The Structure of Minimum Vertex Cuts

Seth Pettie
University of Michigan, Ann Arbor, MI, USA

Longhui Yin
Tsinghua University, Beijing, China

Abstract

In this paper we continue a long line of work on representing the *cut structure* of graphs. We classify the types of minimum *vertex* cuts, and the possible relationships between multiple minimum vertex cuts.

As a consequence of these investigations, we exhibit a simple $O(\kappa n)$-space data structure that can quickly answer pairwise $(\kappa+1)$-connectivity queries in a κ-connected graph. We also show how to compute the "closest" κ-cut to every vertex in near linear $\tilde{O}(m + \text{poly}(\kappa)n)$ time.

2012 ACM Subject Classification Mathematics of computing → Paths and connectivity problems; Theory of computation → Data structures design and analysis

Keywords and phrases Graph theory, vertex connectivity, data structures

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.105

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2102.06805`

Funding *Seth Pettie*: This work was supported by NSF grants CCF-1637546 and CCF-1815316. *Longhui Yin*: Supported by a grant from Tsinghua University.

1 Introduction

One of the strong themes running through graph theory is to understand the *cut structure* of graphs and to apply these structural theorems to solve algorithmic and data structural problems. Consider the following exemplars of this line of work:

Gomory-Hu Tree. Gomory and Hu (1961) [31] proved that any weighted, undirected graph $G = (V, E)$ can be replaced by a weighted, undirected tree $T = (V, E_T)$ such that for every $s, t \in V$, the minimum s-t cut partition in T (removing a single edge, partitioning V into two sets) corresponds to a minimum s-t cut partition in G. These are sometimes called *cut-equivalent trees* [1].

Cactus Representations. Dinitz, Karzanov, and Lomonosov (1976) [13] proved that all the *global* minimum edge-cuts of any weighted, undirected graph $G = (V, E)$ could be succinctly encoded as an (unweighted) *cactus graph*. A cactus is a connected multigraph in which every edge participates in exactly one cycle. It was proved that there exists a cactus $C = (V_C, E_C)$ and an embedding $\phi : V \to V_C$ such that the minimum edge-cuts in C (2 edges in a common cycle) are in 1-1 correspondence with the minimum edge-cuts of G. A corollary of this theorem is that there are at most $\binom{n}{2}$ minimum edge-cuts.

Picard-Queyrenne Representation. In a *directed* s-t flow network there can be exponentially many min s-t cuts. Picard and Queyrenne (1980) [56] proved that the family $\mathcal{S} = \{S \mid (S, \overline{S})$ is a min s-$t\}$ corresponds 1-1 with the downward-closed sets of a partial order, and is therefore closed under union and intersection.

Block Trees, SPQR Trees, and Beyond. Whitney (1932) [61, 62] proved that the cut vertices (articulation points) of an undirected graph $G = (V, E)$ partition E into single edges and 2-edge connected components (blocks). This yields the *block tree* representation. Di

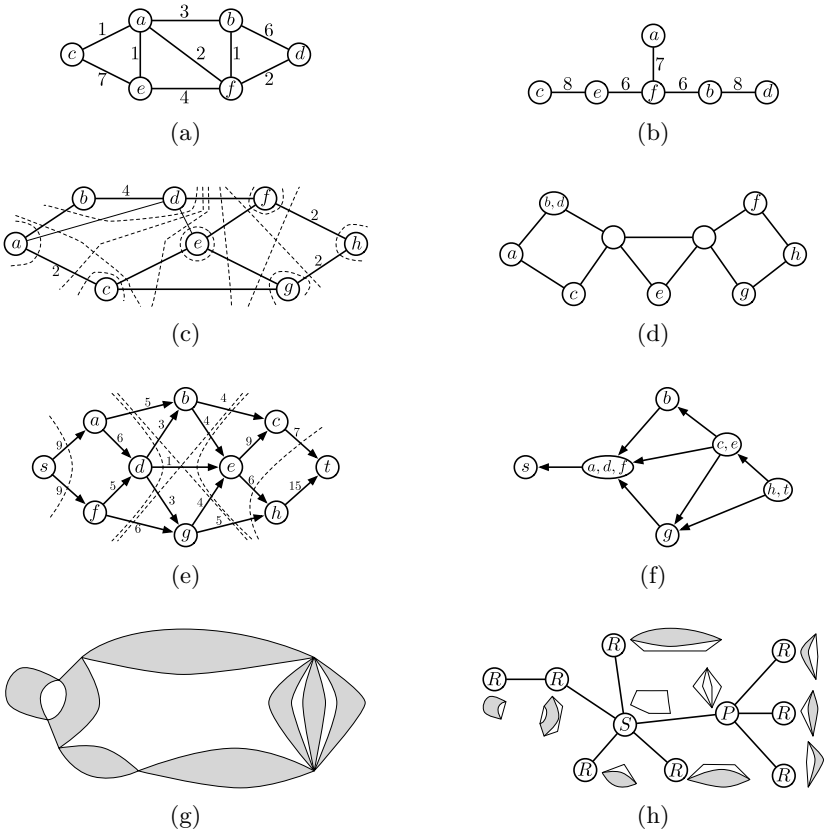

Figure 1 (a) A weighted undirected graph; (b) Its Gomory-Hu (cut-equivalent) tree [31]. (c) A weighted undirected graph (unmarked edges have unit weight); (d) the Cactus representation [13] of its minimum edge cuts. (e) A directed s-t flow network; (f) A dag whose downward-closed sets (that include s but not t) correspond to min s-t cuts (Picard-Queyrenne [56]). (g) An abstract representation of a 2-connected graph; (h) The representation of its 3-connected components as an SPQR tree (Di Battista-Tamassia [4]).

Battista and Tamassia (1989) [5, 4] formally defined the *SPQR tree*, which succinctly encodes all 2-vertex cuts in a biconnected graph, and Kanevsky, Tamassia, Di Battista, and Chen [42] extended this structure to represent 3-vertex cuts in a triconnected graph.[1]

It is natural to ask how, and to what extent, these structures can be extended and generalized. Gusfield and Naor [34] combined the Gomory-Hu tree and the Picard-Queyrenne representation for edge-connetivity. They [33] also described an analogue of Gomory-Hu trees (cut-equivalent trees) for vertex connectivity, i.e., a tree that compactly represents a minimum s-t *vertex* cut for every $s, t \in V$. It used a result of Schnorr [57] on an analogue of Gomory-Hu trees for "roundtrip" flow-values in directed networks. These claims were refuted by Benczur [6], who illustrated that Schnorr's and Gusfield and Naor's proofs were incorrect and could not be rectified. In particular, there are no *cut-equivalent* trees for s-t vertex connectivity and directed s-t cuts. Benczur [6, p. 505-506] suggested a way to construct a *flow-equivalent* tree for vertex connectivity (vertex capacitated s-t flows) using a result of

[1] Many of the structural insights behind [4, 42] were latent in prior work. See, for example. Mac Lane [48], Tutte [59, 60] (1961-6), Hopcroft and Tarjan [36], and Cunningham and Edmonds [12].

Cheng and Hu [9]. This, too, turned out to be incorrect. Hassin and Levin [35] proved that a graph can have $\Omega(n^2)$ s-t vertex-capacitated cut values, which cannot be captured by a flow-equivalent tree. We take these episodes as a reminder that having published proofs (*even incorrect ones*) is essential for facilitating self-correction in science.

The inspiration for this paper is an extended abstract of Cohen, Di Battista, Kanevsky, and Tamassia [11] from STOC 1993. Their goal was to find a cactus-analogue for global minimum vertex cuts, or from a different perspective, to extend SPQR trees [4] and [42] from $\kappa \in \{2, 3\}$ vertex cuts to arbitrarily large κ. As an application of their ideas, they described a data structure for κ-connected graphs occupying space $O(\kappa^3 n)$ that, given u, v, decided whether u, v are separated by a κ-cut or $(\kappa + 1)$-connected. There are no suspect claims in [11]. On the other hand, the paper is 7 pages and leaves many of its central claims unproven.[2] We believe that understanding the *structure* of minimum vertex cuts is a fundamental problem in graph theory, and deserving of a complete, formal treatment.

In this paper we investigate the structure of the set of all minimum vertex cuts and classify the relationships between different minimum vertex cuts. Our work reveals some structural features of minimum κ-cuts not evident in Cohen, Di Battista, Kanevsky, and Tamassia [11], and ultimately allows us to develop a simpler data structure to answer pairwise κ-cut queries in a κ-connected graph. It occupies (optimal) $O(\kappa n)$ space and can be constructed in randomized $\tilde{O}(m + \text{poly}(\kappa)n)$ time, in contrast to [11], which occupies $O(\kappa^3 n)$ space and is constructed in $\exp(\kappa)n^5$ time.[3]

1.1 Related Work

Dinitz and Vainshtein [17, 18] combined elements of the cactus [13] and Picard-Queyrenne [56] representations, which they called the *connectivity carcass*. Given an undirected, unweighted $G = (V, E)$ and $S \subseteq V$ of terminals, λ_S is the size of the minimum edge-cut that separates S. The carcass represents *all* size-λ_S separating cuts in $O(\min\{m, \lambda_S n\})$ space and answers various cut queries in $O(1)$ time.[4]

Benczur and Goemans [7] generalized the cactus representation [13] in a different direction, by giving a compact representation of all cuts that are within a factor 6/5 of the global minimum edge-cut.

Dinitz and Nutov [14] generalized the cactus representation [13] in another direction, by giving an $O(n)$-space representation of all λ and $\lambda + 1$ edge cuts, where λ is the edge-connectivity of the undirected, unweighted graph. Another feature of representations in [17, 18, 14] worth to mention is that they answer connectivity queries with supporting edge insertions in the graph. Unpublished manuscripts [15, 16] give detailed treatments of the λ odd and λ even cases separately.

Georgiadis et al. [30, 22, 28, 29] investigated various notions of 1- and 2-edge and vertex connectivity in *directed* graphs, and the compact representation of edge/vertex cuts.

Gabow [25] provided a $O(m \log n^2/m)$ data structure for all mincuts of a directed graph by drawing a correspondence between cuts and intersecting set families.

---

[2] The full version of this paper was never written (personal communication with R. Tamassia, 2011, and R. Di Battista, 2016).

[3] The algorithm enumerates *all* minimum κ-cuts, which can be as large as $\Omega(2^\kappa(n/\kappa)^2)$; modern vertex connectivity algorithms [23, 27, 26] may reduce the exponent of n in the running time.

[4] The carcass was introduced in extended abstracts [17, 18] and the (simpler) case of odd λ_S was analyzed in detail in a journal article [19]. We are not aware of a full treatment of the case when λ_S is even.

Granot and Hassin [32] generalized the Gomory-Hu tree into node- and arc-capacitated case and gave an algorithm for finding a cut-tree over a set of terminals K by solving $|K|-1$ minimum-cut problems.

Sparsification

One general way to compactly represent connectivity information is to produce a *sparse* graph with the same cut structure. Nagamochi and Ibaraki [50] proved that every unweighted, undirected graph $G = (V, E)$ contains a subgraph $H = (V, E_H)$ with $|E_H| < (k+1)n$ (arboricity $k+1$) such that H is computable in $O(m)$ time and contains exactly the same k'-vertex cuts and k'-edge cuts as G, for all $k' \in \{1, \ldots, k\}$. Benczur and Karger [8] proved that for any capacitated, undirected graph $G = (V, E)$, there is another capacitated graph $H = (V, E_H)$ with $|E_H| = O(\epsilon^{-2} n \log n)$ such that the capacity of *every* cut in G is preserved in H up to a $(1 \pm \epsilon)$-factor. This bound was later improved to $O(\epsilon^{-2} n)$ by Batson, Spielman, and Srivastava [3], which is optimal.

In directed graphs, Baswana, Choudhary, and Roditty [2] considered the problem of finding a sparse subgraph that preserves reachability from a single source, even if d vertices are deleted. They proved that $\Theta(2^d n)$ edges are necessary and sufficient for $d \in [1, \log n]$.

d-Failure Connectivity

An undirected graph can be compactly represented such that connectivity queries can be answered after the deletion of any d vertices/edges (where d could be much larger than the underlying connectivity of the graph). Improving on [54, 43, 20], Duan and Pettie [21] proved that d vertex failures could be processed in $\tilde{O}(d^2)$ time such that connectivity queries are answered in $O(d)$ time, and d edge failures could be processed in $O(d \log d \log \log n)$ time such that connectivity queries are answered in $O(\log \log n)$ time. The size of the [21] structure is $\tilde{O}(m)$ for vertex failures and $\tilde{O}(n)$ for edge failures. Choudhary [10] gave an optimal $O(n)$-space data structure that could answer directed reachability queries after $d \in \{1, 2\}$ vertex or edge failures.

Labeling Schemes

Benczur's refutation [6] of [57, 33] shows that all pairwise vertex connectivities cannot be captured in a *tree* structure, but it does not preclude other representations of this information. Hsu and Lu [37] designed a $O(k \log n)$-bit labeling scheme to determine whether $\kappa(u, v) \geq k$, given just the labels of u and v. This improved [45] and matched an $\Omega(k \log n)$-bit lower bound of Katz, Katz, Korman, and Peleg [44]. By applying it to all $k \in \{1, \ldots, \overline{\kappa}\}$, the Hsu-Lu labeling has size $O(\overline{\kappa}^2 \log n)$ and reports $\min\{\kappa(u, v), \overline{\kappa}\}$. Using a different approach, Izsak and Nutov [38] gave a $O(\overline{\kappa} \log^3 n)$-bit labeling scheme for computing $\min\{\kappa(u, v), \overline{\kappa}\}$. The schemes [37, 45, 44, 38] have large polynomial construction times, and cannot report a u-v cut of size $\kappa(u, v)$

Vertex Connectivity Algorithms

In optimal linear time we can decide whether the connectivity of a graph is $\kappa = 1, \kappa = 2$, or $\kappa \geq 3$ [58, 36]. For larger κ, the state-of-the-art in vertex connectivity has been improved substantially in the last few years. Forster, Nanongkai, Yang, Saranurak, and Yingchareonthawornchai [23] gave a *Monte Carlo* algorithm for computing the vertex connectivity κ of an undirected graph in $\tilde{O}(m + n\kappa^3)$ time, w.h.p.[5] A new result of Li, Nanongkai,

[5] The algorithm does not produce a witness, and hence may err with small probability.

Panigrahi, Saranurak, and Yingchareonthawornchai [46] gave a randomized algorithm running in $O(m^{4/3+o(1)})$ time. The best deterministic algorithm, due to Gao, Li, Nanongkai, Peng, Saranurak, and Yingchareonthawornchai [27], computes the connectivity $\kappa < n^{1/8}$ in $O((m+n^{7/4}\kappa^{O(\kappa)})n^{o(1)})$ time or $O((m+n^{19/20}\kappa^{5/2})n^{o(1)})$ time. For $\kappa > n^{1/8}$, Gabow's algorithm [26] runs in $O(\kappa n^2 + \kappa^2 n \cdot \min\{n^{3/4}, \kappa^{3/2}\})$ time.

The connectivity augmentation problem is to *improve* the global vertex connectivity or specific pairwise connectivities by adding few edges. See, for example, Frank and Jordán [24], Jordán [40], Jackson and Jordán [39], and Nutov [52] for positive results, and Nutov [51] for a hardness of approximation result.

1.2 Organization

In Section 2 we review basic definitions and lemmas regarding vertex cuts. Section 3 gives the basic classification theorem for minimum vertex cuts, and lists some useful corollaries. In short, every pair of cuts have *laminar*, *wheel*, *crossing matching*, or *small* relation. Sections 3.1–3.4 analyze these four categories in more detail. Section 4 exhibits a new $O(\kappa n)$-space[6] data structure that, given two vertices, answers $(\kappa+1)$-connectivity queries in $O(1)$ time, and produces a separating κ-cut (if one exists) in $O(\kappa)$ time. We conclude with some remarks and open problems in Section 5. All missing proofs appear in the full version of the paper [55].

2 Preliminaries

The input is a simple, connected, undirected graph $G=(V,E)$ with $n=|V|$ and $m=|E|$.

Let the subgraph of G induced by A be denoted $G|_A$. We call $U \subset V$ a *cut* if the graph $G|_{V\setminus U}$ is disconnected. A *side* of the cut U is a connected component of $G|_{V\setminus U}$. If P is a side of U and $A \subseteq P$, we say A is *within a side of* U, and let $\mathrm{Side}_U(A) = P$ denote the side containing A. A *region* of a cut U is a side, or the union of several sides of U. Denote $\mathrm{Region}_U(A)$ as the region containing the sides of U that intersects with A. [7] We say a cut *disconnects* or *separates* A and B if they are in distinct sides of U. In particular, if $B = V\setminus(A\cup U)$, we say U *disconnects* or *separates* B *from the rest of the graph.*

A path $\pi = v_1 v_2 \cdots v_l$ is *from* A *to* B, if $v_1 \in A$ and $v_l \in B$. Two paths π, π' from v_1 to v_l are *internally* vertex disjoint if they have no common vertices, except for v_1, v_l. We say U *blocks* π if $U \cap \{v_2, \ldots, v_{l-1}\} \neq \emptyset$.

A k*-cut* is a cut of size k. Define $\kappa(u,v)$ to be the minimum k such that there exists a k-cut separating u and v, where $\{u,v\} \neq E(G)$. Define $\kappa = \kappa(G)$ to be the minimum of $\kappa(u,v)$ over all pairs $\{u,v\} \in \binom{V(G)}{2}\setminus E(G)$. We say G is k*-connected* if $\kappa(G) \geq k$.

In this paper we assume that $\kappa < n/4$ and consider the set of *all* (minimum) κ-cuts.

▶ Remark 1. There is some flexibility in defining the corner cases. Some authors leave $\kappa(u,v)$ undefined when $\{u,v\} \in E(G)$ or define it to be $n-1$. Other authors define connectivity as the maximal number of vertex-disjoint paths. In [11] a k-cut is defined to be a mixed set of edges and vertices whose removal disconnects the graph. Under this definition, when $\{u,v\} \in E(G)$, $\kappa(u,v) = k$ if removing $k-1$ vertices and $\{u,v\}$ disconnects u and v. The last two definitions are equivalent and are compatible with Menger's theorem.

▶ **Theorem 2** (Menger [49]). *Let $G=(V,E)$ be an undirected graph and $\{u,v\}$ a pair* not *in E. Let $U \subset V$ be a minimum size cut disconnecting u and v and Π be a maximum size set of internally vertex disjoint paths from u to v. Then $\kappa(u,v) = |U| = |\Pi|$.*

[6] Formally speaking, this is $O(\kappa n)$ words of space, where a word store the index of a vertex, and takes up $O(\log n)$ bits of space.

[7] Note when A is a singleton set $\{u\}$, $\mathrm{Region}_U(A) = \mathrm{Side}_U(A)$.

The following categories make sense when applied to *non*-minimal vertex cuts, but we are only interested in applying them to *minimum* vertex cuts. Henceforth *cut* usually means *minimum cut*.

Laminar Cuts

Let U be a cut and P be a side of U. If W is a cut and $W \subset U \cup P$, we say W is a *laminar cut* of U in side P.[8]

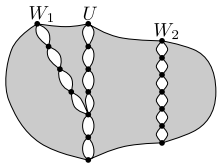

Figure 2 A 7-cut U with two sides, and two 7-cuts W_1, W_2 that are laminar w.r.t. U.

Small Cuts

Informally, when a side of a cut is tiny we call the cut *small*. We define three levels of small cuts. Let U be a cut with sides $A_1, A_2, \ldots, A_a$. We say that

1° U is (I, t)-*small* if there exists an index $i^\sharp$ such that $\sum_{i \neq i^\sharp} |A_i| \leq t$. $A_{i^\sharp}$ is called the *large side* of U and the others the *small sides* of U.

2° U is (II, t)-*small* if there exists $i^\sharp$ such that for every $i \neq i^\sharp$, $|A_i| \leq t$.

3° U is (III, t)-*small*, if there exists $i^\sharp$ such that $|A_{i^\sharp}| \leq t$. In this case $A_{i^\sharp}$ is the *small side* of U.

Note that for any t, I-small cuts are II-small, and II-small cuts are III-small. We typically apply this definition with $t = \kappa$, $t = \Theta(\kappa)$, or $t = \lceil \frac{n-\kappa}{2} \rceil$.

Wheel Cuts

Suppose V can be partitioned into a series of disjoint sets T, $\{C_i\}, \{S_i\}$ ($1 \leq i \leq w, w \geq 4$, subscripts are taken module w), such that the $\{C_i\}$ and $\{S_i\}$ are nonempty (T may be empty), and $C_i \cup T \cup C_{i+2}$ disconnects $S_i \cup C_i \cup S_{i+1}$ from the rest of the graph. We say $(T; C_1, C_2, \ldots, C_w)$ forms a *w-wheel* with *sectors* $S_1, S_2, \ldots, S_w$. We call T the *center* of the wheel, $\{C_i\}$ the *spokes* of the wheel, and $C(i,j) = C_i \cup T \cup C_j$ the *cuts* of the wheel. Define $D(i,j) = S_i \cup C_{i+1} \cup \cdots \cup C_{j-1} \cup S_{j-1}$.

Recall that we are only interested in wheels whose cuts are minimum κ-cuts. The cut of the wheels discussed in this paper are all κ-cuts. It is proved in Lemma 10 that, if $(T; C_1, C_2, \ldots, C_w)$ forms a wheel, then for every i, j such that $j - i \notin \{1, w-1\}$, $C(i,j)$ is a κ-cut with exactly two sides, namely $D(i,j)$ and $D(j,i)$. Note that a w-wheel $(T; C_1, C_2, \ldots, C_w)$ contains x-wheels, $x \in [4, w-1]$. Specifically, for *any* subset $\{i_1, i_2, \ldots, i_x\} \subseteq \{1, 2, \ldots, w\}$ with $x \geq 4$, $(T; C_{i_1}, C_{i_2}, \ldots, C_{i_x})$ forms an x-wheel called a *subwheel* of the original. If a wheel is not a subwheel of any other wheel, it is a *maximal wheel*. If there exists an index $i^\sharp$ such that, $\sum_{i \neq i^\sharp} |S_i| \leq \kappa$, then we say this is a *small wheel*.[9]

[8] These are sometimes called *parellel cuts*.

[9] For a small wheel, all its cuts $C(i,j)$ are (II, $O(\kappa^2)$)-small.

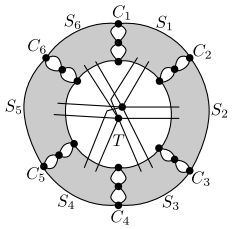

Figure 3 A 6-wheel of 8-cuts with a center of size $|T| = 2$.

Note that wheel cuts resemble the "cycle cuts" of a cactus, or the set of 2−cuts of a cycle.

Matching Cuts and Crossing Matching Cuts

Let U be a cut, A be a side of U, and $P \subseteq U$ be a subset of the cut. We call a cut W a *matching cut of U in side A w.r.t. P* if (i) $U \backslash P \subseteq W \subseteq U \cup A$, (ii) $A \backslash W \neq \emptyset$, and (iii) W disconnects $P \cup (V \backslash (U \cup A))$ from $A \backslash W$. The set $\text{Match}_{U;A}(P) \stackrel{\text{def}}{=} W \backslash U$ is the neighborhood of P restricted to A. Note that a matching cut is a type of laminar cut.

Now suppose U is a cut with exactly two sides A and B, and let $P \subseteq U$ be a non-empty subset of U. We call W a *crossing matching cut of U in side A w.r.t. P* if (i) $W \cap B \neq \emptyset$, (ii) $(U \backslash P) \cup (W \cap A)$ is a matching cut of U in side A w.r.t. P,

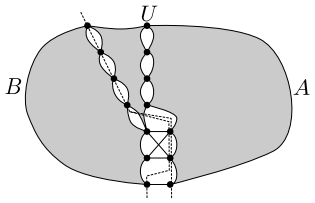

Figure 4 A cut U (drawn vertically) with two sides A and B. Dotted lines indicate two crossing matching cuts w.r.t. P_1 (bottom 3 vertices of U) and P_2 (top 2 vertices of P_1).

One could view U and a crossing matching cut W as a degenerate 4-wheel, in which one sector $S_1 = \emptyset$ is empty. Such cuts should *not* be regarded as wheels, as they do not possess key properties of wheels, e.g., that when U and W are (minimum) κ-cuts, that $|C_1| = \cdots = |C_4| = \frac{\kappa - |T|}{2}$, because $C_1 \cup T \cup C_2$ is not a cut.

Lemmas 3 and 4 are used throughout the paper. Recall here $\kappa = \kappa(G)$ is the vertex connectivity of G.

▶ **Lemma 3.** *Suppose U is a κ-cut and P a side of U. For every $p \in P$ and $u \in U$, there exists a path from p to u that is not blocked by $V \backslash P$.*

▶ **Lemma 4.** *Suppose U and W are two cuts, P is disconnected by U from the rest of the graph G and Q is disconnected by W from the rest of the graph G. Then we have the following two rules:*

- *(Intersection Rule) If $P \cap Q \neq \emptyset$, then $P \cap Q$ is disconnected by $(U \cap Q) \cup (U \cap W) \cup (W \cap P)$ from the rest of the graph G;*
- *(Union Rule) If $V \backslash (U \cup P \cup W \cup Q) \neq \emptyset$, then $P \cup Q$ is disconnected by $(U \backslash Q) \cup (W \backslash P)$ from the rest of the graph G.*

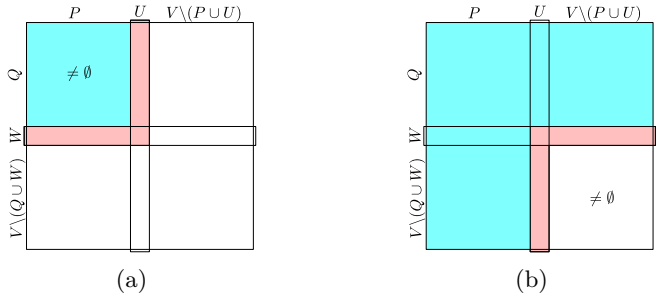

Figure 5 (a) Intersection rule; (b) Union rule.

3 The Classification of Minimum Vertex Cuts

The main *binary* structural theorem for vertex connectivity is, informally, that every two minimum vertex cuts have a relationship that is *Laminar*, *Wheel*, *Crossing Matching*, or *Small*; cf. [11]. Moreover, any strict subset of this list would be inadequate to capture all possible relationships between two vertex cuts.[10]

▶ **Theorem 5.** *Fix a minimum κ-cut U with sides $A_1, A_2, \ldots, A_a$, $a \geq 2$, and let W be any other κ-cut with sides $B_1, B_2, \ldots, B_b$, $b \geq 2$. Denote $T = U \cap W$, $W_i = W \cap A_i$ and $U_j = U \cap B_j$. Then W may be classified w.r.t. U as follows:*

Laminar type. *W is a laminar cut of U, and in particular, there exists indices i^* and j^* such that $B_{j^*} \backslash A_{i^*} = (U \backslash W) \cup (\cup_{i \neq i^*} A_i)$ and $A_{i^*} \backslash B_{j^*} = (W \backslash U) \cup (\cup_{j \neq j^*} B_j)$.*

Wheel type. *$a = b = 2$, and $(T; U_1, W_1, U_2, W_2)$ forms a 4-wheel with sectors $A_1 \cap B_1$, $A_1 \cap B_2$, $A_2 \cap B_2$ and $A_2 \cap B_1$.*

Crossing Matching type. *$a = b = 2$, and w.l.o.g., $A_1 \cap B_1 \neq \emptyset$, $A_2 \cap B_2 \neq \emptyset$, but $A_1 \cap B_2 = \emptyset$. We have $|W_2| = |U_1| > 0$, $|W_1| = |U_2| > 0$, and W is a crossing matching cut of U in side A_1 w.r.t. U_2. Furthermore, if $A_2 \cap B_1 \neq \emptyset$, then $|U_1| \geq |U_2|$.*

Small type. *U is $(I, \kappa - 1)$-small, and the small sides of U are within W, or W is $(I, \kappa - 1)$-small, and the small sides of W are within U.*

▶ Remark 6. Minimum cuts with at least three sides (i.e., $a \geq 3$ or $b \geq 3$) are sometimes called *shredders*. It it known that there are $O(n)$ shredders [41, 47] and that, in the terminology of Theorem 5, two shredders have a *laminar* or *small* relationship.

Proof of Theorem 5. Suppose there is a single index i^* such that $W_{i^*} \neq \emptyset$ and $W_i = \emptyset$ for all $i \neq i^*$. It follows that $W \subseteq A_{i^*} \cup U$ is a laminar cut of U in side A_{i^*}. It remains to prove the other properties of the laminar type. By Lemma 3 there exists paths from any vertex in A_i, $i \neq i^*$, to $U \backslash W$ that are not blocked by W, so they all lie within one side of W; let us denote this side by B_{j^*}. Then $(U \backslash W) \cup (\cup_{i \neq i^*} A_i) \subseteq B_{j^*}$, and because $V = U \cup (\cup_{i=1}^{a} A_i)$, we obtain $B_{j^*} \backslash A_{i^*} = (U \backslash W) \cup (\cup_{i \neq i^*} A_i)$. Now that $U \subseteq W \cup B_{j^*}$ is laminar w.r.t. W, so based on the same reasoning we have $A_{i^*} \backslash B_{j^*} = (W \backslash U) \cup (\cup_{j \neq j^*} B_j)$.

We proceed under the assumption that such indices i^*, j^* do not exist, and without loss of generality assume that $W_1, W_2, U_1, U_2 \neq \emptyset$. We now wish to prove that *all* U_i, W_i are non-empty. Suppose $W_i \stackrel{\text{def}}{=} W \cap A_i = \emptyset$ were empty, then A_i would be contained within a side of W, say $A_i \subseteq B_j$. By Lemma 4 (intersection rule), whenever $A_i \cap B_j \neq \emptyset$, the set $W_i \cup T \cup U_j$ disconnects $A_i \cap B_j$ from the rest of the graph. It follows that

[10] The existence of *Small* cuts as a category – an *a priori* unnatural class – indicates that there may be other ways to capture all minimum vertex cuts through an entirely different classification system.

$$|W_i| + |T| + |U_j| = |T| + |U_j| \geq \kappa = |T| + \sum_{l=1}^{b} |U_l|\,,$$

which implies that U_j is the *only* non-empty U_*-set, contradicting $U_1, U_2 \neq \emptyset$. Therefore, $W_i \neq \emptyset$ for all i and similarly, $U_j \neq \emptyset$ for all j.

Define $\Omega = \{(i,j) \mid A_i \cap B_j \neq \emptyset\}$ to be the side-pairs whose intersections are non-empty. We consider the following possibilities, which are exhaustive.

1° There exist $(i,j),(i',j') \in \Omega$ such that $i \neq i'$, $j \neq j'$. Then by Lemma 4 (intersection rule)

$$\begin{aligned} &|W_i| + |T| + |U_j| \geq \kappa \\ \text{and}\quad &|W_{i'}| + |T| + |U_{j'}| \geq \kappa. \end{aligned}$$

On the other hand,

$$\begin{aligned} &|U_j| + |U_{j'}| + |T| \leq |U| = \kappa \\ \text{and}\quad &|W_i| + |W_{i'}| + |T| \leq |W| = \kappa. \end{aligned}$$

Thus all these inequalities must be equalities, and, adding the fact that all $W_i, U_j \neq \emptyset$, we conclude that $a = b = 2$, $|W_i| = |U_{j'}|$, $|W_{i'}| = |U_j|$. W.l.o.g. we fix $i = j = 1$, $i' = j' = 2$. See Figure 6.

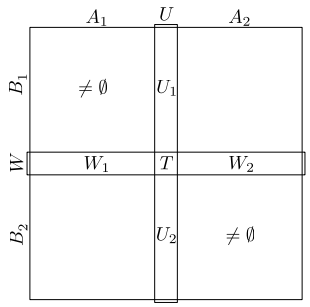

Figure 6 A depiction of cuts U, W in case 1°.

1.1° Suppose $A_1 \cap B_2 \neq \emptyset$ and $A_2 \cap B_1 \neq \emptyset$. Then $|W_i| + |U_j| \geq \kappa - |T|$ for every $\hat{\imath}, \hat{\jmath} \in \{1,2\}$, so we conclude that

$$|U_1| = |U_2| = |W_1| = |W_2| = \frac{\kappa - |T|}{2}.$$

Now that $W_i \cup T \cup U_j$ disconnects $A_i \cap B_j$ from the rest of the graph, $U_1 \cup T \cup U_2 = U$ disconnects $(A_1 \cap B_1) \cup W_1 \cup (A_1 \cap B_2) = A_1$ from $(A_2 \cap B_1) \cup W_2 \cup (A_2 \cap B_2) = A_2$, $W_1 \cup T \cup W_2 = W$ disconnects $(A_1 \cap B_1) \cup U_1 \cup (A_2 \cap B_1) = B_1$ from $(A_1 \cap B_2) \cup U_2 \cup (A_2 \cap B_2) = B_2$, we conclude that $(T; U_1, W_1, U_2, W_2)$ forms a 4-wheel.

1.2° Suppose $A_1 \cap B_2 = \emptyset$ (or symmetrically, that $A_2 \cap B_1 = \emptyset$). Then $A_1 = A_1 \cap (B_1 \cup W) = (A_1 \cap B_1) \cup W_1$. By Lemma 4, $W_1 \cup T \cup U_1$ separates $A_1 \cap B_1$ from the rest of the graph. Since $U_2 \subseteq V \setminus ((A_1 \cap B_1) \cup (U_1 \cup T \cup W_1))$, it follows that $W_1 \cup T \cup U_1$ disconnects U_2 from $A_1 \cap B_1 = A_1 \setminus (W_1 \cup T \cup U_1)$, i.e., it is a matching cut of U in side A_1 w.r.t. U_2. See Figure 7. Because $W_2 = W \cap A_2 \neq \emptyset$ and $(W \cap A_1) \cup (U \setminus U_2) = W_1 \cup T \cup U_1$, W is a crossing matching cut of U in side A_1 w.r.t. U_2.

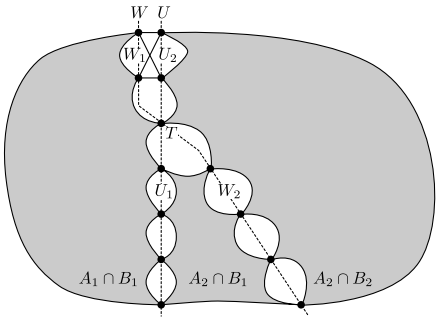

Figure 7 A depiction of the cuts U, W in case 1.2°.

If $A_2 \cap B_1 \neq \emptyset$, by Lemma 4 (intersection rule) $|U_1|+|W_2|+|T| \geq \kappa = |W_1|+|T|+|W_2|$, so $|U_1| \geq |U_2|$.

2° Suppose there exists a $j^\sharp$ such that $\forall i.\forall j \neq j^\sharp.(i,j) \notin \Omega$, i.e., $A_i \cap B_j = \emptyset$. This implies that $\cup_{j \neq j^\sharp} B_j \subseteq U$, and because $U_{j^\sharp} \neq \emptyset$, $\left|\cup_{j \neq j^\sharp} B_j\right|$ is *strictly* smaller than κ. Therefore W is a (I, $\kappa - 1$)-small cut, and all the small sides of W are within U.

3° There exists $i^\sharp$ such that $\forall i \neq i^\sharp.\forall j.(i,j) \notin \Omega$. Symmetric to case 2°; U is (I, $\kappa - 1$)-small, and all the small sides of U are within W.

4° $\Omega = \emptyset$. Then $\cup_{i=1}^{a} A_i \subseteq W$, so $|V| = |U \cup (\cup_{i=1}^{a} A_i)| \leq |U \cup W| \leq 2\kappa$. This is possible, but not one we consider as it contradicts our initial assumption that $n > 4\kappa$. ◀

▶ **Corollary 7** (cf. [41, 47, 53])**.** *If U is a κ-cut that is not (I, $\kappa - 1$)-small and has at least 3 sides, then all other κ-cuts have a laminar type relation with U, or are themselves (I, $\kappa - 1$)-small cuts.*

▶ **Corollary 8.** *Suppose U is a κ-cut that is not (I, $\kappa - 1$)-small, with exactly two sides A and B. Suppose W is a κ-cut with sides K, L (and possibly others), such that $W \cap A \neq \emptyset$, $W \cap B \neq \emptyset$, $A \subseteq K \cup W$, and $L \cap U \neq \emptyset$. Then W only has two sides, and W is a crossing matching cut of U in side A w.r.t. $L \cap U$.*

▶ **Corollary 9.** *Define* $\text{Cuts}_{C;D}$ *to be the set of all κ-cuts that disconnect disjoint, non-empty vertex sets C and D. If* $\text{Cuts}_{C;D} \neq \emptyset$*, it contains a unique minimal element* $\text{MinCut}_{C;D}$*, such that for any cut $U \in \text{Cuts}_{C;D}$,* $\text{Region}_{\text{MinCut}_{C;D}}(C) \subseteq \text{Region}_U(C)$.

Theorem 5 classifies the pairwise relationship between two minimum κ-cuts. In Sections 3.1–3.4 we further explore the properties of wheel cuts, (crossing) matching cuts, laminar cuts, and small cuts.

3.1 Wheels and Wheel Cuts

Recall that a w-wheel $(T; C_1, \ldots, C_w)$ satisfied, by definition, the property that $C_i \cup T \cup C_{i+2}$ formed a κ-cut, but did not say anything explicitly about $C(i,j) = C_i \cup T \cup C_j$. Lemma 10 proves that these are also cuts, and bounds their number of sides.

▶ **Lemma 10.** *Suppose $(T; C_1, C_2, \ldots, C_w)$ forms a w-wheel with sectors $S_1, S_2, \ldots, S_w$. (Subscripts are modulo w.) For any $i \neq j$, $C(i,j)$ is a κ-cut that disconnects $D(i,j)$ from the rest of the graph. Moreover, when $j - i \notin \{1, w-1\}$, $C(i,j)$ has exactly two sides, which are $D(i,j)$ and $D(j,i)$. Furthermore, $|C_i| = \frac{\kappa - |T|}{2}$.*

▶ **Theorem 11.** *Suppose $(T; C_1, C_2, \ldots, C_w)$ forms a w-wheel with sectors $S_1, S_2, \ldots, S_w$. (Subscripts are given by modulo w.) Let X be any minimum κ-cut. Then one of the following is true:*

1° *$X = C(i, j)$ for some $i \neq j$.*

2° *$X \subseteq C(i, i+1) \cup S_i$ for some i, i.e., X is a laminar cut of $C(i, i+1)$.*

3° *X has crossing matching type relation with some $C(i, i+1)$ or some $C(i, i+2)$.*

4° *X is a $(I, \kappa - 1)$-small cut.*

5° *$(T; C_1, C_2, \ldots, C_w)$ is a small wheel.*

6° *There exists $i < j$, such that $X \subseteq S_i \cup T \cup S_j$, and $(T; C_1, \ldots, C_i, X \cap S_i, C_{i+1}, \ldots, C_j, X \cap S_j, C_{j+1}, \ldots, C_w)$ forms a $(w+2)$-wheel; or there exists $i \neq j$, $X \subseteq S_i \cup T \cup C_j$, and $(T; C_1, \ldots, C_i, X \cap S_i, C_{i+1}, \ldots, C_w)$ forms a $(w+1)$-wheel. In other words, the wheel $(T; C_1, C_2, \ldots, C_w)$ is a subwheel of some other wheel.*

3.2 Matching Cuts and Crossing Matching Cuts

Define $N(P)$ to be the neighborhood of $P \subset V$ and $N_A(P) \stackrel{\text{def}}{=} N(P) \cap A$.

▶ **Theorem 12.** *Let U be an arbitrary κ-cut and A a side of U.*

1° *If there exists a matching κ-cut of U in side A w.r.t. P, then it is $W = (U \backslash P) \cup N_A(P)$, and $|P| = |N_A(P)| < |A|$. In particular, $\mathrm{Match}_{U;A}(P) = N_A(P)$.*

2° *When there is such a matching cut W, G contains a matching between P and its neighborhood $N_A(P) = \mathrm{Match}_{U;A}(P)$ in side A.*

3° *Suppose $\mathrm{Match}_{U;A}(P)$ and $\mathrm{Match}_{U;A}(Q)$ exist. If $P \cap Q \neq \emptyset$, then $\mathrm{Match}_{U;A}(P \cap Q)$ exists, and*

$$\mathrm{Match}_{U;A}(P \cap Q) = \mathrm{Match}_{U;A}(P) \cap \mathrm{Match}_{U;A}(Q).$$

If $|A| > |P \cup Q|$, then $\mathrm{Match}_{U;A}(P \cup Q)$ exists, and

$$\mathrm{Match}_{U;A}(P \cup Q) = \mathrm{Match}_{U;A}(P) \cup \mathrm{Match}_{U;A}(Q).$$

Fix a κ-cut U and a side A of U. Define $\Theta = \{P \mid \mathrm{Match}_{U;A}(P) \text{ exists}\}$. According to Part 3° of Theorem 12, Θ is closed under union and intersection, and is therefore characterized by its minimal elements. Define $\Theta^* = \{\cap_{u \in P, P \in \Theta} P \mid u \in U\}$. It can be seen from the definition that $\cap_{u \in P, P \in \Theta} P$ corresponds to the minimum matching cut for vertex u.

In the most extreme case Θ may have $2^\kappa - 1$ elements (e.g., if the graph induced by $U \cup N_A(U)$ is a matching), which may be prohibitive to store explicitly. From definition we know that $|\Theta^*| \leq \kappa$, so it works as a good compression for Θ. Lemmas 13 and 14 also highlights some ways in which Θ^* is a sufficient substitute for Θ.

▶ **Lemma 13.** *Let U be a κ-cut and let Θ be defined w.r.t. the matching cuts of U in a side A. Suppose that $P \in \Theta^*$ and $P \subseteq Q \in \Theta$, and that W is a crossing matching cut of U in side A w.r.t. Q. Then $(W \setminus \mathrm{Match}_{U;A}(Q)) \cup (Q \backslash P) \cup \mathrm{Match}_{U;A}(P)$ is also a crossing matching cut of U in A w.r.t. P. Moreover, if $Q = P_1 \cup P_2 \cup \cdots \cup P_\ell$ where each $P_i \in \Theta^*$, then any pair disconnected by W is also disconnected by some $(W \setminus \mathrm{Match}_{U;A}(Q)) \cup (Q \backslash P_i) \cup \mathrm{Match}_{U;A}(P_i)$.*

▶ **Lemma 14.** *Let U be a κ-cut with two sides A and B, and let Θ be defined w.r.t. its matching cuts in side A. For $P \in \Theta^*$, define $U^*(P)$ to be the cut separating P from $A \setminus \mathrm{Match}_{U;A}(P)$ minimizing $\left|\mathrm{Side}_{U^*(P)}(P)\right|$.*

1° *$U^*(P)$ is either a crossing matching cut of U in side A w.r.t. P, or else there is no such crossing matching cut and $U^*(P) = (U \backslash P) \cup \mathrm{Match}_{U;A}(P)$ is a matching cut.*

2° *Suppose X is a crossing matching cut of U in side A w.r.t. P. If u, v are separated by X, then they are also separated by either $U^*(P)$ or $(X \setminus \mathrm{Match}_{U;A}(P)) \cup P$, which is a laminar cut of U.*

3.3 Laminar Cuts

In this section we analyze the structure of laminar cuts. Throughout this section, U refers to a cut that is *not* (I, $\kappa-1$)-small, *not* a wheel cut $C(i,j)$ in some wheel, and has a side A with $|A| > 2\kappa$.

Consider the set of all cuts W that are laminar w.r.t. U, contained in $U \cup A$ and not (I, $\kappa-1$)-small. It follows that W has a side, call it $S(W)$, that contains $U \setminus W$ and all other sides of U.[11] Define $R(W)$ to be the region containing all other sides of W beside $S(W)$. We call W a *maximal* laminar cut of U if there does not exist another laminar cut W' such that $R(W) \subseteq R(W')$.

▶ **Theorem 15.** *Let U be the reference cut.*

1. *If there exist matching cuts of U in side A, define Θ^* w.r.t. U, A, define $Q = \cup_{P\in\Theta^*} P$, and let $X = (U \setminus Q) \cup \mathrm{Match}_{U;A}(Q)$ be the matching cut in side A having the smallest intersection with U. Then every laminar cut W of U in side A is*
   - **(i)** *a laminar cut of X in region $A \setminus \mathrm{Match}_{U;A}(Q)$, or*
   - **(ii)** *a matching cut of U, or*
   - **(iii)** *a crossing matching cut of X.*
2. *If there are no matching cuts of U in side A, every laminar cut of U in side A is a maximal laminar cut, or a laminar cut of some maximal laminar cut W_i in a side of $R(W_i)$. Moreover, whenever W_i, W_j are distinct maximal laminar cuts, $R(W_i) \cap R(W_j) = \emptyset$.*

3.4 Small Cuts

Fix a vertex u and a threshold $t \le \lceil \frac{n-\kappa}{2} \rceil$. Define $\mathrm{Sm}_t(u)$ to be a cut U minimizing $|\mathrm{Side}_U(u)|$ with $|\mathrm{Side}_U(u)| \le t$. We show that $\mathrm{Sm}_t(u)$, if it exists, is unique. Also, note this is not immediately derived using intersection-union(submodularity) property of cuts, or lemma 4.

▶ **Theorem 16.** *If there exists a (III, t)-small cut that is small w.r.t. u, then there exists a* unique *such cut, denoted $\mathrm{Sm}_t(u)$, such that for any other cut U, $u \notin U$, $\mathrm{Side}_{\mathrm{Sm}_t(u)} \subseteq \mathrm{Side}_U(u)$.*

4 A Data Structure for $(\kappa+1)$-Connectivity Queries

In this section we design an efficient data structure that, given u, v, answers $(\kappa+1)$-connectivity queries, i.e., reports that $\kappa(u,v) = \kappa$ and produces a minimum κ-cut separating u, v, or reports that $\kappa(u,v) \ge \kappa+1$.

We work with the mixed-cut definition of $\kappa(u,v)$ (see Remark 1), which is the minimum size set of vertices and edges that need to be removed to disconnect u and v, or equivalently, the maximum size set of internally vertex-disjoint paths joining u and v.[12]

▶ **Theorem 17.** *Given a κ-connected graph G, we can construct in $\tilde{O}(m + \mathrm{poly}(\kappa)n)$ time a data structure occupying $O(\kappa n)$ space that answers the following queries. Given $u, v \in V(G)$, report whether $\kappa(u,v) = \kappa$ or $\ge \kappa+1$ in $O(1)$ time. If $\kappa(u,v) = \kappa$, report a κ-cut separating u, v in $O(\kappa)$ time.*

---

[11] $S(W)$ is exactly B_{j^*} of Theorem 5, if using its notation on U and W.

[12] If $\{u,v\} \notin E(G)$ and $\kappa(u,v) = \kappa$, then there exists $U \subset V$, $|U| = \kappa$, such that removing U disconnects u, v. If $\{u,v\} \in E(G)$ then there exists $U \subset V$, $|U| = \kappa - 1$, such that removing U and $\{u,v\}$ disconnects u, v. In this case the single-edge path $\{u,v\}$ would count for one of the κ internally vertex disjoint paths, the other $\kappa-1$ passing through distinct vertices of U.

In $O(m)$ time, the Nagamochi-Ibaraki [50] algorithm produces a subgraph G' that has arboricity $\kappa+1$[13] and hence at most $(\kappa+1)n$ edges, such that $\kappa_{G'}(u,v)=\kappa_G(u,v)$ whenever $\kappa_G(u,v)\le\kappa+1$, and $\kappa_{G'}(u,v)\ge\kappa+1$ whenever $\kappa_G(u,v)\ge\kappa+1$. Without loss of generality we may assume G is the *output* of the Nagamochi-Ibaraki algorithm.

Data Structure

Throughout this section we fix the threshold $t=\lceil\frac{n-\kappa}{2}\rceil$. Define $\mathrm{Sm}(u)=\mathrm{Sm}_t(u)$ to be the unique minimum κ-cut with $\mathrm{Side}_{\mathrm{Sm}(u)}(u)\le t$, if any such cut exists, and $\mathrm{Sm}(u)=\perp$ otherwise. The data structure stores, for each $u\in V(G)$, $\mathrm{Sm}(u)$, $|\mathrm{Side}_{\mathrm{Sm}(u)}(u)|$, a $O(\log n)$-bit identifier for $\mathrm{Side}_{\mathrm{Sm}(u)}(u)$, and for each vertex $v\in N(u)\cap\mathrm{Sm}(u)$, a bit $b_{u,v}$ indicating whether $\{\{u,v\}\}\cup\mathrm{Sm}(u)\backslash\{v\}$ is a mixed cut disconnecting u and v. Furthermore, when $|\mathrm{Side}_{\mathrm{Sm}(u)}(u)|\le\kappa-1$, we store $\mathrm{Side}_{\mathrm{Sm}(u)}(u)$ explicitly. When $\mathrm{Sm}(u)=\perp$ we will say $\mathrm{Side}_{\mathrm{Sm}(u)}(u)=G$ and hence $\left|\mathrm{Side}_{\mathrm{Sm}(u)}\right|=n$. The total space is $O(\kappa n)$.

Connectivity Queries

The query algorithm proceeds to the first applicable case. Note in the following, $\mathrm{Sm}(u)$ may be $\perp$, and for all vertices v, we define $v\notin\perp$.

Case I: $\mathrm{Sm}(u)=\mathrm{Sm}(v)$ and $\mathrm{Side}_{\mathrm{Sm}(u)}(u)=\mathrm{Side}_{\mathrm{Sm}(v)}(v)$. Then $\kappa(u,v)\ge\kappa+1$.

Case II: $u\notin\mathrm{Sm}(v)$ and $v\notin\mathrm{Sm}(u)$. Then $\kappa(u,v)=\kappa$. Without loss of generality suppose that $\left|\mathrm{Side}_{\mathrm{Sm}(u)}(u)\right|\le\left|\mathrm{Side}_{\mathrm{Sm}(v)}(v)\right|$. Then $\mathrm{Sm}(u)$ is a κ-cut separating u and v.

Case III: $v\in\mathrm{Sm}(u)\cap N(u)$, or the reverse. The bit $b_{u,v}$ indicates whether $\kappa(u,v)\ge\kappa+1$ or $\kappa(u,v)=\kappa$, in which case $\{\{u,v\}\}\cup\mathrm{Sm}(u)\backslash\{v\}$ is the κ-cut.

Case IV: $v\in\mathrm{Sm}(u),u\in\mathrm{Sm}(v)$. Then $\kappa(u,v)\ge\kappa+1$.

Case V: $v\in\mathrm{Sm}(u),u\notin\mathrm{Sm}(v)$, or the reverse. If $\left|\mathrm{Side}_{\mathrm{Sm}(v)}(v)\right|\le\kappa-1$, directly check whether $u\in\mathrm{Side}_{\mathrm{Sm}(v)}(v)$. If so then $\kappa(u,v)\ge\kappa+1$; if not then $\mathrm{Sm}(v)$ disconnects them. Thus $\left|\mathrm{Side}_{\mathrm{Sm}(v)}(v)\right|\ge\kappa$. If $\left|\mathrm{Side}_{\mathrm{Sm}(v)}(v)\right|\le\left|\mathrm{Side}_{\mathrm{Sm}(u)}(u)\right|$ then $\mathrm{Sm}(v)$ is a κ-cut separating u and v, and otherwise $\kappa(u,v)\ge\kappa+1$.

Lemmas 18, 19, and Theorem 20 establish the *correctness* of the query algorithm. Its construction algorithm is described and analyzed in Section 4.1.

▶ **Lemma 18.** *If $v\in\mathrm{Side}_{\mathrm{Sm}(u)}(u)$, then either $\mathrm{Sm}(v)=\mathrm{Sm}(u)$ or $\mathrm{Sm}(v)$ is a laminar cut of $\mathrm{Sm}(u)$ with $\mathrm{Side}_{\mathrm{Sm}(v)}(v)\subset\mathrm{Side}_{\mathrm{Sm}(u)}(u)$.*

▶ **Lemma 19.** *Suppose u and v are not $(\kappa+1)-$connected, i.e., $\kappa(u,v)=\kappa$. If $\{u,v\}\notin E(G)$, then they are disconnected by $\mathrm{Sm}(u)$ or $\mathrm{Sm}(v)$, and if $\{u,v\}\in E(G)$, then they are disconnected by $\{(u,v)\}\cup\mathrm{Sm}(u)\backslash\{v\}$ or $\{(u,v)\}\cup\mathrm{Sm}(v)\backslash\{u\}$.*

Proof. First suppose $\{u,v\}\notin E(G)$ and let X be any cut separating u and v. When $t=\lceil\frac{n-\kappa}{2}\rceil$ either $|\mathrm{Side}_X(u)|\le t$ or $|\mathrm{Side}_X(v)|\le t$. W.l.o.g. suppose it is the former, then $\mathrm{Sm}(u)$ exists and by Theorem 16, $\mathrm{Side}_{\mathrm{Sm}(u)}(u)\subseteq\mathrm{Side}_X(u)$, so $\mathrm{Sm}(u)$ also separates u and v.

If $\{u,v\}\in E(G)$, suppose $(\kappa-1)$ vertices $W=\{w_1,w_2,\ldots,w_{\kappa-1}\}$ and $\{u,v\}$ disconnect u and v. After removing W from the graph, $G\backslash W$ is still connected. By deleting the edge $\{u,v\}$, the graph breaks into exactly two connected components, say A and B with $u\in A$ and $v\in B$. Then $W\cup\{u\}$ forms a κ-cut with $\mathrm{Side}_{W\cup\{u\}}(v)=B$, and $W\cup\{v\}$

[13] Namely, G' is a union of k forests, as mentioned before.

also forms a κ-cut with $\text{Side}_{W\cup\{v\}}(u) = A$. Clearly we have $n = |W| + |A| + |B| = \kappa - 1 + |A| + |B|$. W.l.o.g. suppose $|A| \leq |B|$, then $|A| \leq \lfloor \frac{n-\kappa+1}{2} \rfloor = \lceil \frac{n-\kappa}{2} \rceil = t$. Thus $\text{Sm}(u)$ exists, $\text{Side}_{\text{Sm}(u)}(u) \subseteq \text{Side}_{W\cup\{v\}}(u)$, and $\text{Sm}(u)$ is either $W \cup \{v\}$ or a laminar cut of $W \cup \{v\}$ in side A. Since $\{u, v\} \in E(G)$, we have $v \in \text{Sm}(u)$. If we remove $\{u, v\}$ from G, then any path from u to v goes through a vertex in W, but any path from u to a vertex in W goes through a vertex in $\text{Sm}(u)\backslash\{v\}$. Therefore, $\{\{u, v\}\} \cup \text{Sm}(u)\backslash\{v\}$ is a mixed cut separating u, v as it blocks all u-v paths. ◀

▶ **Theorem 20.** *The query algorithm correctly answers $(\kappa + 1)$-connectivity queries.*

Proof. Suppose the algorithm terminates in Case I. It follows that $u \notin \text{Sm}(v), v \notin \text{Sm}(u)$, and neither $\text{Sm}(u)$ nor $\text{Sm}(v)$ disconnect u and v. Lemma 19 implies that $\kappa(u, v) \geq \kappa + 1$.

In Case II, if $\text{Sm}(u) \neq \perp$ but $\text{Sm}(v) = \perp$ then $\text{Sm}(u)$ is the cut separating u, v and since $\left|\text{Side}_{\text{Sm}(u)}(u)\right| < \left|\text{Side}_{\text{Sm}(v)}(v)\right| = n$, then the query is answered correctly. If both $\text{Sm}(u), \text{Sm}(v) \neq \perp$, then by Lemma 18, $v \notin \text{Side}_{\text{Sm}(u)}(u)$ and once again the query is answered correctly.

In Case III, by Lemma 19, if u and v are separated by a κ-cut, they are separated by $\{\{u, v\}\} \cup \text{Sm}(u)\backslash\{v\}$ (if $\text{Sm}(u) \neq \perp$) or $\{\{u, v\}\} \cup \text{Sm}(v)\backslash\{u\}$ (if $\text{Sm}(v) \neq \perp$), and this information is stored in the bit $b_{u,v}, b_{v,u}$.

If we get to Case IV then $\{u, v\} \notin E(G)$ and neither $\text{Sm}(u)$ nor $\text{Sm}(v)$ separate u, v, hence by Lemma 19, $\kappa(u, v) \geq \kappa + 1$ and the query is answered correctly.

Case V is the most subtle. Because $v \in \text{Sm}(u)$ and $\{u, v\} \notin E(G)$, Lemma 19 implies that *if* $\kappa(u, v) = \kappa$, then u, v must be separated by $\text{Sm}(v)$. If $\text{Sm}(v) = \perp$ then $\kappa(u, v) \geq \kappa + 1$ and the query is answered correctly. If $|\,\text{Side}_{\text{Sm}(v)}(v)| \leq \kappa - 1$ then the query explicitly answers the query correctly by direct lookup. Thus, we proceed under the assumption that $\text{Sm}(v) \neq \perp$ exists and is not small.

If $u \in \text{Side}_{\text{Sm}(v)}(v)$ then $\text{Sm}(v)$ does not disconnect u and v, and by Lemma 18, $\left|\text{Side}_{\text{Sm}(v)}(v)\right| > \left|\text{Side}_{\text{Sm}(u)}(u)\right|$, so the query is handled correctly in this case.

If $u \notin \text{Side}_{\text{Sm}(v)}(v)$ then $\text{Sm}(v)$ separates u and v, so we must argue that $\left|\text{Side}_{\text{Sm}(v)}(v)\right| \leq \left|\text{Side}_{\text{Sm}(u)}(u)\right|$ for the query algorithm to work correctly. It cannot be that $\text{Sm}(v)$ and $\text{Sm}(u)$ have a laminar relation, so by Theorem 5 they must have a crossing matching, wheel, or small type relation. If they have the small-type relation then the small sides of $\text{Sm}(u)$ are contained in $\text{Sm}(v)$ (contradicting $u \notin \text{Sm}(v)$) or the small sides of $\text{Sm}(v)$ are contained in $\text{Sm}(u)$, but we have already ruled out this case. Thus, the remaining cases to consider are wheel and crossing matching type.

Suppose $\text{Sm}(u), \text{Sm}(v)$ form a 4-wheel $(T; C_1, C_2, C_3, C_4)$. Then $u \notin \text{Sm}(v)$ appears in a sector of the wheel, say S_1. Then $C(1, 2)$ is a cut violating the minimality of $\text{Sm}(u) = C(1, 3)$.

Suppose $\text{Sm}(u), \text{Sm}(v)$ have a crossing matching type relation. Let $A_1 = \text{Side}_{\text{Sm}(u)}(u)$ and A_2 be the other side of $\text{Sm}(u)$, and $B_1 = \text{Side}_{\text{Sm}(v)}(v)$ and B_2 be the other side of $\text{Sm}(v)$. Then $u \in A_1 \cap B_2$, and it must be that the diagonal quadrant $A_2 \cap B_1 = \emptyset$. Suppose otherwise, i.e., $A_2 \cap B_1 \neq \emptyset$, and let $X = (\text{Sm}(u) \cap B_2) \cup (\text{Sm}(v) \cap A_1) \cup (\text{Sm}(u) \cap \text{Sm}(v))$. Then by Corollary 9 X is a κ-cut with $\text{Side}_X(u) = A_1 \cap B_2$, contradicting the minimality of $\text{Sm}(u)$. Thus, $\text{Sm}(v)$ is a crossing matching cut of $\text{Sm}(u)$ in side A_2 w.r.t. some $Q \subseteq \text{Sm}(u) \cap B_1$ with $v \in Q$. By Theorem 5 and $u \in A_1 \cap B_2 \neq \emptyset$, we have

$$|A_1 \cap \text{Sm}(v)| = |B_2 \cap \text{Sm}(u)| \geq |A_2 \cap \text{Sm}(v)| = |B_1 \cap \text{Sm}(u)| = |Q|.$$

Thus,

$$|\,\text{Side}_{\text{Sm}(u)}(u)| = |A_1| > |(A_1 \cap B_1) \cup Q| = |\,\text{Side}_{\text{Sm}(v)}(v)|,$$

establishing the correctness in the crossing matching case. (The strictness of the inequality is because $A_1 \cap B_2 \neq \emptyset$.) ◀

Refer to the full version for a $\tilde{O}(m + \text{poly}(\kappa)n)$-time algorithm to construct this data structure, in particular, to find all minimal cuts $\{\text{Sm}(u)\}_{u \in V}$.

4.1 Construction of the Data Structure

We assume Nagamochi-Ibaraki sparsification [50] has already been applied, so G has arboricity $\kappa + 1$ and $O(\kappa n)$ edges. We use the recent Forster et al. [23] algorithm for computing the connectivity $\kappa = \kappa(G)$ in $\tilde{O}(\text{poly}(\kappa)n)$ time and searching for κ-cuts.

▶ **Corollary 21** (Consequence of Forster, Nanongkai, Yang, Saranurak, and Yingchareonthawornchai [23]). *Given $x \in V(G)$ and an integer $s \leq \lceil \frac{n-\kappa}{2} \rceil$, we can, with high probability $1 - 1/\text{poly}(n)$, compute $\text{Sm}_s(x)$ in $\tilde{O}(|\text{Side}_{\text{Sm}_s(x)}(x)| \cdot \kappa^3)$ time, or determine that $\text{Sm}_s(x)$ does not exist in $\tilde{O}(s\kappa^3)$ time.*

We call the procedure of Corollary 21 FindSmall(x, s).

▶ **Definition 22.** *Let U be a cut, A a side of U. We use the notation $\overline{A} = G \backslash (U \cup A)$ to be the region of all other sides of U. Define $G(U, \overline{A})$ to be the graph induced by $U \cup \overline{A}$, supplemented with a κ-clique on U. If W is a cut in $G(U, \overline{A})$, define $\text{Side}_W^{G(U,\overline{A})}(x)$ to be $\text{Side}_W(x)$ in the graph $G(U, \overline{A})$.*

Lemma 23 is useful for constructing the algorithm in lemma 24.

▶ **Lemma 23.** *Let U be a κ-cut, A be a side of U, and W be a set of κ vertices in $G(U, \overline{A})$. Then W is a κ-cut in $G(U, \overline{A})$ if and only if W is a laminar cut of U in one of the sides of $\overline{A}$. Moreover, when W is such a cut, for any vertex $u \in U \backslash W$,*

$$\text{Side}_W(u) = \text{Side}_W^{G(U,\overline{A})}(u) \cup A.$$

Lemma 24 shows how, beginning with a cut X where $\text{Side}_X(u)$ is small, can find another cut Y (if one exists) where $\text{Side}_Y(u)$ is about M, in $\tilde{O}(M\kappa^4)$ time. The difficulty is that there could be an unbounded number of cuts "between" X and Y that would prevent the FindSmall algorithm from finding Y directly.

▶ **Lemma 24.** *For any integer $M \leq t/2$, vertex u, and cut X with $A = \text{Side}_X(u), |A| \leq 2M$, the algorithm* Expand$(u, A, M)$ *runs in time $\tilde{O}(M\kappa^4)$ and, w.h.p., returns a cut Y satisfying the following properties.*

- $\text{Side}_X(u) \subseteq \text{Side}_Y(u)$.
- $|\text{Side}_Y(u)| \leq 2M$.
- *If there exists a cut Z that is (III, M)-small w.r.t. u, then $|\text{Side}_Y(u)| \geq |\text{Side}_Z(u)|$.*

The content of Expand(u, A, M) *is given below.*

Initially $Y \leftarrow X$. While $|\text{Side}_Y(u)| < M$,

a. *For each vertex $v \in Y$, in parallel,*
  i. *In the graph $G(Y, \overline{\text{Side}_Y(u)})$, run* FindSmall$(v, M)$.

b. *The moment any call to* FindSmall *halts in step (i) with a cut W, stop all such calls and set $Y \leftarrow W$. If all $|Y|$ calls to* FindSmall *run to completion without finding a cut, halt and return Y.*

We use Corollary 25 to find $\text{Sm}_t(u)$ for potentially many vertices u in bulk.

▶ **Corollary 25** (Consequence of Picard and Queyrenne [56]). *Fix two disjoint, non-empty vertex sets C and D. In $O(\kappa^2(n - |C| - |D|))$ time, we can output a cut $S(v)$ for every $v \in V \backslash (C \cup D)$, such that if $\text{Sm}(v)$ exists and $C \subseteq \text{Side}_{\text{Sm}(v)}(v)$, then $S(v) = \text{Sm}(v)$.*

We are now ready to present the entire construction algorithm.

Preamble. The algorithm maintains some κ-cut $T(u)$ for each u, which is initially $\perp$, and stores $|\operatorname{Side}_{T(u)}(u)|$. If the algorithm makes no errors, $T(u) = \operatorname{Sm}_t(u) = \operatorname{Sm}(u)$ at the end of the computation. The procedure $\mathsf{Update}(u, U)$ updates $T(u) \leftarrow U$ if U is a better cut, i.e., $|\operatorname{Side}_U(u)| \leq \min\{|\operatorname{Side}_{T(u)}(u)| - 1, t\}$, and does nothing otherwise. $\mathsf{Update}(A, U)$ is short for $\mathsf{Update}(u, U)$ for all $u \in A$.

Step1: very small cuts. For each $u \in V$, let $U \leftarrow \mathsf{FindSmall}(u, 100\kappa)$ and then $\mathsf{Update}(u, U)$. This takes $\tilde{O}(n\kappa^4)$ time.

Step2: unbalanced cuts. For each index i such that $100\kappa < 2^i \leq t$, let $\alpha = 2^i$, and pick a uniform sample $V_i \subset V$ of size $(n \log n)/\alpha$.[14] For each $u \in V_i$, compute $\operatorname{Sm}_\alpha(u) \leftarrow \mathsf{FindSmall}(u, \alpha)$. If $\operatorname{Sm}_\alpha(u) = \operatorname{Sm}(u) \neq \perp$, we first do an $\mathsf{Update}(\operatorname{Side}_{\operatorname{Sm}(u)}(u), \operatorname{Sm}(u))$, then compute $Y \leftarrow \mathsf{Expand}(u, \operatorname{Sm}(u), \alpha)$. For each $v \in Y$, compute $W_v \leftarrow \mathsf{FindSmall}(w, \alpha)$ and then $\mathsf{Update}(v, W_v)$. We then run the algorithm of Corollary 25 with $C = \operatorname{Side}_{\operatorname{Sm}(u)}(u)$ and $D = V \backslash (Y \cup \operatorname{Side}_Y(u))$, which returns a set of cuts $\{S(v)\}_{v \in V \backslash (C \cup D)}$. For each such $v \in \operatorname{Side}_Y(u) \backslash \operatorname{Side}_{\operatorname{Sm}(u)}(u)$, do an $\mathsf{Update}(v, S(v))$. For each index i, the running time is $\tilde{O}(|V_i| \cdot \alpha\kappa^4) = \tilde{O}(n\kappa^4)$, which is $\tilde{O}(n\kappa^4)$ overall.

Step3: balanced cuts. Sample $O(\log n)$ pairs $(x, y) \in V^2$. For each such pair, compute $U \leftarrow \mathsf{FindSmall}(x, t)$. If $U \neq \perp$, apply the algorithm of Corollary 25 to $C = \operatorname{Side}_{\operatorname{Sm}(x)}(x)$ and $D = \{y\}$, which returns a set $\{S(v)\}$. Then do an $\mathsf{Update}(v, S(v))$ for every $v \in V \backslash (C \cup D)$. By Corollary 21 this takes $\tilde{O}(\kappa^3 n)$ time.

Step4: adjacent vertices At this point it should be the case that $T(u) = \operatorname{Sm}(u)$ for all u. For each $v \in T(u) \cap N(u)$ compute and set the bit $b_{u,v}$. (This information can be extracted from the calls to FindSmall and the algorithm of Corollary 25 in the same time bounds.)

▶ **Lemma 26.** *Suppose* $\operatorname{Sm}(u)$ *and* $\operatorname{Sm}(v)$ *exists,* $u \in \operatorname{Side}_{\operatorname{Sm}(v)}(v)$*, and suppose there is a cut* W *such that* $\kappa \leq \left|\operatorname{Side}_{\operatorname{Sm}(v)}(v)\right| < |\operatorname{Side}_W(u)| \leq t$*. Then* $v \in W \cup \operatorname{Side}_W(u)$*.*

Lemma 26 is critical to proving the correctness of the algorithm's search strategy.

▶ **Theorem 27.** *The construction algorithm correctly computes* $\{\operatorname{Sm}(v)\}_{v \in V}$ *and runs in time* $\tilde{O}(n\kappa^4)$*.*

Proof. If $|\operatorname{Side}_{\operatorname{Sm}(v)}(v)| \leq 100\kappa$, then $T(v) = \operatorname{Sm}(v)$ after Step 1, with high probability.

Suppose that $|\operatorname{Side}_{\operatorname{Sm}(v)}(v)| \in [2^j, 2^{j+1}]$ and $2^{j+1} \leq t$. Then with high probability, at least one vertex $x \in V_j$ is sampled in Step 2 such that $x \in \operatorname{Side}_{\operatorname{Sm}(v)}(v)$. Step 2 (Expand) computes a cut Y such that $|\operatorname{Side}_Y(x)| \geq |\operatorname{Side}_{\operatorname{Sm}(v)}(v)|$, so by Lemma 26, either $v \in \operatorname{Side}_Y(x)$ or $v \in Y$. In the former case $\operatorname{Sm}(v)$ is computed using the Corollary 25 algorithm. In the latter case $\operatorname{Sm}(v)$ is computed directly using FindSmall.

Finally, if $\operatorname{Sm}(v)$ is balanced, say $|\operatorname{Side}_{\operatorname{Sm}(v)}(v)| \geq t/4$, then w.h.p. we would pick a pair (x, y) in Step 3 such that $x \in \operatorname{Side}_{\operatorname{Sm}(v)}(v)$ and $y \in V \backslash (\operatorname{Sm}(v) \cup \operatorname{Side}_{\operatorname{Sm}(v)}(v))$. If this holds the algorithm of Corollary 25 correctly computes $\operatorname{Sm}(v)$. ◀

[14] The Forster et al. [23] algorithm samples vertices proportional to their degree. Note that after the Nagamochi-Ibaraki [50] sparsification, the minimum degree is at least κ and the density of every induced subgraph is at most $\kappa + 1$, so it is equally effective to do vertex sampling.

5 Conclusion

This paper was directly inspired by the extended abstract of Cohen, Di Battista, Kanevsky, and Tamassia [11]. Our goal was to substantiate the main claims of this paper, and to simplify and improve the data structure that answers $(\kappa+1)$-connectivity queries.

We believe that our structural theorems can, ultimately, be used to develop even more versatile vertex-cut data structures. For one example, is it possible to succinctly represent *all* minimum vertex cuts so that the following queries can be answered efficiently?
Is-it-a-cut?$(u_1,\ldots,u_\kappa)$**:** Return *true* iff $\{u_1,\ldots,u_\kappa\}$ forms a κ-cut.
List-cuts(u)**:** Return all the κ-cuts containing u.
Note that if all minimum cuts have a wheel or laminar relationship, then they can be represented as a tree, as in [4, 11]. Whether there is a clean representation that *also* captures all *small* and *crossing matching* cuts is an open problem.

We assumed throughout the paper that κ was not too large, specifically $\kappa < n/4$. When $n < 2\kappa$, *all* cuts are (I,κ)-small by our classification, and the classification theorem (Theorem 5) says very little about the structure of such cuts. Understanding the structure of minimum vertex cuts when κ is large, relative to n, is an interesting open problem.

References

1 Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. Cut-equivalent trees are optimal for min-cut queries. In *Proceedings of the 61st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 105–118, 2020. doi:10.1109/FOCS46700.2020.00019.

2 S. Baswana, K. Choudhary, and L. Roditty. Fault tolerant subgraph for single source reachability: generic and optimal. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*, pages 509–518, 2016. doi:10.1145/2897518.2897648.

3 Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-Ramanujan sparsifiers. *SIAM J. Comput.*, 41(6):1704–1721, 2012. doi:10.1137/090772873.

4 G. Di Battista and R. Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, 15:302–318, 1996.

5 Giuseppe Di Battista and Roberto Tamassia. Incremental planarity testing (extended abstract). In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 436–441, 1989. doi:10.1109/SFCS.1989.63515.

6 A. A. Benczúr. Counterexamples for directed and node capacitated cut-trees. *SIAM J. Comput.*, 24(3):505–510, 1995.

7 A. A. Benczúr and M. X. Goemans. Deformable polygon representation and near-mincuts. In M. Grötschel and G. O. H. Katona, editors, *Building Bridges: Between Mathematics and Computer Science*, volume 19 of *Bolyai Society Mathematical Studies*, pages 103–135. Springer, 2008.

8 András A. Benczúr and David R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *SIAM J. Comput.*, 44(2):290–319, 2015. doi:10.1137/070705970.

9 Chung-Kuan Cheng and T. C. Hu. Ancestor tree for arbitrary multi-terminal cut functions. *Ann. Oper. Res.*, 33(3):199–213, 1991. doi:10.1007/BF02115755.

10 K. Choudhary. An optimal dual fault tolerant reachability oracle. In *Proceedings 43rd Int'l Colloq. on Automata, Languages, and Programming (ICALP)*, 2016.

11 Robert F Cohen, Giuseppe Di Battista, Arkady Kanevsky, and Roberto Tamassia. Reinventing the wheel: an optimal data structure for connectivity queries (extended abstract). In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC)*, pages 194–200, 1993.

12 William H. Cunningham and Jack Edmonds. A combinatorial decomposition theory. *Canadian J. Math.*, 32(3):734–765, 1980. doi:10.4153/CJM-1980-057-7.

13 E. A. Dinic, A. V. Karzanov, and M. V. Lomonosov. On the structure of the system of minimum edge cuts in a graph. *Studies in Discrete Optimization*, pages 290–306, 1976. (in Russian).

14 Y. Dinitz and Z. Nutov. A 2-level cactus model for the system of minimum and minimum+1 edge-cuts in a graph and its incremental maintenance. In *Proceedings 27th ACM Symposium on Theory of Computing (STOC)*, pages 509–518, 1995.

15 Y. Dinitz and Z. Nutov. A 2-level cactus tree model for the system of minimum and minimum+1 edge cuts of a graph and its incremental maintenance. Part I: the odd case. Unpublished manuscript, 1999.

16 Y. Dinitz and Z. Nutov. A 2-level cactus tree model for the system of minimum and minimum+1 edge cuts of a graph and its incremental maintenance. Part II: the even case. Unpublished manuscript, 1999.

17 Yefim Dinitz and Alek Vainshtein. The connectivity carcass of a vertex subset in a graph and its incremental maintenance. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 716–725, 1994. doi:10.1145/195058.195442.

18 Yefim Dinitz and Alek Vainshtein. Locally orientable graphs, cell structures, and a new algorithm for the incremental maintenance of connectivity carcasses. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 302–311, 1995. URL: http://dl.acm.org/citation.cfm?id=313651.313711.

19 Yefim Dinitz and Alek Vainshtein. The general structure of edge-connectivity of a vertex subset in a graph and its incremental maintenance. odd case. *SIAM J. Comput.*, 30(3):753–808, 2000. doi:10.1137/S0097539797330045.

20 R. Duan and S. Pettie. Connectivity oracles for failure prone graphs. In *Proceedings 42nd ACM Symposium on Theory of Computing*, pages 465–474, 2010.

21 R. Duan and S. Pettie. Connectivity oracles in graphs subject to vertex failures. *SIAM J. Comput.*, 49(6):1363–1396, 2020.

22 Donatella Firmani, Loukas Georgiadis, Giuseppe F. Italiano, Luigi Laura, and Federico Santaroni. Strong articulation points and strong bridges in large scale graphs. *Algorithmica*, 74(3):1123–1147, 2016. doi:10.1007/s00453-015-9991-z.

23 Sebastian Forster, Danupon Nanongkai, Liu Yang, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Computing and testing small connectivity in near-linear time and queries via fast local cut algorithms. In *Proceedings of the 31st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2046–2065. SIAM, 2020. doi:10.1137/1.9781611975994.126.

24 András Frank and Tibor Jordán. Minimal edge-coverings of pairs of sets. *J. Comb. Theory, Ser. B*, 65(1):73–110, 1995. doi:10.1006/jctb.1995.1044.

25 Harold N Gabow. A representation for crossing set families with applications to submodular flow problems. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 202–211, 1993.

26 Harold N. Gabow. Using expander graphs to find vertex connectivity. *J. ACM*, 53(5):800–844, 2006. doi:10.1145/1183907.1183912.

27 Yu Gao, Jason Li, Danupon Nanongkai, Richard Peng, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Deterministic graph cuts in subquadratic time: Sparse, balanced, and k-vertex. *CoRR*, abs/1910.07950, 2019. arXiv:1910.07950.

28 Loukas Georgiadis, Giuseppe F. Italiano, Luigi Laura, and Nikos Parotsidis. 2-edge connectivity in directed graphs. *ACM Trans. Algorithms*, 13(1):9:1–9:24, 2016. doi:10.1145/2968448.

29 Loukas Georgiadis, Giuseppe F. Italiano, Luigi Laura, and Nikos Parotsidis. 2-vertex connectivity in directed graphs. *Inf. Comput.*, 261:248–264, 2018. doi:10.1016/j.ic.2018.02.007.

30 Loukas Georgiadis, Giuseppe F. Italiano, and Nikos Parotsidis. Strong connectivity in directed graphs under failures, with applications. *SIAM J. Comput.*, 49(5):865–926, 2020. doi:10.1137/19M1258530.

31 R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9, 1961.

32 Frieda Granot and Refael Hassin. Multi-terminal maximum flows in node-capacitated networks. *Discrete applied mathematics*, 13(2-3):157–163, 1986.

33 D. Gusfield and D. Naor. Efficient algorithms for generalized cut trees. In *Proceedings First ACM-SIAM Symposium on Discrete Algorithms*, pages 422–433, 1990.

34 Dan Gusfield and Dalit Naor. Extracting maximal information about sets of minimum cuts. *Algorithmica*, 10(1):64–89, 1993.

35 Refael Hassin and Asaf Levin. Flow trees for vertex-capacitated networks. *Discrete applied mathematics*, 155(4):572–578, 2007.

36 J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2(3):135–158, 1973.

37 Tai-Hsin Hsu and Hsueh-I Lu. An optimal labeling for node connectivity. In *Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC)*, pages 303–310. Springer, 2009.

38 Rani Izsak and Zeev Nutov. A note on labeling schemes for graph connectivity. *Information processing letters*, 112(1-2):39–43, 2012.

39 Bill Jackson and Tibor Jordán. Independence free graphs and vertex connectivity augmentation. *Journal of Combinatorial Theory, Series B*, 94(1):31–77, 2005.

40 Tibor Jordán. On the optimal vertex-connectivity augmentation. *Journal of Combinatorial Theory, Series B*, 63(1):8–20, 1995.

41 Tibor Jordán. On the number of shredders. *Journal of Graph Theory*, 31(3):195–200, 1999.

42 A. Kanevsky, R. Tamassia, G. Di Battista, and J. Chen. On-line maintenance of the four-connected components of a graph. In *Proceedings 32nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 793–801, 1991.

43 B. M. Kapron, V. King, and B. Mountjoy. Dynamic graph connectivity in polylogarithmic worst case time. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1131–1142, 2013.

44 Michal Katz, Nir A Katz, Amos Korman, and David Peleg. Labeling schemes for flow and connectivity. *SIAM J. Comput.*, 34(1):23–40, 2004.

45 A. Korman. Labeling schemes for vertex connectivity. *ACM Trans. on Algorithms*, 6(2), 2010.

46 Jason Li, Danupon Nanongkai, Debmalya Panigrahi, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Vertex connectivity in poly-logarithmic max-flows. In *Proceedings of the 53rd ACM Symposium on Theory of Computing (STOC)*, 2021.

47 Gilad Liberman and Zeev Nutov. On shredders and vertex connectivity augmentation. *Journal of Discrete Algorithms*, 5(1):91–101, 2007.

48 Saunders Mac Lane. A structural characterization of planar combinatorial graphs. *Duke Math. J.*, 3(3):460–472, 1937. doi:10.1215/S0012-7094-37-00336-3.

49 Karl Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10:96–115, 1927.

50 H. Nagamochi and T. Ibaraki. A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, 7(5&6):583–596, 1992.

51 Zeev Nutov. Approximating connectivity augmentation problems. *ACM Trans. Algorithms*, 6(1):5:1–5:19, 2009. doi:10.1145/1644015.1644020.

52 Zeev Nutov. Improved approximation algorithms for minimum cost node-connectivity augmentation problems. *Theory Comput. Syst.*, 62(3):510–532, 2018. doi:10.1007/s00224-017-9786-5.

53 Zeev Nutov and Masao Tsugaki. On (t, k)-shredders in k-connected graphs. *Ars Comb.*, 83, 2007.

54 M. Pătraşcu and M. Thorup. Planning for fast connectivity updates. In *Proceedings 48th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 263–271, 2007.

55 Seth Pettie and Longhui Yin. The structure of minimum vertex cuts. *CoRR*, abs/2102.06805, 2021. arXiv:2102.06805.

56 J.-C. Picard and M. Queyranne. On the structure of all minimum cuts in a network and applications. In *Combinatorial Optimization II*, volume 13 of *Mathematical Programming Studies*, pages 8–16. Springer, 1980.

57 C.-P. Schnorr. Bottlenecks and edge connectivity in unsymmetrical networks. *SIAM J. Comput.*, 8(2):265–274, 1979.

58 R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

59 W. T. Tutte. A theory of 3-connected graphs. *Nederl. Akad. Wetensch. Proc. Ser. A 64 = Indag. Math.*, 23:441–455, 1961.

60 W. T. Tutte. *Connectivity in Graphs*. University of Toronto Press, 1966.

61 H. Whitney. Congruent graphs and the connectivity of graphs. *American J. Mathematics*, 54(1):150–168, 1932. doi:10.2307/2371086.

62 Hassler Whitney. Non-separable and planar graphs. *Trans. Amer. Math. Soc.*, 34(2):339–362, 1932. doi:10.2307/1989545.