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

Glencora Borradaile* and Philip Klein

April 14, 2006

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.

*Partially supported by an NSERC PGS-D Fellowship

1 Introduction

The study of maximum st-flow in planar graphs has a long history. In 1957, Berge [3] 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 pushed 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 non-residual, the algorithm requires at most m augmentations, where m is the number of arcs.

In 1979, Itai and Shiloach [14] showed that each iteration of the uppermost path algorithm could be implemented in $O(\log n)$ time, where n is the number of nodes. Consequently, the algorithm can be carried out in $O(n \log n)$ time (using the fact that a simple planar graph with n nodes 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 [15], 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 $O(n)$ 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 [18] 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 $O(n\sqrt{n} \log n)$. In 1989, Miller and Naor [17] 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. In 2001, Fakcharoenphol and Rao [6] 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.

1.1 Weihe's algorithm

Finally, eleven years ago, in a signature accomplishment, Weihe [22] published an $O(n \log n)$ algorithm for planar directed maximum st-flow. It was a tour de force. 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, one satisfying the following three requirements.

1: Each node 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 (i) splicing together every two successive arcs sharing an endpoint of degree one, and (ii) replacing each node of high degree by a cycle, increasing the number of nodes 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 [16] to computing shortest-path distances in the dual.[2]

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

[2] The 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].

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, as pointed out by Biedl, Brjova, and Vinar, [4] 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. 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.

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. Here is an abstract description of our algorithm (after preprocessing).

> 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 describe the algorithm in more detail and prove some invariants. In Section 3, we state Theorem 3.2 and show that it implies the bound on the number of iterations. In Section 4, we prove Theorem 3.2. In Section 5, we give the notation and terminology used throughout the paper.

2 Algorithm

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

Main algorithm:
Designate some face with t on its boundary as the infinite face, denoted f_∞.
Find a circulation in G_{input} with respect to which the residual graph G_0 has no clockwise cycles.
Return MAXFLOW(G_0, c_0, s, t).

The second step is carried out as follows [16]: in the dual graph G^*_{input}, interpret capacities $c_{\text{input}}(d)$ as lengths of darts, and use, e.g., Dijkstra's algorithm to compute shortest-path distances from f_∞. For each face f, let $\phi[f]$ be the f_∞-to-$f$distance. Define a circulation η as follows: for each dart d, define $\eta[d] = \phi[f_L] - \phi[f_R]$, where f_L and f_R are the faces to the left and right of d (as you travel along d).

Next the arc set of G_0 is defined as follows: for each arc a of G_{input}, if $\eta[a] = c(a)$ then add $\text{rev}(a)$ as an arc of G_0, else add a itself. The capacities $c(d)$ are defined by $c(d) = c_{\text{input}}(d) - \eta[d]$.

In what follows, *arc* always refers to an arc of G_0, *anti-arc* refers to a dart whose reverse is an arc, and *dart* can be either. We use $\text{rev}(d)$ or $\text{rev}(P)$ to denote the reverse of a dart d or path P. We use $\circ$ to denote path concatenation.

The procedure MAXFLOW takes a planar graph G_0 with no clockwise cycles such that t is on the boundary of the infinite face. We start with an abstract description of MAXFLOW:

Abstract version of MAXFLOW(G_0, c, s, t)**:**
initialize $\mathbf{f}$ to be the st-flow of value zero.
while there is an s-to-t path that is residual with respect to $\mathbf{f}$,
  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 MAXFLOW(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 spanning tree T of the graph and the corresponding dual spanning tree T^*. We use $T[v]$ to refer to the v-to-root path in T.

Implementation of MAXFLOW(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 nodes not in T.
4 initialize T^* to consist of the set of arcs of G not in T.
4 repeat
5 if $T[s]$ is residual then saturate $T[s]$, modifying $\mathbf{f}$.
6 let d be the last nonresidual dart in $T[s]$.
7 if $\text{head}_{G^*}(\text{rev}(d))$ is a descendent in T^* of $\text{tail}_{G^*}(\text{rev}(d))$ then return $\mathbf{f}$.
8 let e be the parent dart in T^* of $\text{tail}_{G^*}(\text{rev}(d))$.
9 eject e from T^* and insert $\text{rev}(d)$ into T^*.
10 eject d from T and insert e into T.

Right-first search [19] in Step 2 constructs a tree T spanning every node 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, 20, 21], enabling Steps 5, 6, and 10 to be implemented to run in $O(\log n)$ time. The dual tree T^* is represented by an Euler-tour tree data structure [13], so Steps 7, 8 and 9 can be implemented in $O(\log n)$ time.

We refer to an iteration as a *pivot step*. To show that MAXFLOW(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 2.1. *For every arc a, exactly one of a and $\text{rev}(a)$ is in exactly one of T and T^*.*

Proof that the algorithm maintains the invariant: Steps 2 and 3 establish the invariant, and Steps 9 and 10 preserve it. □

Invariant 2.2. *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. It follows from a classical result on planar graphs that, disregarding orientations, 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 $a \circ T[\text{head}_G(a)]$ is right of $T[\text{tail}_G(a)]$. Let z be the least common ancestor in T of $\text{head}_G(a)$ and $\text{tail}_G(a)$. Let $C = a \circ T[\text{head}_G(a), z] \circ \text{rev}(T[\text{tail}_G(a), z])$. 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. a is directed out of C in G^*. Since a is the only arc on the boundary of C that is not in T and f_∞ is not enclosed by C, a points towards f_∞ in T^*.

Next, note that, in each nonterminating pivot step, $\text{head}_{G^*}(\text{rev}(d))$ is not a descendent in T^* of $\text{tail}_{G^*}(\text{rev}(d))$, so Step 9 preserves the invariant. □

Invariant 2.3. *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 positive. Initially f is the all-zeroes st-flow, so arcs are residual and anti-arcs are nonresidual. Since T^* initially consists of 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. □

Say a dart d is a *non-tree dart* if $d \notin T$ and $\text{rev}(d) \notin T$.

Lemma 2.4. *There is no clockwise simple cycle whose non-tree darts are residual.*

Proof. Suppose for a contradiction that C was such a cycle. Let S be the set of non-tree darts in C. By Invariant 2.1, for every dart $d \in S$, the tree T^* contains either d or $\text{rev}(d)$. Since every dart in S is residual, Invariant 2.3 implies that T^* contains every dart in S and the reverse of no 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, S contains at least one dart d. The path $T^*[\text{head}_{G^*}(d)]$ cannot exit from C but ends at f_∞, a contradiction. □

We show in the next two corollaries that the network-simplex version of MAXFLOW implements the abstract version.

Corollary 2.5. *For every node v, there is no residual path strictly left of $T[v]$.*

Proof. Suppose P is a simple v-to-t residual path that is strictly left of $T[v]$. Then the cycle-space vector $\delta(P) - \delta(T[v])$ is clockwise and nonzero. Hence it has an outermost cycle C_o that is clockwise, contradicting Lemma 2.4. □

Corollary 2.6. *The st-flow* **f** *returned by the algorithm is maximum.*

Proof. When the algorithm terminates in Step 7, $\text{head}_{G^*}(\text{rev}(d))$ is a descendent in T^* of $\text{tail}_{G^*}(\text{rev}(d))$. Let C be the reverse of the simple cycle formed by $\text{rev}(d)$ with the path in T^* from $\text{head}_{G^*}(\text{rev}(d))$ to $\text{tail}_{G^*}(\text{rev}(d))$. In the primal G, the darts of C form a directed cut $\Gamma_G^+(S)$. Every dart in C except d is a non-tree dart, so the $\text{head}_G(d)$-to-t path in T^* does not use any dart in C or the reverse of any dart in C. This shows that S does not contain t. Originally there was a residual path from every node in G to t, so the net flow across $\Gamma_G^+(S)$ is positive. Hence S contains s. Since every dart comprising the cut is nonresidual, there is no residual s-to-t path. □

3 Analysis

We now show that there are at most $3m$ pivot steps in the MAXFLOW algorithm.

Lemma 3.1. *A dart is residual when inserted into T.*

Proof. Let d be the rootmost nonresidual dart in $T[s]$. Let e be the edge chosen by the pivot step and let b be the corresponding dart in T^*.

Let $h = \text{tail}_{G^*}(b)$. We also have that $h = \text{tail}_{G^*}(\text{rev}(d))$. Let z be the least common ancestor of $\text{head}(b)$ and $\text{tail}(b)$ in T. $C = b \circ T[\text{head}(b), z] \circ \text{rev}(T[\text{tail}(b), z])$ is a simple cycle. Since b is the only nontree dart of C and since C cannot enclose f_∞, C encloses h. It follows that d is in $T[\text{tail}(b), z]$: $T[\text{head}(b), z]$ is not altered by the pivot step, so b will be a dart in T after pivoting. By Invariant 2.2, b is residual. □

Let d be a dart. We now have the following facts with regards to the MAXFLOW algorithm:

Fact 1: If d is residual at time i and non-residual at time j, there was an augmentation including d at some time between i and j.

Fact 2: If d is non-residual at time i and residual at time j, there was an augmentation including $\text{rev}(d)$ at some time between i and j.

Fact 3: d is residual when it is inserted into T.

Fact 4: d is non-residual when it is ejected from T.

In Section 4 we will prove the following theorem:

Theorem 3.2. *If arc a belongs to an augmentation at time i and dart $\text{rev}(a)$ belongs to an augmentation at time j ($j > i$), then arc a cannot belong to an augmentation at any time after j.*

Claim 3.3. *An arc is ejected at most once.*

Proof. Let a be an arc. a is residual at time 0. 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, it must be non-residual by Fact 4. Fact 1 implies that there was an augmentation including a at some time k_0 where $0 < k_0 < i_1$.

To be ejected at time i_2, a must have been inserted at some time j_1 where $i_1 < j_1 < i_2$. At time j_1, a is residual by Fact 3. By Fact 2, there was an augmentation including $\text{rev}(a)$ 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 $\text{rev}(a)$ at time $k_1 > k_0$, a cannot be augmented after time k_1 by Theorem 3.2.

Finally, to be ejected at time i_2, a must be non-residual 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 a cannot be augmented after time k_1. This is a contradiction. □

Corollary 3.4. *An anti-arc is ejected at most twice.*

Proof. Let d be the reverse of arc a. d is non-residual at time 0.

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

Likewise, by Fact 4, d is non-residual 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 non-residual. Since a cannot be augmented after time k_2 by Claim 3.3, d can never become residual again and so cannot be inserted or ejected again. □

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

Theorem 3.5. *There are at most* $3m$ *pivot steps in the* MAXFLOW *algorithm.*

4 Unusable Arcs

Corollary 4.1. *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. If there is a counterclockwise flow cycle, the cycle is clockwise residual, contradicting Lemma 2.4. □

Lemma 4.2 (Prohibited augmentations). *The following situations are not permitted if A is a leftmost augmentation and the given node 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 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 2.4.

3: Suppose uv is an edge 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 $\text{rev}(uv) \notin R$. $A[s, u]$ must intersect R at some node: certainly $A[s, u]$ intersects R at s. Let x be the last intersection of $A[s, u]$ with R. Either $x \in R(t, v)$ or $x \in R(v, s]$. If $x \in R(v, s]$, then $A[x, v] \circ R[v, x]$ is a clockwise residual cycle. If $x \in R(t, v)$ then $R[x, v]$ is a residual path that is left of $A[x, v]$, violating the property that A is leftmost: $A[s, x] \circ R[x, v] \circ A[v, t]$ is residual and is left of A. □

Definition 4.3 (Unusable Arc). *An* unusability witness cycle *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 unusability witness cycle* for an arc a *if a is the first dart of L. We say an arc a is* unusable *if there is an unusability witness cycle for a.*

Lemma 4.4. *Any unusability witness cycle 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 arc of Q^2,

4: there is flow through $\text{start}(R)$*, and*

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

Proof. Let $L \circ M$ be the unusability witness cycle 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 2.4, $L \circ M$ cannot consist entirely of residual darts. Let c be the first nonresidual arc 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. □

Definition 4.5. *For an unusable arc a, let Δ_a denote the unusability witness cycle 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 2 for an illustration of the relation between Definitions 4.3 and 4.5.

Property 4.6. *Suppose a is unusable. There is no residual path from a node in $Q_a^2(\cdot,\cdot]$ to a node in Q^2 that is enclosed by Δ_a.*

Proof. Assume for a contradiction that W is such a residual α-to-β path. Then $W \circ Q^1[\text{end}(W),\cdot] \circ Q^2[\cdot,\text{start}(W)]$ is an unusability witness cycle for a that encloses fewer faces than Δ_a does. □

Property 4.7. *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 non-residual dart of Q_a^1. b carries flow. Let F_t be any head(b)-to-t flow path and let F_s be any s-to-start(R_a) flow path. $\text{rev}(F_t) \circ Q_a^1[\text{head}(b),\cdot] \circ Q_a^2 \circ \text{rev}(F_s)$ is a residual t-to-s path. □

Property 4.8. *There are no flow paths enclosed by Δ_a between nodes on the boundary of Δ_a .*

Proof. Assume for contradiction that F is such a flow path. Let $\alpha = \text{start}(F)$ and $\beta = \text{end}(F)$. Then $C_1 = \Delta_a[\alpha,\beta] \circ \text{rev}(F)$ 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 2.4. Since all nonresidual darts of Δ_a are in Q_a^1, we infer that $\Delta_a[\alpha,\beta]$ overlaps 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 R_a overlaps $\Delta_a[\beta,\alpha]$.

There are two cases to consider:

Case 1: start(R_a) is a node 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 unusability witness cycle enclosing fewer faces than Δ_a. If a is in $\Delta_a[\beta,\alpha]$ then C_2 is an unusability witness cycle enclosing fewer faces than Δ_a.

Case 2: start(R_a) is a node 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 unusability witness cycle enclosing fewer faces than Δ_a.

Each case contradicts the minimality condition of Δ_a. □

Corollary 4.9. *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 4.8. □

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 4.4,

Corollary 4.10. *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 node. Let w be the first such node in F_t. Let F_s' be the maximal suffix of F_s that is not internal to Δ_a. By Corollary 4.9, 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 node of F_s'.

Let $F = F_t[\text{start}(R_a), w] \circ F_x'[w, \text{end}(R_a)]$. F is a start(R_a)-to-end(R_a) flow path that is not internal to Δ_a. Therefore F does not cross R_a. By Lemma 5.3, F is either right of or left of R_a.

1: If F is right of R_a, $R_a \circ \text{rev}(F)$ is a clockwise residual cycle, contradicting Lemma 2.4.

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

Lemma 4.11. *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. Let F_s be a s-to-end(R_a) flow path. Let $A_1 = A[s, \text{tail}(a)]$, and let $P_1 = Q_a^2 \circ R_a \circ \text{rev}(F_s)$.

Let A_2 be the maximal suffix of A_1 that does not cross P_1. Let $P_2 = P_1[\cdot, \text{start}(A_2)]$ Then $A_2 \circ P_2$ is a non-self-crossing cycle, which we denote by C_1. Since A_2 and P_2 are residual before augmentation, the cycle C_1 must be c.c.w. by Lemma 2.4.

We next define a another c.c.w. non-self-crossing cycle, C_2. If P_2 does not include start(R_a), define $P_3 = P_2$ and $C_2 = C_1$. Suppose P_2 includes start(R_a). Let F_t be a start(R_a)-to-t flow path and let F_t' be the maximal prefix of F_t that is enclosed by C_1. By Corollary 4.10, end(F_t') does not belong to F_s. By Corollary 4.9, the dart of F_t after F_t' is not enclosed by Δ_a. We conclude that end(F_t') is in A_2. In this case we define $P_3 = P_2[\cdot, \text{start}(F_t')]$ and $C_2 = F_t' \circ A_2[\text{end}(F_t'), \cdot] \circ P_3$.

By Lemma 5.1, we can obtain a new c.c.w. non-self-crossing cycle C_3 by combining C_2 with rev(Δ_a) along the common path P_3. Let A_3 denote a maximal prefix of $A[\text{head}(a), \cdot]$ that does not cross C_3. The following cases are shown in Figure 3.

Case 1: end(A_3) is on rev$(F_t') \circ R_a$. Since rev$(F_t') \circ R_a$ is a subpath of the t-to-s residual path rev$(F_t) \circ R_a \circ F_s$, this case contradicts Part 3 of Lemma 4.2.

Case 2: end(A_3) is on Q_a^1. Let A_4 be the maximal suffix of A_3 that is internal to Δ_a. The only boundary nodes of Δ_a that are not boundary nodes of C_3 are the nodes of P_3. Since $A_3[\cdot, \text{start}(A_4)]$ does not leave C_3, start(A_4) must be a node of P_3. This case therefore contradicts Property 4.6. □

Lemma 4.12 (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 $d = \text{rev}(a)$. 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 node of $A[\cdot, \text{tail}(a)]$ that is in F. Let $L = \text{rev}(A[x, \text{tail}(a)])$ and let $M = F[x, \text{tail}(a)]$.

By the choice of x, L does not cross L. 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, $A[x, \text{tail}(a)]$ must make a c.c.w. cycle with it by Part 2 of Lemma 4.2. Therefore $M \circ L$ is a c.w. cycle and is an unusability witness cycle for a. □

Lemma 4.13 (Unusable Arc Persistence). *Once an arc is unusable, it is always unusable.*

Proof. Let a be an unusable arc and let A be the leftmost residual s-to-t path. Augmenting 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 4.2 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 4.

Case 1: $A_1 = A[\cdot, \text{tail}(b)]$. Let F_s be any s-to-endR_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 start$(A_2)$ is not enclosed by Δ_a, F_s' starts outside the interior of Δ_a, and so not part of F_s' is interior to Δ_a by Property 4.8. Let $C_1 = F_s' \circ \text{rev}(R_a[\text{tail}(b), \cdot]) \circ \text{rev}(A_2)$. By the choice of A_2, C is non-self-crossing. By Part 2 of Lemma 4.2, rev(C) is c.c.w. and so C is c.w. Let C_2 be the composition of C_1 and Δ_a. C_2 is c.w. and since the interior of C_1 is disjoint from the interior of Δ_a, C_2 is non-self-crossing. $C_1 = F_s' \circ Q_a \circ R_a[\cdot, \text{tail}(c)] \circ \text{rev}(A_2)$ is an unusability witness cycle for a after augmentation since rev(A_2) is residual after augmentation.

Case 2: $A_1 \neq A[\cdot, \text{tail}(b)]$. Let c be the last dart of A_1 that is enclosed by Δ_a. Since both Q_a^2 and R_a belong to t-to-s residual paths, by Part 3 of Lemma 4.2, head(b) 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. $C = A_1 \circ P$ is a non-self-crossing c.w. cycle. Let F_t be any start$(R_a) - to - t$ flow path and let F_t' be the maximal prefix of F_t that is enclosed in C. Let d be the first dart of F_t' that is not enclosed by C. F_t does not cross P by Property 4.8 since P is in Δ_a. F_t crosses $c \circ A'$ from right to left because F_t' is enclosed by C, F_t is in no part enclosed by Δ_a and c is enclosed by Δ_a. But A cannot cross F_t from left to right by Part 3 of Lemma 4.2. This is a contradiction.

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

The proof of Theorem 3.2 follows from Lemmas 4.11, 4.12, and 4.13.

5 Notation and terminology

5.1 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 $\text{rev}(\cdot)$ to be the function that takes each dart to the corresponding dart in the opposite direction.[3] The head and tail of a dart d (written $\text{head}_G(d)$ and $\text{tail}_G(d)$) 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 $i = 2, \dots, k$, $\text{head}_G(d_{i-1}) = \text{tail}(d_i)$. It is a cycle of darts if in addition $\text{head}_G(d_k) = \text{tail}_G(d_1)$. A path/cycle of darts is *simple* if no node occurs twice as the head of a dart in the path.

If $P = d_1 \dots d_k$ is a path or cycle, 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 nodes of P, we use $P[x, y]$ to denote the subpath P' such that $\text{start}(P') = x$ and $\text{end}(P') = y$.[4] 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.

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 $d_1 \dots d_k e_1 \dots e_\ell$.

A *directed spanning tree* of G is a set T of darts such that (i) every node but one (the *root*) is the tail of exactly one dart, and (ii) there are no cycles. For a node v, $T[v]$ denotes the unique path of darts in T whose start node is v and whose end node is the root. For nodes u and v, $T[u, v]$ denotes the unique u-to-v path in T if v is an ancestor of v in T.

The corresponding undirected spanning tree is the set of arcs represented by darts in T.

5.2 Planar Graphs

According to the traditional, geometric definition, a planar embedding of a graph is a drawing of the graph on the plane or on the surface of a sphere so that nodes 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 nodes or arcs decomposes into connected components, called *faces*. That is, faces are the maximal regions bounded by the embeddings of the nodes 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 nodes 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 rotating the embedding of d clockwise roughly 90 degrees.[5]

[3] Formally, the dart set is $A \times \{\pm 1\}$, and $\text{rev}(\langle a, i \rangle)) = \langle a, -i \rangle$.

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

[5] One can alternatively define embeddings and planar graphs *combinatorially*, without reference to topology. The idea of a combinatorial embedding was implicit in the work of Heffter[11]. Edmonds [5] first made the idea explicit, and Youngs [23] formalized the idea. A combinatorial embedding is sometimes called a *rotation system*. For connected planar graphs, the definition can be shown to be equivalent to the usual one involving geometric embeddings.

We define a finite embedded graph to be a pair $G = \langle \pi, A \rangle$ where A is any given finite set, and π is a permutation of the set of darts corresponding to A. We use $V(G)$ to denote the orbits of π. The elements of $V(G)$ are the *nodes* of G. (Note that nodes are defined in terms of arcs, rather than the other way round.) Each orbit of a permutation is a permutation cycle $(d_1\ d_2\ \dots\ d_k)$. For a dart d, we define $\text{tail}_G(d)$ to be the orbit of π containing d. We define $\text{head}_G(d) = \text{tail}_G(\text{rev}(d))$.

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

5.3 Vector spaces

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

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

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 node v with the permutation cycle 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 permutation cycle of darts forming the counterclockwise boundary of the face, so $\delta(f)$ is the sum of $\delta(d)$ over such darts.[6]

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.

5.4 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 as 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 $\{\delta(f) \ : \ f \text{ a face of } G, f \neq f_0\}$ is a basis for the cycle space of G. We conventionally take f_0 to be the infinite face f_∞. Therefore any circulation η can be represented as a linear combination of these basis vectors. We use ϕ to denote the vector of coefficients for this linear combination, so

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

We call ϕ a *potential assignment*, and we refer to $\phi[f]$ as the *potential* of face f.[7] 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 ϕ 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.) We say a node 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/vertex if the face/dart/vertex is internal to $\delta(C)$.

To define the faces of the embedded graph, we define another permutation π^* of the set of darts by composing π with rev(): $\pi^* = \pi \circ \text{rev}()$. Then the *faces* of the embedded graph $\langle \pi, E\rangle$ are defined to be the orbits of π^*. We say that an embedding π of a *connected* graph G is *planar* if it satisfies Euler's formula: $n - m + \phi$, where n=number of nodes, m=number of arcs, and ϕ=number of faces.

The dual of a connected embedded graph $G_\pi = \langle \pi, E\rangle$ is defined to be the embedded graph $G^* = \langle \pi^*, E\rangle$.

[6] These equatings are formal equalities when we use combinatorial embeddings.

[7] This use of potentials was introduced by Hassin [9] for *st*-planar graphs and by Miller and Naor [17] for general planar graphs.

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

Outermost cycle: For a c.c.w. circulation η, let ϕ be the corresponding potential assignment. Because η is c.c.w., ϕ is nonnegative. Let $F_0 = \{f \ : \ \phi[f] = 0\}$. Let H_0 be the subgraph of the dual G^* induced by the set F_0, and let $F_{0,\infty}$ be the connected component in H_0 that includes f_∞. Let H be the subgraph of G^* induced by the set of faces not in $F_{0,\infty}$. For each connected component S of H, $\Gamma^+(S)$ is a simple c.c.w. cycle, called an *outermost* cycle of η. Note that η assigns at least one to every dart on an outermost cycle. The outermost cycles of a c.w. circulation are similarly defined.

5.5 The Left/Right Relation

An x-to-y path A is *left of* x-to-y path B if $\delta(A) - \delta(B)$ is a clockwise circulation. Likewise A is *right of* x-to-y path B if $\delta(A) - \delta(B)$ is a counterclockwise circulation. Left of and right of are transitive properties. 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.

Crossings Let x be a node in paths A and B. Let d be the first dart of $A[\cdot, x]$ that is not in B. $A[x, \cdot]$ *enters B on the right* if the faces to the left and right of d are right of B.We likewise define entering on the left and leaving from the left/right. Path A is said to *cross* path B if there is a node x in both A and B such that $A[\cdot, x]$ enters B on the right and $A[x, \cdot]$ leaves B on the left (or vice versa). If A and B do not cross for any node, then they are non-crossing. A cycle is *non-self-crossing* if for every pair of subpaths P and Q of the cycle, P does not cross Q. See Figure 1 for examples of these situations.

Lemma 5.1 (Composition Lemma). *Let $P_1 \circ Q$ and $P_2 \circ \mathrm{rev}(Q)$ be c.c.w. non-self-crossing cycles. Then $P_1 \circ P_2$ is a non-self-crossing cycle.*

Lemma 5.2. *Let P and Q be two distinct simple x-to-y paths that do not cross. Every face of the graph induced by P and Q is bounded by a single subpath of P and a single subpath of Q.*

Theorem 5.3 (Non-Crossing Lemma). *If P and Q are two distinct x-to-y paths that do not cross. P is either right of or left of Q.*

5.6 Flow

Given a graph G and nodes s and t, an *st-flow* is a vector $\mathbf{f}$ in arc space such that $\mathbf{f} \cdot \delta(v) = 0$ for every node except s and t.[8] 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 arcs) 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 non-residual. A path/cycle is residual if all its darts are residual. It is well-known that a 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 $\min_{d \in P} c(d) - \mathbf{f}[d]$ then we say the augmentation *saturates* the path P. In this case, at least one dart of P becomes nonresidual.

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

References

[1] Umut A. Acar, Guy E. Blelloch, Robert Harper, Jorge L. Vittes, and Shan Leung Maverick Woo. Dynamizing static algorithms, with applications to dynamic trees and history independence. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 531–540, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[2] S. Alstrup, J. Holm, K. d. Lichtenberg, and M. Thorup. Maintaining information in fully-dynamic trees with top trees. http://arXiv.org/abs/cs/0310065, November 2003.

[3] C. Berge. Two theorems in graph theory. *Proceedings of the National Academy of the Sciences of the U.S.A.*, 43:842–844, 1957.

[4] Therese C. Biedl, Brona Brejova, and Tomas Vinar. Simplifying Flow Networks. In Mogens Nielsen and Branislav Rovan, editors, *Mathematical Foundations of Computer Science 2000 (MFCS)*, volume 1893 of *Lecture Notes in Computer Science*, pages 192–201, Bratislava, August 2000. Springer.

[5] Jack R. Edmonds. A combinatorial representation for polyhedral surfaces. *Notices of the American Mathematical Society*, 7:646, 1960.

[6] J. Fakcharoenphol and S. Rao. Planar graphs, negative weight edges, shortest paths, near linear time. In *Proceedings of the IEEE Foundations of Computer Science*, pages 232–241, 2001.

[7] Greg N. Frederickson. Ambivalent data structures for dynamic 2-edge-connectivity and k smallest spanning trees. In *IEEE Symposium on Foundations of Computer Science*, pages 632–641, 1991.

[8] A. V. Goldberg. *Efficient Graph Algorithms for Sequential and Parallel Computers*. PhD thesis, MIT, 1987.

[9] R. Hassin. Maximum flows in (s, t) planar networks. *Information Processing Letters*, 13:107, 1981.

[10] R. Hassin and D. B. Johnson. An $o(n \log^2 n)$ algorithm for maximum flow in undirected planar networks. *SIAM Journal on Computing*, 14:612–624, 1985.

[11] L. Heffter. Uber das problem der nachbargebiete. *Math. Annalen*, 38:477–508, 1891.

[12] M. R. Henzinger, P. N. 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] Monika R. Henzinger and Valerie King. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *Journal of the ACM*, 46(4):502–516, 1999.

[14] A. Itai and Y. Shiloach. Maximum flow in planar networks. *SIAM Journal on Computing*, 8:135–150, 1979.

[15] D.B. Johnson. Efficient algorithms for shortest paths in sparse graphs. *Journal of the ACM*, 24:1–13, 1977.

[16] S. Khuller, J. Naor, and P. Klein. The lattice structure of flow in planar graphs. *SIAM Journal on Discrete Math*, 6(3):477–490, 1993.

[17] G. L. Miller and J. Naor. Flow in planar graphs with multiple sources and sinks. *SIAM Journal on Computing*, 24(5):1002–1017, 1995.

[18] J. H. 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.

[19] Heike Ripphausen-Lipa, Dorothea Wagner, and Karsten Weihe. The vertex-disjoint menger problem in planar graphs. In *SODA '93: Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 112–119, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[20] D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26(3):362–391, 1983.

[21] Robert E. Tarjan and Renato F. Werneck. Self-adjusting top trees. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 813–822, 2005.

[22] K. Weihe. Maximum (s, t)-flows in planar networks in $o(|v|log|v|)$ time. *Journal of Computer and System Sciences*, 55:454–476, 1997.

[23] J.W.T. Youngs. Minimal imbeddings and the genus of a graph. *Journal of Mathematical Mechanic*, 12:303–315, 1963.

Appendix: Figures

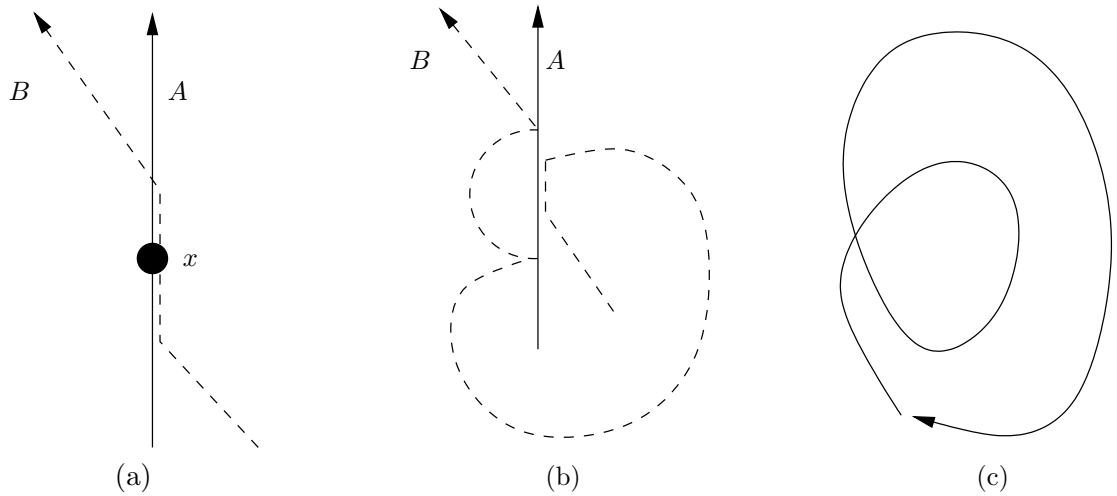

Figure 1: (a) A crosses B: $B[\cdot, x]$ enters A on the right and $B[x, \cdot]$ leaves A on the left. (b) A and B are noncrossing. (c) This is a self-crossing cycle.

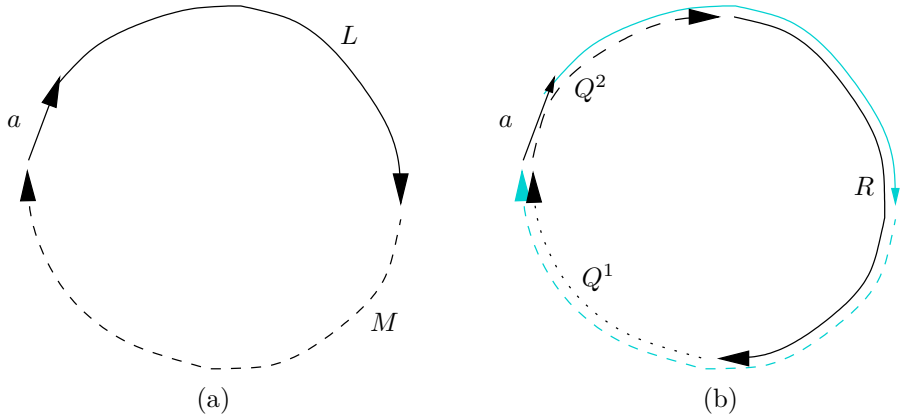

Figure 2: (a) An unusability witness cycle for arc a as given by Definition 4.3. (b) The unusability witness cycle for arc a as given by Lemma 4.4.

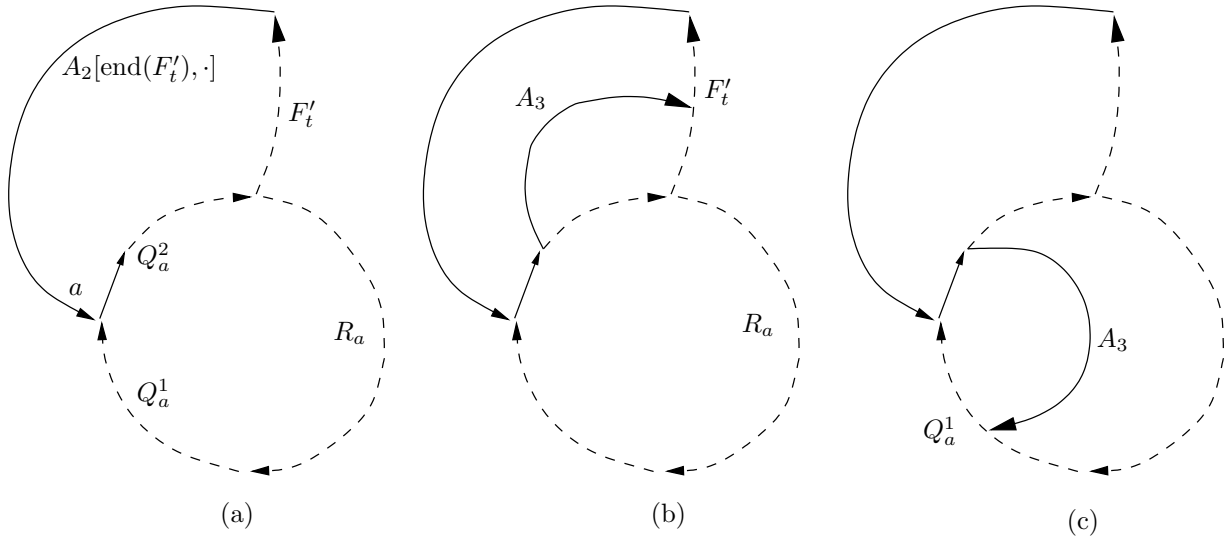

Figure 3: (a) An example of a possible augmentation that uses an unusable arc, a as outlined in Lemma 4.11. In this particular example, $C_3 = A_2[\mathrm{end}(F'_t), \cdot] \circ \mathrm{rev}(Q^1_a) \circ \mathrm{rev}(R_a) \circ F'_t$. (b) The first counterexample as described in Case 1 of Lemma 4.11. A cannot escape C_3 via F'_t or R_a. (c) The second counterexample as described in Case 2 of Lemma 4.11. A cannot escape C_3 via Q^1_a.

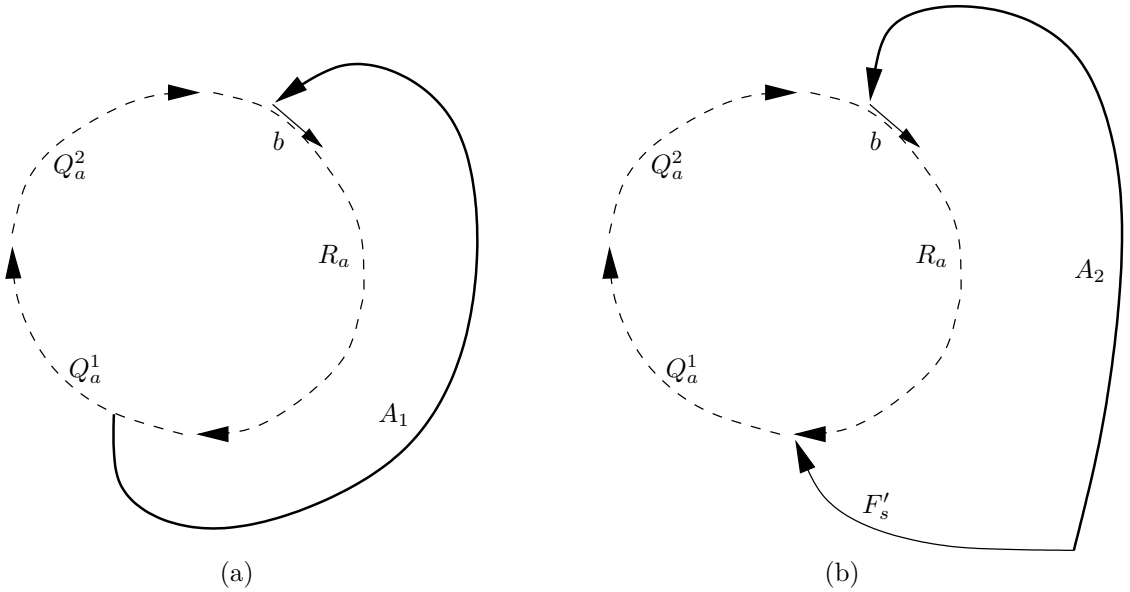

Figure 4: Illustrations of the cases in Lemma 4.13. (a) In Case 1, $Q^1_a[\mathrm{start}(A_1), \cdot] \circ Q^2_a \circ R_a[\cdot, \mathrm{head}(b)] \circ \mathrm{rev}(A_1)$ is the new unusability witness cycle. (b) In Case 2, $F'_s \circ Q^1_a \circ Q^2_a \circ R_a[\cdot, \mathrm{head}(b)] \circ \mathrm{rev}(A_1)$ is the new unusability witness cycle.