Max flow vitality in general and planar graphs

Giorgio Ausiello∗  Paolo G. Franciosa†  Isabella Lari‡  Andrea Ribichini§

Abstract

The vitality of an arc/node of a graph with respect to the maximum flow between two fixed nodes is defined as the reduction of the maximum flow caused by the removal of that arc/node. In this paper we address the issue of determining the vitality of arcs and/or nodes for the network flow problem over various classes of graphs and digraphs. First of all we show how to compute the vitality of all arcs in a general undirected graph by solving \( n-1 \) max flow instances, i.e., in worst case time \( O(n \cdot \text{MF}(n, m)) \), where \( \text{MF}(n, m) \) is the time needed to solve a max-flow instance. In \( st \)-planar graphs (directed or undirected) we can compute the vitality of all arcs and all nodes in \( O(n) \) worst case time. Moreover, after determining the vitality of arcs and/or nodes, and given a planar embedding of the graph, we can determine the vitality of a “contiguous” (w.r.t. that embedding) set of arcs/nodes in time proportional to the size of the set. In the case of general undirected planar graphs, the vitality of all nodes/arcs is computed in \( O(n \log n) \) worst case time, while for the directed planar case we solve the same problem in \( O(np) \), where \( p \) is the number of arcs in a path from \( s^* \) to \( t^* \) in the dual graph.

1 Introduction

Given a graph with capacities associated to arcs, and given two special nodes \( s \) and \( t \), the problem of determining the maximum flow that can be transferred from \( s \) to \( t \) has been deeply studied since the 1950’s. Here we present algorithms for computing how the maximum flow can be influenced by the removal of any single arc, or any single node, or in some cases by the simultaneous removal of a set of arcs/nodes. This is a special case of the vitality concept: given a real-valued function \( f(G) \) of a graph \( G \), the vitality of a resource \( x \) of the graph is usually defined as the value \( |f(G) - f(G \setminus x)| \), where \( G \setminus x \) denotes the graph after the removal of resource \( x \). Vitality can be seen as a special case of a centrality index, as defined in [20].

The vitality of arcs and nodes in a graph has been studied with respect to the distance between two fixed nodes \( x, y \) in a graph \( [7, 21, 22, 23, 25, 26, 31, 32] \). In this case, the vitality of an arc measures the distance increase between \( x \) and \( y \) when the arc is removed, and obviously only arcs on a shortest path from \( x \) to \( y \) may have positive vitality. In [4], the problem of determining a spanner of a graph with the additional constraint of preserving the vitality of arcs has been addressed.

The problem of computing the vitality of arcs with respect to max-flow has been studied since 1963, only a few years after the seminal paper by Ford and Fulkerson in 1956. Wollmer [33] presented a method for determining the most vital link (i.e., the arc with maximum vitality) in a railway network. A more general problem has been studied in [29], where an algorithm is proposed for finding the \( k \) arcs whose simultaneous removal causes the largest decrease in max-flow, based on an enumerative approach. Wood [34] has shown that this problem is NP-hard in the strong

∗Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Università di Roma “La Sapienza”, via Ariosto 25, 00185 Roma, Italy. Email: ausiello@dis.uniroma1.it.
†Dipartimento di Scienze Statistiche, Università di Roma “La Sapienza”, piazzale Aldo Moro 5, 00185 Roma, Italy. Email: paolo.franciosa@uniroma1.it.
‡Dipartimento di Scienze Statistiche, Università di Roma “La Sapienza”, piazzale Aldo Moro 5, 00185 Roma, Italy. Email: isabella.lari@uniroma1.it.
§Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Università di Roma “La Sapienza”, via Ariosto 25, 00185 Roma, Italy. Email: ribichini@dis.uniroma1.it.
sense, while its approximability has been studied in [2]. The same problem can be solved in $O(n^3k)$ on planar graphs [28]. Corley and Chang [8] have shown that removing nodes can be reduced to removing arcs in a transformed network.

A slightly different problem consists in determining a “robust” flow assignment, i.e., a flow assignment in which the flow loss due to the removal of one arc is minimized. In this setting, flow is not “re-routed” due to the arc removal. Aneja, Chandrasekaran and Nair [3] propose a strongly polynomial solution based on LP, while the extension of this problem to the removal of $k$ arcs has been shown to be NP-hard in [9], even for $k = 2$. Recently, the most vital arc or set of $k$ arcs in a flow network which carries flow over time has been studied in [24].

Despite the abundant literature on most vital arcs, the problem of efficiently determining the vitality of all arcs or all nodes has not been addressed yet. For the following classes of graphs we obtain the following results ($n$ is the number of nodes, $m$ is the number of arcs):

**general undirected:** vitality of all arcs in $O(n \cdot \text{MF}(n, m))$ time, where $\text{MF}(n, m)$ is the time needed to solve a max-flow instance on an undirected graph with $n$ nodes and $m$ arcs;

**st-planar, both undirected and directed:** vitality of all nodes and all arcs in $O(n)$ time. Fixing an st-planar embedding, after $O(n)$ preprocessing time and using $O(n)$ space, we retrieve the vitality of a contiguous set of $k$ arcs in $O(k)$ time;

**planar undirected:** vitality of all nodes and all arcs in $O(n \log n)$ time;

**planar directed:** vitality of all nodes and all arcs in $O(np)$, where $p$ is the number of arcs in a path from face $s^*$ to face $t^*$ in the dual of an arbitrary planar embedding of the graph.

## 2 Definitions and preliminaries

We are given a weighted directed graph $G = (N, A, c)$, where $N$ is a set of $n$ nodes, $A \subseteq N \times N$ is a set of $m$ arcs, and $c : A \to \mathbb{R}^+$ is a strictly positive function that assigns capacities to arcs. We fix two special nodes: $s$, the source, and $t$, the sink, and we assume $G$ is connected—i.e., the underlying undirected graph is connected. A feasible flow assignment from $s$ to $t$ is a function $f : A \to \mathbb{R}^+$ such that:

- $0 \leq f(e) \leq c(e)$ for each $e \in A$ (capacity constraint),
- $\sum_{x \in N^-(v)} f((x, v)) = \sum_{y \in N^+(v)} f((v, y))$, for each $v \in N \setminus \{s, t\}$ (conservation constraint),

where $N^-(v)$ (resp., $N^+(v)$) is the set of nodes $\{x \in N \mid (x, v) \in A\}$ (resp., $\{x \in N \mid (v, x) \in A\}$).

The capacity constraint ensures that the flow on each arc does not exceed its capacity, while the conservation constraint ensures that for each node, other than $s$ or $t$, the flow entering the node equals the flow leaving the node. The flow from $s$ to $t$ under a feasible flow assignment $f$ is defined as $F(f) = \sum_{y \in N^+(s)} f((s, y)) - \sum_{x \in N^-(s)} f((x, s))$. A maximum flow from $s$ to $t$ is the maximum value of $F(f)$ over all feasible flow assignments $f$. A flow assignment giving a maximum flow is called a maximum flow assignment. Since $s$ and $t$ are usually fixed, we omit to specify “from $s$ to $t$” in the sequel, and we denote the maximum flow on $G$ from $s$ to $t$ simply by maxFlow($G$).

In an undirected graph $G = (N, A, c)$ (we adopt the same notation for directed and undirected graphs, as in [1]), a feasible flow assignment $f_u$ can be defined by considering a feasible flow assignment $f$ on the directed graph obtained by substituting each undirected arc $(x, y)$ in $A$ by a pair of directed arcs $(x, y)$ and $(y, x)$, both having the same capacity. The flow value $f_u(x, y)$ from $x$ to $y$ on the undirected arc $(x, y)$ is defined as $f_u(x, y) = f((x, y)) - f((y, x))$. Note that the flow is “directed”, so that $f_u(x, y) = -f_u(y, x)$.

An st-cut of a connected graph is a partition of nodes into two subsets $S, T$ such that $s \in S$ and $t \in T$. A cut is also identified by the set of arcs between $S$ and $T$, i.e., the set $A \cap ((S \times T) \cup (T \times S))$. Given a cut $C = (S, T)$ and an arc $e \in A \cap ((S \times T) \cup (T \times S))$, we say that $e$
crosses $C$, and $C$ crosses $e$, as well. The capacity of a cut is defined as $c(S, T) = \sum_{e \in (A \cap (S \times T))} c(e)$, and a minimum st-cut is an st-cut having minimum capacity. Note that the capacity is oriented, so that in general $c(A, B) \neq c(B, A)$. Since nodes $s$ and $t$ are fixed, we denote a minimum st-cut in graph $G$ as $\text{minCut}(G)$. The well known Min-Cut Max-Flow theorem \cite{flow} shows that $\text{maxFlow}(G) = c(\text{minCut}(G))$, for any weighted graph $G$. Given an arc $e = (x, y)$, by $\text{minCut}_e(G)$ we denote a minimum capacity cut among all st-cuts of $G$ that cross $(x, y)$. Obviously, $\text{minCut}_e(G)$ is not necessarily a minimum cut, so $c(\text{minCut}_e(G)) \geq c(\text{minCut}(G))$, and $\text{minCut}_e(G) = c(\text{minCut}(G))$ if and only if $e$ belongs to some minimum cut.

In what follows, given an arc $e \in A$, we shortly denote the graph $G = (N, A \setminus \{e\}, c)$ by $G - e$ and, given a node $v \in N$, we denote the subgraph induced by $N \setminus \{v\}$ by $G - v$. The same notations are extended to sets of arcs or nodes, thus $G - A'$, with $A' \subseteq A$, is the graph $(N, A \setminus A', c)$ and $G - N'$, with $N' \subseteq N$, is the subgraph induced by $N \setminus N'$.

The vitality of a resource $R$ with respect to maximum flow, according to the general concept of vitality in \cite{vitality}, is defined as $\text{flowVit}(R) = \text{maxFlow}(G) - \text{maxFlow}(G - R)$, where $R$ can be a single arc/node, or a set of arcs/nodes.

Given a planar embedded directed graph $G$, its dual graph $G^*$ is defined as a directed weighted multigraph, thus possibly having multiple arcs, whose nodes correspond to faces of $G$ and such that for each arc $e = (x, y)$ in $G$ there is an arc $e^* = (f, g)$ in $G^*$, where $f$ corresponds to the face to the left of $e$ in $G$ and $g$ corresponds to the face to the right of $e$ in $G$. The length of $e^*$ equals the capacity of $e$. For each arc $e$ we also include in $G^*$ a reverse arc, i.e., arc $(g, f)$, whose length is set to 0. We also say that $G$ is the primal graph of $G^*$. The dual graph of a planar embedded undirected graph is a planar undirected multigraph, and is defined analogously (reverse arcs are not needed). It is easy to see that $G^*$ is a planar graph, and that duality also maps each node $v$ in $G$ to a face $v^* \in G^*$, and each face $f$ in $G$ to a node $f^*$ in $G^*$.

The distance $\text{dist}_G(x, y)$ from node $x$ to node $y$ is the length of a shortest path in $G$ from $x$ to $y$. We extend the definition of distance to pairs of node sets, so that $\text{dist}_G(A, B) = \min_{u \in A, v \in B} \text{dist}_G(u, v)$ for any sets $A, B$ of nodes. In particular, the distance from a node $v$ to an arc $e = (x, y)$ is $\text{dist}_G(v, e) = \text{dist}_G(\{v\}, (x, y))$.

An st-planar graph is a planar graph that admits an st-planar embedding, i.e., a planar embedding with nodes $s$ and $t$ on the same face. W.l.o.g., we assume $s$ and $t$ are on the outer face. An st-planar embedding of an st-planar graph can be found in $O(n)$ worst-case time. Given an st-planar embedded graph, in which $s$ is placed to the left and $t$ is placed to the right (see the left part of Figure \ref{fig:st-planar}), we draw two semi-infinite lines from $s$ to the left and from $t$ to the right, splitting the outer face into an upper face $U$ and a lower face $L$, thus $U^*$ and $L^*$ will be two special nodes in $G^*$. We denote by $G^*_r$ the graph obtained from $G^*$ by contracting arc $e^*$ (see the right part of Figure \ref{fig:st-planar}). Since arc lengths are not negative, it follows that $\text{dist}_{G^*_r}(x, y) \leq \text{dist}_{G^*}(x, y)$, for any arc $e$ and any pair of nodes $x, y$. With respect to distances, contracting arc $e = (x, y)$ corresponds to adding a reverse arc $e_r(x, y)$ and to setting the length of $e$ and $e_r$ to zero.

Given an st-planar embedded graph $G$, we say a set $S$ of arcs is contiguous if the set of dual arcs $S^*$ defines a connected component in $G^*$. Note that the contiguity property depends on the embedding of the graph.

2.1 A simple general consideration

All our algorithms rely on the following consideration:

**Lemma 1** For each arc $e$,

$$\text{flowVit}(e) = \max \{0, c(\text{minCut}(G)) - (c(\text{minCut}_e(G)) - c(e))\}$$

**Proof.** By the Min-Cut Max-Flow theorem, $\text{flowVit}(e) = c(\text{minCut}(G)) - c(\text{minCut}(G - e))$. Obviously, $c(\text{minCut}(G - e)) \leq c(\text{minCut}(G))$.

Let us first assume that $c(\text{minCut}(G - e)) < c(\text{minCut}(G))$. In this case, $\text{flowVit}(e) = c(\text{minCut}(G)) - c(\text{minCut}(G - e)) > 0$, and $\text{minCut}(G - e) \cup \{e\}$ is the minimum capacity cut
among all cuts crossing arc $e$. Hence, $\text{minCut}_e(G) \setminus \{e\}$ is a minimum cut in $G - e$, and its capacity is $c(\text{minCut}_e(G)) - c(e)$. Otherwise, let $c(\text{minCut}(G - e)) = c(\text{minCut}(G))$. In this case, by definition, $\text{flowVit}(e) = 0$ and $c(\text{minCut}_e(G)) - c(e) \geq c(\text{minCut}(G))$, thus giving the thesis. □

3 General undirected graphs

The vitality of each arc $e$ can be computed, by definition, by solving a max-flow problem on $G - e$, thus we can compute the vitality of all arcs by $m$ calls to a max-flow routine. We show here that it is possible to solve the same problem by only $n - 1$ calls to a max-flow routine.

Lemma 1 shows that, in order to compute $\text{flowVit}(e)$ for any given arc $e = (x, y)$ in a general undirected graph, it is sufficient to compute $c(\text{minCut}_e(G))$. Let $(C, \overline{C})$ be any st-cut that crosses $e$, then either:

1. $\{x, s\} \subseteq C$ and $\{y, t\} \subseteq \overline{C}$ or
2. $\{y, s\} \subseteq C$ and $\{x, t\} \subseteq \overline{C}$.

Therefore, we can find the minimum capacity cut by comparing the best cut of type [1] and the best cut of type [2]. A minimum capacity cut of type [1] can be found by applying a standard min-cut algorithm to a graph $G'$ obtained from $G$ by adding two arcs $(x, s)$ and $(y, t)$ with very high capacities (e.g., greater than the sum of all the capacities in the graph). Obviously, a minimum st-cut in $G'$ cannot separate $x$ from $s$, nor can it separate $y$ from $t$, thus it necessarily crosses arc $e$. The same argument can be applied to compute minimum capacity cuts of type [2].

Gomory and Hu [12] showed that in any undirected graph a set $\mathcal{C}$ of at most $n - 1$ cuts exists so that for each pair of nodes $x, y$ a minimum $xy$-cut can be found in $\mathcal{C}$. This means that the $\binom{n}{2}$ pairs of nodes in $G$ can be separated by using only $n - 1$ different minimum cuts. Moreover, these cuts can be implicitly represented by a cut tree:

Definition 1 A cut tree $T = (N, A_T, w)$ of a weighted undirected graph $G = (N, A, c)$ is a tree with real weighted arcs that represents minimum capacity cuts for all pairs of nodes in $N$. More precisely, for any two nodes $x, y \in N$, let $e$ be a minimum weight arc in the unique path joining $x$ and $y$ in $T$: then $T$ is a cut tree of $G$ if and only if

(i) $w(e)$ equals the capacity of a minimum $xy$-cut in $G$, and
A first algorithm for computing a cut tree has been proposed in [12], and a simpler approach is shown in [13].

A flow tree differs from a cut tree in the fact that property (ii) is not required. Hence, a flow tree only represents the capacities of the minimum cuts, not the cuts themselves, for all pairs of nodes. A very simple algorithm for computing a flow tree is given in [13]. All the algorithms in [12, 13] require $n - 1$ maximum flow computations.

The concept of flow tree has been generalized by Cheng and Hu in [6]. In their more general setting, an arbitrary real function $f$ is defined on the set of cuts (i.e., node bipartitions) and, given any two nodes $x, y$, an $xy$-cut that minimizes $f$ has to be computed.

**Definition 2** Given an undirected graph $G = (N, A)$ and a real function $f$ defined on the set of all bipartitions of $N$, an ancestor tree $T_f$ is a binary tree with leaves $N$ such that each internal node $v$ of $T_f$ represents a minimum cut (w.r.t. $f$) separating each leaf in the left subtree of $v$ from each leaf in the right subtree of $v$.

The minimum (w.r.t. $f$) cut separating two nodes $x$ and $y$ in $G$ can be found by looking at the lowest common ancestor of $x$ and $y$ in $T_f$. For example, we can define $f$ so that balanced bipartitions are preferred, or impose any other arbitrary constraint and/or cost function to cuts. As a special case, $f$ can be the sum of the capacities of arcs crossing the cut, as in the classical max-flow problem. Cheng and Hu showed that for any directed graph $G$ and any cost function $f$ it is always possible to compute an ancestor tree $T_f$. Assuming that, given a pair $x, y$, a routine is available for computing a minimum $xy$-cut in $G$ according to the cost function $f$, building an ancestor tree requires $n - 1$ calls to that routine—plus overall $O(n^2)$ worst case time restructuring operations.

Note that, while an ancestor tree exists for any cost function $f$, it is not always possible to define a cut tree according to $f$. For example, let us define $f$ as in the classical min cut problem, with the exception that partitions in which one side contains only one node have cost $+\infty$: a cut tree should have at least one leaf and, the cut defined by the arc incident on $v$ in the cut tree would define a partition in which one side contains only node $v$.

We are now ready to describe our algorithm for computing arc vitalities for general undirected graphs. We first compute $c(\text{minCut}(G))$, and then we build an ancestor tree $T_{st}$ according to cost function $f_{st}$ defined as follows:

$$f_{st}(C, \overline{C}) = \begin{cases} +\infty & \text{if } \{s, t\} \subseteq C \text{ or } \{s, t\} \subseteq \overline{C} \\ c(C, \overline{C}) & \text{otherwise} \end{cases}$$

For each arc $e = (x, y)$ in $G$, we find on $T_{st}$ the capacity of a minimum $xy$-cut that also separates $s$ from $t$. Minimizing the cost function $f_{st}$ gives $\text{minCut}_e(G)$, for each $e$, and, by Lemma 1 allows us to compute $\text{flowVit}(e)$ in constant time.

**Theorem 2** Given an undirected weighted graph $G = (N, A, c)$ and two nodes $s, t$, we can compute $\text{flowVit}(e)$, for all $e \in A$, in $O(n \cdot \text{MF}(n, m) + m \cdot n)$ worst-case time, where $\text{MF}(n, m)$ is the time needed to compute a maximum flow.

**Proof.** Building the ancestor tree $T_{st}$ requires $n - 1$ calls to a routine that, given two nodes $x, y$, computes a minimum cut that separates both $x$ from $y$ and $s$ from $t$. As described in the beginning of this section, such a cut can be found by solving two standard max-flow instances, namely, on a graph $G'$ obtained from $G$ by adding two arcs $(x, s)$ and $(y, t)$ with very high capacities and on a graph $G''$ obtained from $G$ by adding two arcs $(x, t)$ and $(y, s)$ with very high capacities.

For each arc $e = (x, y)$, the value $\text{flowVit}(e)$ can be computed by Lemma 1 where $c(\text{minCut}_e(G))$ is found on $T_{st}$ by searching for the lowest common ancestor of $x, y$. This trivially requires $O(n)$ for each arc, leading to an overall $O(m \cdot n)$ additional worst-case time.

1. Lowest common ancestors could be found more efficiently, but in our case this is not the dominant asymptotic cost.
By applying the currently fastest algorithms for max-flow in general graphs, i.e., King, Rao and Tarjan’s algorithm \[19\] for \( m = \Omega(n^{1+\varepsilon}) \) with \( \varepsilon > 0 \), and Orlin’s algorithm \[27\] for sparse graphs, we can state the following result.

**Corollary 3** For any undirected graph \( G = (N, A, c) \), we can compute \( \text{flowVit}(e) \), for all \( e \in A \), in \( O(n^2 \cdot m) \) worst-case time.

## 4 Max flow algorithms for planar graphs

Algorithms for finding the maximum flow in a planar graph vary according to whether the input graph is directed or undirected, and whether it is \( st \)-planar (i.e., it can be drawn on a plane with \( s \) and \( t \) on the same face) or not.

The first algorithm proposed for directed \( st \)-planar graphs is due to Ford and Fulkerson \[11\] and consists in repeatedly saturating the uppermost path of a planar embedding of the graph. Itai and Shiloach \[16\] proposed an \( O(n \log n) \) implementation of this procedure, by using a priority queue for finding the saturating arc of each uppermost path. Later, Hassin \[14\] proved that, if \( G \) is \( st \)-planar and \( G^* \) is the dual of an \( st \)-planar embedding of \( G \), a minimum \( st \)-cut in \( G \) corresponds to a shortest path in \( G^* \) and the maximum flow can be computed in linear time starting from a single source shortest path (SSSP) tree. Using the algorithm by Henzinger et al. \[15\] for the SSSP tree problem in planar graphs, the minimum \( st \)-cut in \( G \) and the corresponding maximum flow can be found in \( O(n) \) time.

If \( G \) is undirected and planar, but not \( st \)-planar, the best approach currently known is due to Reif \[30\], and it is based on the fact that a minimum \( st \)-cut of \( G \) corresponds to a minimum cycle in the dual graph \( G^* \) separating a chosen face \( s^* \) adjacent to \( s \) from a chosen face \( t^* \) adjacent to \( t \), and viceversa (see \[16\]). Applying a divide and conquer technique, Reif’s algorithm computes a max-flow in \( O(n \log^2 n) \). By plugging in the SSSP tree algorithm for planar graphs by Henzinger et al. \[15\], this bound can be improved to \( O(n \log n) \).

If \( G \) is directed and planar, the divide and conquer approach by Reif for the undirected case cannot be applied. For this case, Borradaile and Klein \[5\] presented an \( O(n \log n) \) time algorithm, based on a repeated search of left-most circulations.

## 5 Vitality in directed or undirected \( st \)-planar graphs

### 5.1 Vitality of arcs

We first refer to directed graphs. Exploiting the strong correspondence for \( st \)-planar graphs between flows in \( G \) and distances from \( U^* \) to \( L^* \) in \( G^* \), the definition of \( \text{flowVit} \) gives for each arc \( e \):

\[
\text{flowVit}(e) = \text{dist}_{G^*}(U^*, L^*) - \text{dist}_{G^*}(U^*, L^*)
\]

Obviously, \( \text{dist}_{G^*}(U^*, L^*) \leq \text{dist}_{G^*}(U^*, L^*) \). Fixing an \( st \)-planar embedding of \( G \), we can state the same equality using only distances in \( G^* \).

**Lemma 4** Given an \( st \)-planar embedded directed graph \( G \) and an arc \( e \in G \), we have:

\[
\text{flowVit}(e) = \max \left\{ 0, \text{dist}_{G^*}(U^*, L^*) - (\text{dist}_{G^*}(U^*, e^*) + \text{dist}_{G^*}(e^*, L^*)) \right\}
\]

**Proof.** Graph \( G^*_e \) derives from \( G^* \) after contracting arc \( e^* = (f_1^*, f_2^*) \), where \( f_1 \) and \( f_2 \) are the faces respectively to the left and to the right of \( e \). This also corresponds, in the primal graph, to merging faces \( f_1 \) and \( f_2 \) into a single face \( f \) (see Figure \[1\]). Obviously, \( \text{dist}_{G^*}(U^*, L^*) \leq \text{dist}_{G^*}(U^*, L^*) \). Due to \([1]\), we only need to show that if \( \text{dist}_{G^*}(U^*, L^*) < \text{dist}_{G^*}(U^*, L^*) \) then

\[
\text{dist}_{G^*}(U^*, L^*) = \text{dist}_{G^*}(U^*, e^*) + \text{dist}_{G^*}(e^*, L^*)
\]
If \( \text{dist}_{G^*}(U^*, L^*) < \text{dist}_{G'}(U^*, L^*) \), then the shortest path \( \pi_e \) from \( U^* \) to \( L^* \) in \( G^*_e \) necessarily passes through \( f^* \). Path \( \pi_e \) from \( U^* \) to \( L^* \) is composed by a shortest path in \( G^* \) from \( U^* \) to one among \( f^*_1 \) and \( f^*_2 \), concatenated to a shortest path in \( G^* \) from one among \( f^*_1 \) and \( f^*_2 \) to \( L^* \), thus giving the thesis. \( \Box \)

**Theorem 5** Given a directed or undirected st-planar weighted graph, we can compute \( \text{flowVit}(e) \), for all \( e \in A \), in \( O(n) \) worst-case time.

**Proof.** Let us first consider the directed case. Lemma 4 only uses graph \( G^* \), thus it is possible to compute the vitality of each arc \( e \) without explicitly computing \( G^*_e \). It suffices to store, for each node \( f^* \) in \( G^* \), the pair of distances \( \text{dist}_{G'}(U^*, f^*) \) and \( \text{dist}_{G'}(f^*, L^*) \). These can be computed by means of two single-source shortest path trees, the first in \( G^* \) from \( U^* \) to each other node and the second in the reversal of \( G^* \) from \( L^* \) to each other node. Shortest path trees can be found in planar graphs in \( O(n) \) worst case time using the technique in [15]. The same approach can be applied to the undirected case. \( \Box \)

### 5.2 Extension to contiguous arc failures and node failures

Let \( F \) be a set of contiguous arcs in an st-planar embedding of an st-planar graph \( G \). We recall that the set of the dual arcs \( F^* \) defines a connected subgraph in \( G^* \). In particular, in case \( F \) is the set of all arcs incident on a same node \( v \), then \( F \) is contiguous in any st-planar embedding of \( G \).

By definition, \( \text{flowVit}(F) = \text{maxFlow}(G) - \text{maxFlow}(G - F) \) and, thanks to the Min-Cut Max-Flow theorem and the result in [11], \( \text{flowVit}(F) = \text{dist}_{G'}(U^*, L^*) - \text{dist}_{(G - F)}(U^*, L^*) \). The dual graph \((G - F)^*\) is obtained from \( G^* \) by contracting all arcs in \( F^* \) or, equivalently, setting to 0 the length of all arcs in \( F \). Let \( K^* \) be the set of endpoints of all arcs in \( F^* \); since \( F^* \) is connected, each node in \( K^* \) has the same distance from \( U^* \). Thus, \( \text{dist}_{(G - F)}(U^*, x^*) = \min_{x^* \in K^*}(\text{dist}_{G'}(U^*, x^*)) \), for each \( x^* \in K^* \).

It follows that \( \text{flowVit}(F) \) can be obtained again using only distances in \( G^* \). In fact,

\[
\text{dist}_{(G - F)}(U^*, L^*) = \min \left\{ \begin{array}{c}
\text{dist}_{G'}(U^*, L^*) \\
\text{dist}_{G'}(U^*, K^*) + \text{dist}_{G'}(K^*, L^*)
\end{array} \right\}
\]

since the portion of any path in \((G - F)^*\) inside \( K^* \) has zero length. By definition, \( \text{dist}_{G'}(U^*, K^*) \) and \( \text{dist}_{G'}(K^*, L^*) \) can be computed in \( O(|K|) \) time as \( \min_{a^* \in K^*}(\text{dist}_{G'}(U^*, a^*)) \) and \( \min_{b^* \in K^*}(\text{dist}_{G'}(b^*, L^*)) \), respectively.

The above argument yields the following theorem:

**Theorem 6** Given an st-planar embedding of a directed or undirected st-planar graph \( G \), it is possible to preprocess \( G \) in \( O(n) \) worst case time so that, for any set of contiguous arcs \( F \), \( \text{flowVit}(F) \) can be answered in \( O(|F|) \) worst-case time.

The arguments leading to Theorem 5 can be specialized in order to compute the vitality of nodes. Deleting a node \( v \) corresponds to deleting the set of all arcs incident on \( v \), and this set of arcs is contiguous in any planar embedding of \( G \). Thanks to equation 2, \( \text{dist}_{(G - v)}(U^*, L^*) \) can be derived by computing \( \text{dist}_{G'}(U^*, F(v)^*) \), where \( F(v) \) is the set of faces adjacent to \( v \). This is the minimum among distances (in the dual graph \( G^* \)) from \( U^* \) to all nodes corresponding to faces surrounding \( v \), and similarly from all these nodes to \( L^* \). Thus, after the two spanning trees from \( U^* \) and to \( L^* \) have been computed in \( O(n) \) time as in [15], we can associate to each node \( v \) the values of \( \text{dist}_{G'}(U^*, F(v)^*) \) and \( \text{dist}_{G'}(F(v)^*, L^*) \), still in overall \( O(n) \) worst case time.

**Theorem 7** Given an st-planar embedding of a directed or undirected st-planar graph \( G \), it is possible to compute \( \text{flowVit}(v) \), for all nodes \( v \), in \( O(n) \) worst case time.
6 General planar graphs

We saw that in the case of st-planar graphs the same techniques yield efficient solutions for the max-flow problem both for undirected and directed graphs, and also the vitality problem for undirected and directed st-planar graphs can be solved by the same algorithm.

This is not true for general planar graphs, therefore we consider separately the undirected and the directed cases. In fact, also for the max-flow problem, the \(O(n \log^2 n)\) algorithm proposed by Reif [30] for undirected planar graphs, which can be improved to \(O(n \log n)\) by applying the SSSP algorithm in [15] (and further improved to \(O(n \log \log n)\) by applying the results of Italiano et al. [17]), does not apply to the directed case, where a more complex approach is required, as it is shown in [5] and [10].

In the following we propose an \(O(n \log n)\) algorithm for computing the vitality of all arcs and all nodes for the case of general undirected planar graphs, while for general directed planar graphs we need \(O(n \cdot p)\) time, where \(p\) is the number of arcs in a path from \(s^*\) to \(t^*\) in the dual graph.

Efficient algorithms for computing a max-flow in general planar graphs (directed or undirected) are based on the following property of the dual graph \(G^*\). We assume a planar embedding of the graph is fixed:

**Property 1 (see Property 1 in [30])** A minimum cut in \(G\) corresponds to a minimum cycle in \(G^*\) that separates face \(s^*\) from face \(t^*\).

A cycle in the dual graph \(G^*\) that separates face \(s^*\) from face \(t^*\) is called an st-separating cycle.

**Property 2 (see [30])** Choose an arbitrary face \(f_s\) adjacent to \(s\) and an arbitrary face \(f_t\) adjacent to \(t\), and let \(\pi\) be a shortest path from \(f_s^*\) to \(f_t^*\) in \(G^*\). Any shortest st-separating cycle \(\gamma\) crosses \(\pi\) exactly once.

Due to Property 2 the length of \(\gamma\) can be found as

\[
c(\gamma) = \min_{f^* \in \pi} \{c(\gamma_f)\}
\]

where \(\gamma_f\) is a minimum st-separating cycle that contains node \(f^*\).

6.1 Vitality in planar undirected graphs

We briefly recall Reif’s algorithm. Reif “cuts” \(G^*\) along a shortest path \(\pi\) defined as in Property 2 obtaining graph \(G^c\) (see Figure 2): each node \(f^*\) in \(\pi\) is split into two nodes \(f_s^*\) and \(f_t^*\), arcs incident on \(f^*\) above \(\pi\) are moved to \(f_s^*\) and arcs incident on \(f^*\) below \(\pi\) are moved to \(f_t^*\). Arcs on \(\pi\) are copied both on \(f_s^*\) and on \(f_t^*\), so that path \(\pi\) is doubled. Then the median node \(f^*\) in \(\pi\) is found and a shortest cycle \(\gamma_f\) among all the st-separating cycles containing \(f^*\) (i.e., a shortest path in \(G^c\) from \(f_s^*\) to \(f_t^*\)) is computed by a SSSP algorithm starting from \(f_s^*\). Cycle \(\gamma_f\) divides \(G^c\) in a left part \(G_l^c\) and a right part \(G_r^c\) (assuming \(s\) is drawn to the left of \(t\)), where the boundary \(\gamma_f\) belongs to both \(G_l^c\) and \(G_r^c\). The same technique is applied recursively to \(G_l^c\) and \(G_r^c\). This gives rise to a recursion tree with depth \(O(\log n)\), where the sum of the sizes of all instances on a same level is \(O(n)\). Thus, the overall worst case time to find the shortest st-separating cycle in \(G^*\) is \(O(\text{SSSP}(n) \cdot \log n)\), where \(\text{SSSP}(n)\) is the time needed to compute a SSSP tree in a planar graph. The complexity of Reif’s algorithm is \(O(n \log n)\) if shortest path trees are computed in \(O(n)\) time as in [15].

Our algorithm for computing vitalities in undirected planar graphs is based on Lemma 1 and an extension of Property 1. More in detail, we can compute \(\text{flowVit}(e)\) for each arc \(e = (x, y)\) by computing a minimum cycle in \(G^*\) that separates both \(x^*\) from \(y^*\) and \(s^*\) from \(t^*\). If we denote by \(\gamma_{f,e}\) a shortest cycle in \(G^*\) that separates \(s^*\) from \(t^*\) and contains both \(f^*\) and \(e^*\), we can state that

\[
\minCut_e(G) = \min_{f^* \in \pi} \{c(\gamma_{f,e})\}
\]
Exploiting \((4)\), we can compute \(\text{flowVit}(e)\) by computing \(\min_{f^* \in \pi} \{c(\gamma_{f,e})\}\).

Let us concentrate on evaluating \(c(\gamma_{f,e})\) for a single \(f^* \in \pi\) and for all \(e\) in \(G\). We build a SSSP tree on \(G\) starting from \(f^* u\) and a SSSP tree on the reversal of \(G\) starting from \(f^* \ell\), where \(f^* u\) and \(f^* \ell\) are the two nodes in which \(f^*\) is split in \(G\) (see Figure 3). We can now evaluate \(c(\gamma_{f,e})\) for all \(e\), analogously to Lemma \(1\) as

\[
c(\gamma_{f,e}) = \text{dist}_{G^c}(f^* u, e^*) + \text{dist}_{G^c}(e^*, f^* \ell)
\]

(5)

If we trivially apply the above procedure to each \(f^* \in \pi\), working each time on the whole graph \(G\) and computing SSSP trees as in \([15]\), we obtain an \(O(n \cdot p)\) algorithm, where \(p\) is the number of arcs in \(\pi\).

We show now that the divide and conquer technique in \([30]\) can also be applied for computing \(\min_{f^* \in \pi} \{c(\gamma_{f,e})\}\). We exploit the following property.

**Property 3** For any \(f^* \in \pi\), let \(G^c_{\ell}\) and \(G^c_r\) be the left and right portions of \(G^c\) defined by \(\gamma_f\), where both \(G^c_{\ell}\) and \(G^c_r\) include cycle \(\gamma_f\): for any \(f^*\) that precedes (resp., follows) \(f^*\) in \(\pi\), and for any arc \(e^*\) in \(G^c_{\ell}\) (resp., \(G^c_r\)), cycle \(\gamma_{f^*,e}\) can be found in \(G^c_{\ell}\) (resp., \(G^c_r\)).

Property \(3\) can be proved as in \([30]\). We give here some intuition: as depicted in Figure 3, if a minimum cycle separating \(s^*\) from \(t^*\) and \(x^*\) from \(y^*\) crosses \(\gamma_f\), then another minimum cycle with the same property exists in \(G^c_{\ell}\). In fact, since \(\gamma_f\) is a shortest cycle containing \(f^*\), then the portion \(a-b\) of \(\gamma_{f,e}\) that lies out of \(G^c_{\ell}\) has the same length as the dotted shortest path joining \(a\) and \(b\).
We are now ready to describe our algorithm for undirected planar graphs. Let \( f^* \) be the median node in \( \pi \), that is split into \( f^*_u \) and \( f^*_t \). We compute a SSSP tree from \( f^*_u \) to each other node in \( G^c \) and a SSSP tree from \( f^*_t \) to each other node in the reversal of \( G^c \). We set an initial value for \( \text{minCut}_c(G) \) as we do for \( st \)-planar graphs using Lemma 4. Thanks to Property 3 any possible improvement to \( \text{minCut}_c(G) \), for each \( e \in G^c_i \), is found by a recursive call to the same procedure on \( G^c_i \), and analogously for each \( e \in G^c_i \).

The complexity analysis of our algorithm proceeds as in [30]. Computing SSSP trees in linear time, as in [15], we have that:

**Theorem 8** Given an undirected planar weighted graph \( G = (N,A) \), we can compute \( \text{flowVit}(e) \), for all \( e \in A \), in \( O(n \log n) \) worst-case time.

A similar approach allows us to compute the vitality of nodes in a general undirected planar graph. Fixing an arbitrary embedding of the graph, let \( F(x) \) be the set of faces adjacent to node \( x \).

Removing from \( G \) all arcs incident to \( x \) corresponds to merging all the faces in \( F(x) \) into a single face, and contracting nodes \( F^*(x) \) into a single node in \( G^* \). Let us denote this contracted dual graph by \( G^*_x \). The vitality of node \( x \in G \) can be evaluated by computing the length of the shortest \( st \)-separating cycle \( \gamma^x \) in \( G^*_x \), which is given by

\[
c(\gamma^x) = \min_{f^* \in \pi} \{ \text{dist}_{G^c}(f^*_u, x^*) + \text{dist}_{G^c}(f^*_t, x^*) \}
\]

where \( f^*_u \) and \( f^*_t \) are the upper and lower copies of \( f^* \) in \( G^c \).

**Theorem 9** Given an undirected planar weighted graph \( G = (N,A) \), we can compute \( \text{flowVit}(x) \), for all \( x \in N \), in \( O(n \log n) \) worst-case time.

**Proof.** Given a node \( x \) in \( G \), we have

\[
\text{flowVit}(x) = \max \left\{ 0, \ c(\gamma) - \min_{f^* \in \pi} \{ \text{dist}_{G^c}(f^*_u, x^*) + \text{dist}_{G^c}(f^*_t, x^*) \} \right\}
\]

where \( \gamma \) is the shortest \( st \)-separating cycle in \( G^* \). The value of \( \text{dist}_{G^c}(f^*_u, x^*) \) and \( \text{dist}_{G^c}(f^*_t, x^*) \) can be found by a recursive algorithm that splits \( \pi \) on the median node \( f^* \) and computes a SSSP trees in \( G^c \) (resp., in the reversal of \( G^c \)) starting from \( f^*_u \) (resp., from \( f^*_t \)). After computing a SSSP tree from \( f^*_u \) (resp., \( f^*_t \)), the value \( \text{dist}_{G^c}(f^*_u, x^*) \) (resp., \( \text{dist}_{G^c}(f^*_t, x^*) \)) is computed in linear time for all faces \( x^* \) by taking the minimum among distances from \( f^*_u \) (resp., from \( f^*_t \)) to nodes in \( F(x)^* \). If SSSP trees are computed as in [15], the overall worst-case time is \( O(n \log n) \). \( \square \)

### 6.2 Vitality in directed planar graphs

The approach we used for undirected planar graphs does not apply to the directed case. In fact (see [13]), known attempts to apply Reif’s recursion to computing max-flow in general directed planar graphs contain flaws, due to the fact that in the directed case Property 2 does not hold and, more in general, it is not always possible to find shortest cycles that do not cross each other. We compute the vitality of all arcs in general directed planar graphs as follows (see Figure 4). We define a path \( \pi \), as described in Property 2. Next, for each node \( f^* \in \pi \), we compute a SSSP tree \( T_f \) rooted at \( f^* \) and, after reversing all arcs in \( G^* \), a second SSSP tree \( T_f^r \), still rooted at \( f^* \). For each arc \( c \) in the input graph, we can now possibly update the length of \( c(\gamma_{f,a}) \), in a manner similar to the general planar undirected case. However, since shortest cycles may cross \( \pi \) more than once in the directed case, special care must be taken when combining distances from \( T_f \) with distances from \( T_f^r \). In particular, it must be ensured that each considered cycle actually separates \( s^* \) from \( t^* \). This may be achieved by slightly modifying the shortest path algorithm used to compute \( T_f \)
Figure 4: Finding a shortest st-separating cycle through $f^*$ in a general directed planar graph. The solid portion of the cycle belongs to $T_f$, while the dashed portion belongs to $T_{f^*}$.

and $T_{f^*}$, so that two distances are kept for each node, one with an even, and another with an odd number of crossings of $\pi$. Moreover, portions of paths approaching $\pi$ must be appropriately managed. When combining distances in order to evaluate $c(\gamma_{f,e})$, it suffices to ensure that the resulting cycle contains an even number of crossings, to guarantee that it separates $s^*$ from $t^*$.

Since in the directed case shortest cycles may cross one another, Reif’s recursion technique does not apply and, after selecting a node $f^* \in \pi$ and finding $T_f$ and $T_{f^*}$, we cannot in general restrict successive computations, for different nodes of $\pi$, to subgraphs of $G^*$. This implies that, for each arc $f^*$ of $\pi$, our algorithm computes a constant number of SSSP trees over the entire graph. If Dijkstra’s algorithm is used (slightly modified, in order to propagate for each node distances with an even and an odd number of crossings of $\pi$, as stated above, we get an $O(p \cdot n \log n)$ worst-case running time, where $p$ denotes the number of arcs in $\pi$. If, instead, a modification of the shortest paths algorithm for planar graphs of Henzinger et al. [15] is used (Henzinger’s algorithm is based on recursively dividing the input graph, and applying Dijkstra-like steps on small enough regions, and may therefore be modified similarly to Dijkstra’s algorithm), this bound is improved by a log factor. The above arguments lead to the following theorem:

**Theorem 10** Given a directed planar weighted graph $G = (N,A)$, we can compute $\text{flowVit}(e)$, for all $e \in A$, in $O(n \cdot p)$ worst-case time, where $p$ is the number of arcs in a path from $s^*$ to $t^*$ in the dual graph.

### 7 Conclusions and further work

In this paper we have shown how to solve the problem of computing vitality of all arcs and all nodes for several classes of graphs. In several cases the performance of our algorithms is optimal. Some questions are left open:

- is it possible to find non-trivial solutions, i.e., better then $O(m \cdot \text{MF}(m,n))$ worst-case time, for computing vitalities of all arcs in general directed graphs?

- is it possible to adapt the algorithm of Borradaile and Klein [3] to the computation of all arc vitalities for planar directed graphs (not st-planar)?

### References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows - theory, algorithms and applications*. Prentice Hall, 1993.

[2] D. S. Altner, Ö. Ergun, and N. A. Uhan. The maximum flow network interdiction problem: Valid inequalities, integrality gaps, and approximability. *Oper. Res. Lett.*, 38(1):33–38, January 2010.
[3] Y. P. Aneja, R. Chandrasekaran, and K. Nair. Maximizing residual flow under an arc destruction. *Networks*, 38(4):194–198, 2001.

[4] G. Ausiello, P. G. Franciosa, G. F. Italiano, and A. Ribichini. On resilient graph spanners. *Algorithmica*, 74(4):1363–1385, 2016.

[5] G. Borradale and P. Klein. An $O(n \log n)$ algorithm for maximum $st$-flow in a directed planar graph. *J. ACM*, 56(2):9:1–9:30, April 2009.

[6] C. K. Cheng and T. C. Hu. Ancestor tree for arbitrary multi-terminal cut functions. *Annals of Operations Research*, 33(3):199–213, 1991.

[7] H. W. Corley and D. Y. Sha. Most vital links and nodes in weighted networks. *Oper. Res. Lett.*, 1(4):157–160, September 1982.

[8] H. W. Corley, Jr. and H. Chang. Finding the $n$ most vital nodes in a flow network. *Management Science*, 21(3):362–364, 1974.

[9] D. Du and R. Chandrasekaran. The maximum residual flow problem: NP-hardness with two-arc destruction. *Networks*, 50(3):181–182, 2007.

[10] J. Erickson. Maximum flows and parametric shortest paths in planar graphs. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA ’10, pages 794–804, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

[11] L. R. Ford, Jr. and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[12] R. E. Gomory and T. C. Hu. Multi–terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.

[13] D. Gusfield. Very simple methods for all pairs network flow analysis. *SIAM Journal on Computing*, 19(1):143–155, 1990.

[14] R. Hassin. Maximum flow in $(s,t)$ planar networks. *Inf. Process. Lett.*, 13(3):107, 1981.

[15] M. R. 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.

[16] A. Itai and Y. Shiloach. Maximum flow in planar networks. *SIAM Journal on Computing*, 8(2):135–150, 1979.

[17] G. F. Italiano, Y. Nussbaum, P. Sankowski, and C. Wulff-Nilsen. Improved algorithms for min cut and max flow in undirected planar graphs. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC ’11, pages 313–322, New York, NY, USA, 2011. ACM.

[18] H. Kaplan and Y. Nussbaum. Minimum s-t cut in undirected planar graphs when the source and the sink are close. In *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, March 10-12, 2011, Dortmund, Germany*, pages 117–128, 2011.

[19] V. King, S. Rao, and R. Tarjan. A faster deterministic maximum flow algorithm. *Journal of Algorithms*, 17(3):447–474, 1994.

[20] D. Koschützki, K. A. Lehmann, L. Peeters, S. Richter, D. Tenfelde-Podehl, and O. Zlotowski. *Centrality Indices*, pages 16–61. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[21] K.-C. Lin and M.-S. Chern. The fuzzy shortest path problem and its most vital arcs. *Fuzzy Sets and Systems*, 58(3):343–353, 1993.
[22] K. Malik, A. K. Mittal, and S. K. Gupta. The $k$ most vital arcs in the shortest path problem. 
*Operations Research Letters*, 9(4):223–227, 1989.

[23] A. W. McMasters and T. M. Mustin. Optimal interdiction of a supply network. 
*Naval Research Logistics Quarterly*, 17(3):261–268, 1970.

[24] S. Morowati-Shalilvand and J. Mehri-Tekmej. Finding most vital links over time in a flow network. 
*An International Journal of Optimization and Control: Theories & Applications (IJOCTA)*, 2(2):173–186, 2012.

[25] E. Nardelli, G. Proietti, and P. Widmayer. A faster computation of the most vital edge of a shortest path. 
*Inf. Process. Lett.*, 79(2):81–85, 2001.

[26] E. Nardelli, G. Proietti, and P. Widmayer. Finding the most vital node of a shortest path. 
*Theor. Comput. Sci.*, 296(1):167–177, 2003.

[27] J. B. Orlin. Max flows in $O(nm)$ time, or better. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC ’13, pages 765–774, New York, NY, USA, 2013. ACM.

[28] C. A. Phillips. The network inhibition problem. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, May 16-18, 1993, San Diego, CA, USA, pages 776–785, 1993.

[29] H. D. Ratliff, G. T. Sicilia, and S. H. Lubore. Finding the $n$ most vital links in flow networks. 
*Management Science*, 21(5):531–539, 1975.

[30] J. H. Reif. Minimum $s - t$ cut of a planar undirected network in $O(n \log^2(n))$ time. 
*SIAM Journal on Computing*, 12(1):71–81, 1983.

[31] B. Schieber, A. Bar-Noy, and S. Khuller. The complexity of finding most vital arcs and nodes. 
Technical report, College Park, MD, USA, 1995.

[32] M. Stahlberg. Finding the most vital edges for shortest paths - algorithms and complexity for special graph classes, 2016. Bachelor Thesis, Technische Universität Berlin.

[33] R. D. Wollmer. Some methods for determining the most vital link in a railway network. 
Technical report, RAND Corporation, 1963.

[34] R. K. Wood. Deterministic network interdiction. *Math. Comput. Model.*, 17(2):1–18, January 1993.