Now we define what it means to be unusable. Unusability is given by a structure in the residual graph called an *obstruction*.

Definition 5.2 (Unusable Arc). *An obstruction cycle (or more simply, an obstruction) is a clockwise non-self-crossing cycle $L \circ M$ where L is residual and M consists entirely of arcs. We say it is an obstruction for an arc a if $\langle a, 1 \rangle$ is the first dart of L . We say an arc a is unusable if there is an obstruction for a .*

We give a second, equivalent representation for an obstruction which will be useful in proving the Unusability Theorem.

Lemma 5.3. *Any obstruction for a can be written as $Q^1 \circ Q^2 \circ R$ where*

1. $Q^1 \circ Q^2$ consists entirely of arcs,
2. $Q^2 \circ R$ is residual,
3. a is the first dart of Q^2 ,
4. there is flow through $start(R)$, and

5. there is flow through $\text{end}(R)$.

Proof. Let $L \circ M$ be an obstruction for a . Since G_0 has no clockwise cycles, $L \circ M$ cannot consist entirely of arcs. Let b be the first anti-arc of L . By Lemma 3.6, $L \circ M$ cannot consist entirely of residual darts and so M cannot consist entirely of residual darts. Let c be the first nonresidual dart of M .

Let $Q^1 = M[\text{tail}(c), \cdot]$, let $Q^2 = L[\cdot, \text{tail}(b)]$, and let $R = L[\text{tail}(b), \cdot] \circ M[\cdot, \text{tail}(c)]$. By choice of b , $L[\cdot, \text{tail}(b)]$ consists entirely of arcs, so property 1 holds. By choice of c , $M[\cdot, \text{tail}(c)]$ is residual, so property 2 holds. Since a is the first dart of L , property 3 holds. Since b is a residual anti-arc, $\text{rev}(b)$ carries flow, so property 4 holds. Since c is a nonresidual arc, it carries flow, so property 5 holds. \square

Definition 5.4. For an unusable arc a , let Δ_a denote the obstruction for a that encloses the minimum number of faces (breaking ties arbitrarily). Write Δ_a as $Q_a^1 \circ Q_a^2 \circ R_a$, and let Q_a denote $Q_a^1 \circ Q_a^2$.

Refer to Figure 5 for an illustration of the relation between Definitions 5.2 and 5.4.

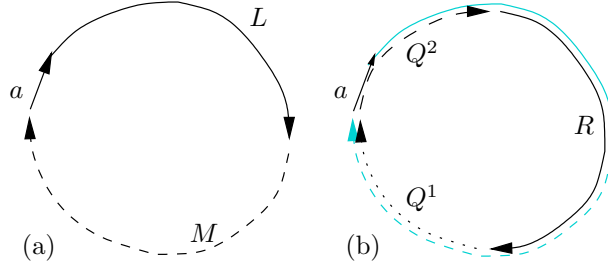


Figure 5: (a) An obstruction for arc a as given by Definition 5.2. (b) The obstruction for arc a as given by Lemma 5.3.

The definition of a minimally enclosing obstruction provides a number of properties of the residual graph.

Property 5.5. Suppose a is unusable. There is no residual path from a vertex in $Q_a^2(\cdot, \cdot]$ to a vertex in Q_a^1 that is enclosed by Δ_a .

Proof. Assume for a contradiction that W is such a residual path. Then $W \circ Q_a^1[\text{end}(W), \cdot] \circ Q_a^2[\cdot, \text{start}(W)]$ is an obstruction for a that encloses fewer faces than Δ_a does. That is, W can be used to replace R_a in the obstruction. \square

Property 5.6. Suppose a is unusable. Q_a^2 belongs to a t -to- s residual path.

Proof. Since Δ_a is a clockwise cycle, it cannot be residual, so Q_a^1 cannot be residual. Let b be the last nonresidual dart of Q_a^1 . Since Q_a^1 contains only arcs, b carries flow and this flow must be routed to t . Let F_t be any $\text{head}(b)$ -to- t flow

path and let F_s be any s -to- $start(R_a)$ flow path. Since the reverse of a flow path is residual, the path $rev(F_t) \circ Q_a^1[head(b), \cdot] \circ Q_a^2 \circ rev(F_s)$ is a residual t -to- s path. \square

Property 5.7. *There are no flow paths enclosed by Δ_a between vertices on the boundary of Δ_a .*

Proof. Assume for contradiction that F is such a simple flow path. Let $\alpha = start(F)$ and $\beta = end(F)$. Then $C_1 = \Delta_a[\alpha, \beta] \circ rev(F)$ and $C_2 = F \circ \Delta_a[\beta, \alpha]$ are clockwise non-self-crossing cycles, each enclosing fewer faces than Δ_a .

We will refer to the following:

Argument 1 Note that $rev(F)$ is residual. If $\Delta_a[\alpha, \beta]$ were residual then C_1 would be a residual clockwise cycle, contradicting Lemma 3.6. Since all nonresidual darts of Δ_a are in Q_a^1 , we infer that $\Delta_a[\alpha, \beta]$ must include at least one dart of Q_a^1 .

Argument 2 Note that F consists entirely of arcs. If $\Delta_a[\beta, \alpha]$ consisted entirely of arcs then C_2 would be a clockwise cycle in G_0 , a contradiction. Since all anti-arcs of Δ_a are in R_a , we infer that $\Delta_a[\beta, \alpha]$ must include at least one dart of R_a .

There are two cases to consider:

Case 1 $start(R_a)$ is a vertex of $\Delta_a[\beta, \alpha]$. By Argument 1, Q_a^1 is not a subpath of $\Delta_a[\beta, \alpha]$. If a is in $\Delta_a[\alpha, \beta]$ then C_1 is an obstruction enclosing fewer faces than Δ_a . If a is in $\Delta_a[\beta, \alpha]$ then C_2 is an obstruction enclosing fewer faces than Δ_a .

Case 2 $start(R_a)$ is a vertex of $\Delta_a(\alpha, \beta)$. By Argument 2, R_a is not a subpath of $\Delta_a[\alpha, \beta]$, so $end(R_a)$ is outside $\Delta_a[\alpha, \beta]$. By Argument 1, Q_a^1 is not a subpath of $\Delta_a[\beta, \alpha]$, so α is in Q_a^1 . Therefore the first arc of Q_a^2 , which is a , is in $\Delta_a[\alpha, \beta]$, so C_1 is an obstruction enclosing fewer faces than Δ_a .

Each case contradicts the minimality condition of Δ_a . \square

Note that Δ_a can enclose a flow path, but such a flow path must start at s and s must be enclosed by Δ_a . However, we have the following:

Corollary 5.8. *No flow paths enter Δ_a .*

Proof. Since t is on the infinite face, any flow path entering Δ_a must exit Δ_a to reach t . Such a flow path violates Property 5.7. \square

For an unusable arc a there is a $start(R_a)$ -to- t flow path and an s -to- $end(R_a)$ flow path by Parts 4 and 5 of Lemma 5.3.

Corollary 5.9. *For an unusable arc a , any $start(R_a)$ -to- t flow path does not intersect any s -to- $end(R_a)$ flow path.*

Proof. Let F_t be any $start(R_a)$ -to- t flow path and let F_s be any s -to- $end(R_a)$ flow path. Suppose for a contradiction that F_t and F_s share a vertex. Let w be the first such vertex in F_t . Let F'_s be the maximal suffix of F_s that is not internal to Δ_a . By Corollary 5.8, F'_s is the only part of F_s that is not internal to Δ_a . Since no arc of F_t is interior to Δ_a , w must be a vertex of F'_s .

Let $F = F_t[start(R_a), w] \circ F'_s[w, end(R_a)]$. F is a $start(R_a)$ -to- $end(R_a)$ flow path that is not internal to Δ_a . By the Noncrossing Theorem, F is either right of or left of R_a . There are two cases.

Case 1 If F is right of R_a , $R_a \circ rev(F)$ is a clockwise residual cycle, contradicting Lemma 3.6.

Case 2 If F is left of R_a , F is also left of $rev(Q_a)$ by transitivity. Hence $F \circ Q_a$ is a clockwise cycle in G_0 , a contradiction. \square

Now we have all the tools needed to prove the Unusability Theorem. We prove it in three parts. First we show that an unusable arc cannot belong to a leftmost residual path (Lemma 5.10). Next we show that if an arc a satisfies the condition of the Unusability Theorem, there is an obstruction for a in the residual graph (Lemma 5.11). Finally we show that obstructions persist under leftmost augmentations (Lemma 5.12) and the Unusability Theorem follows.

Lemma 5.10 (Unusable Arc Consequence). *A leftmost augmenting path contains no unusable arcs.*

Proof. Let A be the leftmost augmenting path, and assume for a contradiction that it goes through an unusable arc a .

The goal is to first construct a non-self-crossing cycle, C , that encloses a and does not enclose t . $A[tail(a), \cdot]$ must therefore cross C . We will show that this results in a contradiction.

By the definition of Δ_a , there is an s -to- $end(R_a)$ flow path. Let F_s be any such path and let $P_1 = Q_a^2 \circ R_a \circ rev(F_s)$. P_1 is a residual $head(a)$ -to- s path. Let A_1 be the maximal suffix of $A[s, tail(a)]$ that does not cross P_1 . Let $C_1 = A_1 \circ P_1$: C_1 is a residual cycle and since there are no c.w. residual cycles, it is c.c.w. C_1 is also non-self-crossing by construction.

We next define another c.c.w. non-self-crossing cycle, C_2 , whose boundary is given by subpaths of Q_a^2 , A , and a flow path not enclosed by Δ_a . If P_1 does not include $start(R_a)$, define $C_2 = C_1$. Note that in this case, P_1 is a subpath of Q_a^2 . Otherwise, we consider the following construction. By the definition of Δ_a , there is a $start(R_a)$ -to- t flow path. Let F_t be any such path and let F'_t be the maximal prefix of F_t that is enclosed by C_1 (possibly the empty path). By Corollary 5.9, F_t and F_s do not share any vertices and by Corollary 5.8, no part of F_t is enclosed by Δ_a . We conclude that $end(F'_t)$ is in A_1 and define $C_2 = F'_t \circ A_1[end(F'_t), \cdot] \circ P_1[\cdot, start(F'_t)]$. Note that $P_1[\cdot, start(F'_t)] = Q_a^2[tail(a), \cdot]$. By definition, C_2 is non-self-crossing.

Notice that in both cases, R_a is not enclosed by C_2 . Let P_2 be the maximal prefix of $R_a \circ Q_a^1$ that is not enclosed by C_2 . Applying the Composition Lemma to

C_2 and $rev(P_2)$, we get a non-self-crossing cycle, C whose boundary is composed of subpaths of F_t , A , $rev(Q_a^1)$, $rev(R_a)$, and possibly $rev(Q_a^2)$. Further, C is c.c.w. and encloses a .

Let A_3 denote the maximal prefix of $A[head(a), \cdot]$ that does not cross C . The last two cases are illustrated in Figure 6.

Case 1 $end(A_3) \in Q_a^2 \circ R_a$. Let $P_3 = Q_a^2 \circ R_a$: P_3 is a boundary of C and since A_3 is enclosed by C , A_3 is right of $P_3[start(A_3), end(A_3)]$, violating Part 1 of Lemma 5.1.

Case 2 $end(A_3) \in rev(F'_t)$. Since $rev(F'_t)$ is a subpath of a t -to- s residual path, this case contradicts Part 3 of Lemma 5.1.

Case 3 $end(A_3) \in Q_a^1$. Let A_4 be the maximal suffix of A_3 that is internal to Δ_a . The only boundary vertices of Δ_a that are not boundary vertices of C are the vertices of Q_a^2 so $start(A_4)$ must be a vertex of Q_a^2 . This case therefore contradicts Property 5.5. \square

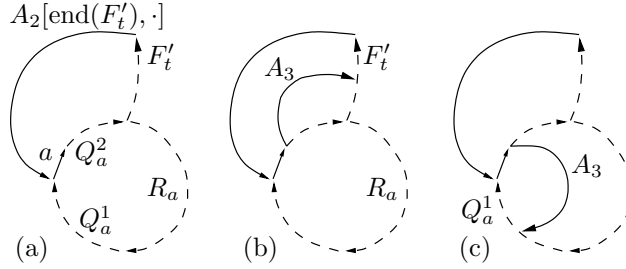


Figure 6: (a) An example of a possible augmentation that uses an unusable arc, a as outlined in Lemma 5.10. In this particular example, $C = A_2[end(F'_t), \cdot] \circ rev(Q_a^1) \circ rev(R_a) \circ F'_t$. (b) The counterexample as described in Case 2 of Lemma 5.10. A cannot escape C via F'_t or R_a . (c) The counterexample as described in Case 3 of Lemma 5.10. A cannot escape C via Q_a^1 .

Lemma 5.11 (Unusable Arc Creation). *If augmentation A uses arc a in the reverse direction, a will be unusable after augmentation.*

Proof. Let a be an arc and let A be the leftmost residual path. Suppose d is a dart in A where $d = rev(a)$. Since a is residual in the reverse direction, a must carry flow. Let F be any s -to- $head(a)$ flow path. Let x be the last vertex of $A[\cdot, tail(a)]$ that is in F . Let $L = rev(A[x, tail(a)])$ and let $M = F[x, tail(a)]$.

Both L and M are simple and by the choice of x , L does not cross M . L is residual after augmentation and a is the first dart of L . M consists entirely of arcs. Since $rev(M)$ is residual before augmentation, $A[x, tail(a)]$ must make a c.c.w. cycle with it by Part 2 of Lemma 5.1. Therefore $M \circ L$ is a c.w. non-self crossing cycle: it is an obstruction for a . \square

Lemma 5.12 (Unusable Arc Persistence). *Once an arc becomes unusable, it remains unusable.*

Proof. Let a be an unusable arc and let A be the leftmost residual s -to- t path. Saturating A can only change the fact that R_a is residual. Assume that A and R_a share a dart.

Let b be the first dart of R_a that is in A . Let A_1 be the maximal suffix of $A[\cdot, \text{tail}(b)]$ that is not enclosed by Δ_a . Since A cannot enter R_a from the right by Part 2 of Lemma 5.1 and since the right of R_a is enclosed by Δ_a , A_1 is not a trivial path.

The following cases are illustrated in Figure 7.

Case 1 $A_1 \neq A[\cdot, \text{tail}(b)]$. Let c be the last dart of A_1 that is enclosed by Δ_a . Both Q_a^2 and R_a belong to t -to- s residual paths (Property 5.6 and by consequence of Lemma 5.3, resp.). Since c is enclosed by Δ_a and so enters Δ_a from the right, $\text{head}(c)$ cannot belong to either Q_a^2 or R_a by Part 3 of Lemma 5.1. So $\text{head}(c)$ is on Q_a^1 . Let $P = Q_a^1[\text{head}(c), \cdot] \circ Q_a^2 \circ R_a[\cdot, \text{tail}(b)]$. Since A_1 does not cross P , A_1 is either left of or right of P .

(a) A_1 is left of P . Let F_t be any $\text{start}(R_a)$ -to- t flow path as guaranteed by Part 4 of Lemma 5.3. This flow path is a part of a t -to- s residual path P' guaranteed by Property 5.6. Let C be the cycle obtained by applying the Composition Lemma to Δ_a and A_1 : this cycle encloses the last dart d of Q_a^2 . Since $d \in P'$ and F_t is in not part enclosed by Δ_a by Property 5.7, P' enters C via a part A_1 . Since C is c.w. by construction, A_1 must enter P' from the right. This contradicts Part 3 of Lemma 5.1.

(b) A_1 is right of P . Since $\text{rev}(A_1)$ is residual after augmentation, $\text{rev}(A_1) \circ P$ is an obstruction for a after augmentation.

Case 2 $A_1 = A[\cdot, \text{tail}(b)]$. Let F_s be any s -to- $\text{end}(R_a)$ flow path. Let A_2 be the maximal suffix of A_1 that does not cross F_s . Let $F'_s = F_s[\text{start}(A_2), \cdot]$. Since $\text{start}(A_2)$ is not enclosed by Δ_a , F'_s starts outside the interior of Δ_a , and so by Property 5.7 no part of F'_s is interior to Δ_a . Let $C = F'_s \circ \text{rev}(R_a[\text{tail}(b), \cdot]) \circ \text{rev}(A_2)$. By the choice of A_2 , C is non-self-crossing. By Part 2 of Lemma 5.1, $\text{rev}(C)$ is c.c.w. and so C is c.w. Let C_1 be the composition of C and Δ_a . Since the interior of C is disjoint from the interior of Δ_a and C_1 is c.w., C_1 is non-self-crossing. Let $Q^1 = F'_s \circ Q_a^1$ and $R = R_a[\cdot, \text{tail}(b)] \circ \text{rev}(A_2)$. Then $C_1 = Q^1 \circ Q_a^2 \circ R$ is an obstruction for a after augmentation since $\text{rev}(A_2)$ is residual after augmentation. \square

6 Closing remarks

We defined the leftmost flow in Section 2.6. An equivalent definition is:

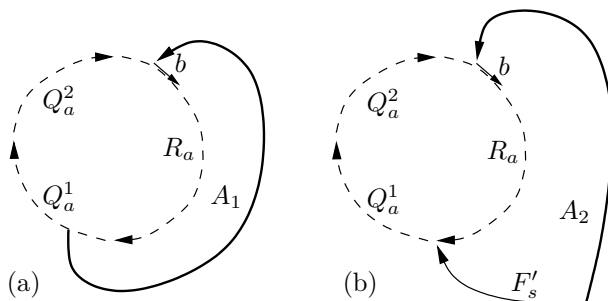


Figure 7: Illustrations of the cases in Lemma 5.12. (a) In Case 1, $Q_a^1[\text{start}(A_1), \cdot] \circ Q_a^2 \circ R_a[\cdot, \text{head}(b)] \circ \text{rev}(A_1)$ is a new obstruction. (b) In Case 2, $F'_s \circ Q_a^1 \circ Q_a^2 \circ R_a[\cdot, \text{head}(b)] \circ \text{rev}(A_1)$ is a new obstruction.

A flow is the leftmost flow of its value if it admits no clockwise residual cycle.

It is easy to see that this definition is equivalent: if a flow \mathbf{f} admits a clockwise residual cycle, one can use this cycle to show that there is another flow \mathbf{f}' that is left of \mathbf{f} . Clearly, the algorithm maintains a leftmost flow. In fact, the preprocessing step, that of finding a 0-value flow which saturates clockwise cycles is equivalent to finding a leftmost 0-value flow. Each iteration of the algorithm finds a leftmost flow of a greater value.

We mention two open problems in maximum flow in planar graphs. The first is maximum flow subject to vertex capacities. The best known result is by Khuller and Naor [16]. Can our algorithm be adapted to address this problem? It is interesting to note that the statements in Section 5 can be applied to this more general case when vertices also have capacities. However, as pointed out in [17], 0-value flows do not have a lattice structure when vertex capacities are introduced, so it is not clear how to find a leftmost flow of value 0, or if a leftmost flow is well defined.

The second problem is maximum flow with multiple sources and/or sinks. As Miller and Naor point out [19], planarity is not reserved by the traditional reduction from multiple-source max flow to single-source max-flow. A natural next step, therefore, is to determine whether our algorithm extends to multiple-source, single-sink max flow. One can assume that there is at most one source in each face of the graph, and under this definition it is possible to define a leftmost flow and define a leftmost source from which to push flow in each iteration of the algorithm. While the algorithm extends to the multiple-source case, the analysis as given does not. Is it possible to extend the analysis?

References

- [1] U. Acar, G. Blelloch, R. Harper, J. Vitter, and S. Woo. Dynamizing static algorithms, with applications to dynamic trees and history independence. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 531–540, 2004.
- [2] S. Alstrup, J. Holm, K. de Lichtenberg, and M. Thorup. Maintaining information in fully dynamic trees with top trees. *ACM Transactions on Algorithms*, 1(2):243–264, 2005.
- [3] T. Biedl, B. Brejová, and T. Vinař. Simplifying flow networks. In *Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science*, volume 1893 of *Lecture Notes in Computer Science*, pages 192–201, 2000.
- [4] D. Eppstein, G. F. Italiano, R. Tamassia, R. E. Tarjan, J. Westbrook, and M. Yung. Maintenance of a minimum spanning forest in a dynamic planar graph. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1 – 11, 1990.
- [5] J. Fakcharoenphol and S. Rao. Planar graphs, negative weight edges, shortest paths, near linear time. In *Proceedings of the 42th Annual Symposium on Foundations of Computer Science*, pages 232–241, 2001.
- [6] C. Ford and D. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.
- [7] G. Frederickson. Fast algorithms for shortest paths in planar graphs with applications. *SIAM Journal on Computing*, 16:1004–1022, 1987.
- [8] A. Goldberg. *Efficient Graph Algorithms for Sequential and Parallel Computers*. PhD thesis, MIT, 1987.
- [9] R. Hassin. Maximum flow in (s, t) planar networks. *Information Processing Letters*, 13:107, 1981.
- [10] R. Hassin and D. Johnson. An $O(n \log^2 n)$ algorithm for maximum flow in undirected planar networks. *SIAM Journal on Computing*, 14:612–624, 1985.
- [11] M. Henzinger and V. King. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *Journal of the ACM*, 46(4):502–516, 1999.
- [12] M. Henzinger, P. Klein, S. Rao, and S. Subramanian. Faster shortest-path algorithms for planar graphs. *Journal of Computer and System Sciences*, 55(1):3–23, 1997.
- [13] A. Itai and Y. Shiloach. Maximum flow in planar networks. *SIAM Journal on Computing*, 8:135–150, 1979.

- [14] D. Johnson. Efficient algorithms for shortest paths in sparse graphs. *Journal of the ACM*, 24:1–13, 1977.
- [15] D. Johnson and S. Venkatesan. Using divide and conquer to find flows in directed planar networks in $O(n^{3/2} \log n)$ time. In *Proceedings of the 20th Annual Allerton Conference on Communication, Control, and Computing*, pages 898–905, 1982.
- [16] S. Khuller and J. Naor. Flow in planar graphs with vertex capacities. *Algorithmica*, 11(3):200–225, 1994.
- [17] S. Khuller, J. Naor, and P. Klein. The lattice structure of flow in planar graphs. *SIAM Journal on Discrete Mathematics*, 6(3):477–490, 1993.
- [18] P. Klein. Multiple-source shortest paths in planar graphs. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 146–155, 2005.
- [19] G. Miller and J. Naor. Flow in planar graphs with multiple sources and sinks. *SIAM Journal on Computing*, 24(5):1002–1017, 1995.
- [20] J. Reif. Minimum s - t cut of a planar undirected network in $O(n \log^2 n)$ time. *SIAM Journal on Computing*, 12:71–81, 1983.
- [21] H. Ripphausen-Lipa, D. Wagner, and K. Weihe. Efficient algorithms for disjoint paths in planar graphs. In W. Cook, L. Lovasz, and P. Seymour, editors, *Combinatorial Optimization: Papers from the DIMACS Special Year*, volume 20 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 295–354. American Mathematical Society, 1995.
- [22] D. Sleator and R. Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26(3):362–391, 1983.
- [23] D. Sommerville. *An introduction to the geometry of n dimensions*. London, 1929.
- [24] R. Tarjan and R. Werneck. Self-adjusting top trees. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 813–822, 2005.
- [25] K. Weihe. Maximum (s, t) -flows in planar networks in $O(|V| \log |V|)$ time. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 178–189, 1994.
- [26] K. Weihe. Maximum (s, t) -flows in planar networks in $O(|V| \log |V|)$ time. *Journal of Computer and System Sciences*, 55(3):454–476, 1997.
- [27] H. Whitney. Planar graphs. *Fundamenta mathematicae*, 21:73–84, 1933.