Isolating Cuts, (Bi-)Submodularity, and Faster Algorithms for Connectivity

Chandra Chekuri
University of Illinois at Urbana-Champaign, IL, USA

Kent Quanrud
Purdue University, West Lafayette, IN, USA

Abstract

Li and Panigrahi [37], in recent work, obtained the first deterministic algorithm for the global minimum cut of a weighted undirected graph that runs in time $o(mn)$. They introduced an elegant and powerful technique to find *isolating cuts* for a terminal set in a graph via a small number of s-t minimum cut computations.

In this paper we generalize their isolating cut approach to the abstract setting of symmetric bisubmodular functions (which also capture symmetric submodular functions). Our generalization to bisubmodularity is motivated by applications to element connectivity and vertex connectivity. Utilizing the general framework and other ideas we obtain significantly faster randomized algorithms for computing global (and subset) connectivity in a number of settings including hypergraphs, element connectivity and vertex connectivity in graphs, and for symmetric submodular functions.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases cuts, vertex connectivity, hypergraphs, fast algorithms, submodularity, bisumodularity, lattices, isolating cuts, element connectivity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.50

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2103.12908` [10]

Funding *Chandra Chekuri*: Supported in part by NSF grants CCF-1910149 and CCF-1907937.

Acknowledgements We thank the reviewers for their helpful comments.

1 Introduction

We investigate fast algorithms for several fundamental connectivity problems in (weighted) undirected graphs as well as their generalizations to the abstract setting of submodular and bisubmodular functions. The motivation for this work arose from the recent paper of [37] that described a new algorithmic approach for finding the *global minimum cut* in an undirected graph. For a graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}_{>0}$, the global minimum cut problem is to find the minimum weight subset of edges whose removal disconnects the graph; alternatively it is to find a set S, where $\emptyset \subsetneq S \subsetneq V$, that minimizes $w(\delta(S))$[1]. When G is unweighted, this is called the edge connectivity of the graph. There has been extensive work on algorithms for this problem, and its study has led to many important theoretical developments. Karger developed a near-linear time randomized algorithm [30] that runs in $O(m \log^3 n)$ time with some recent improvements in the log factors via better data structures [21, 42]. Here m is the number of edges and n is number of nodes in the

[1] For $A \subset V$, $\delta(A)$ denote the set of edges in G with exactly one end point in A. $w(\delta(A))$ is notation for $\sum_{e \in \delta(A)} w(e)$.

48th International Colloquium on Automata, Languages, and Programming (ICALP 2021).
Editors: Nikhil Bansal, Emanuela Merelli, and James Worrell; Article No. 50; pp. 50:1–50:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

graph. However, the best deterministic algorithm until recently was $\tilde{O}(mn)$ via two very different approaches [24, 49]. Li and Panigrahi developed a new approach that improved this bound. Their algorithm runs in time $O(m^{1+o(1)})$ plus the time to compute $O(\text{polylog}(n))$ (s,t)-minimum cut computations in a graph with m edges and n nodes. Their approach uses the (s,t)-minimum cut algorithm as a black box.

Isolating cuts. A key technique in [37] is an algorithm to find *isolating cuts*. To describe this notion, let $R \subseteq V$ be subset of nodes that we call terminals. Given $r \in R$, a set $S \subseteq V$ is an isolating cut for r (with respect to R) if $S \cap R = \{r\}$. Consider the problem of finding, for *each* $r \in R$, a minimum weight isolating cut, that is; a cut $S_r \subseteq V$ where $S_r = \arg\min_{S \subseteq V, S \cap R = \{v\}} w(\delta(S))$. Note that if $R = V$ this is trivial since $S_r = \{r\}$ for each r. However, the problem is non-trivial when $R \subset V$ is a proper subset of V. A naive approach would require $|R|$ (s,t)-minimum cut computations. [37] described a simple and elegant procedure that computes all the isolating cuts for any given R in time proportional to $O(\log |R|)$ (s,t)-minimum cut computations. This, combined with simple random sampling, can be used to easily derive a randomized algorithm for global minimum cut that relies on $O(\text{polylog}(n))$ (s,t)-minimum cut computations. Note that even though the total time corresponds to $O(\text{polylog}(n))$ (s,t)-minimum cuts, the second phase of their algorithm requires computing $|R|$ (s,t)-minimum cuts, but in smaller graphs whose total size is $O(m)$ and thus can be folded into a single (s,t)-minimum cut on roughly the same input size as the original graph. Their algorithm gives a new randomized approach to global minimum cut; however, it does not lead to a faster algorithm than the existing near-linear time algorithm. Instead [37] focuses on deterministic running times and avoids random sampling by relying on several technical tools including deterministic expander decompositions to obtain a deterministic algorithm. We note, however, that the algorithm in [37] applies to the more general problem of finding the Steiner minimum cut: given $X \subseteq V$, the goal is to find a minimum cut separating a pair of nodes in X. See [25, 29] for applications.

Vertex and element connectivity. Our focus here is not on deterministic algorithms per se but rather on the applicability of the isolating cut approach to derive faster (randomized) algorithms in settings beyond edge connectivity. There has been tremendous recent and ongoing progress in fast algorithms for (s,t)-flow and cut problems and leveraging these algorithms for global connectivity is opened up by the new approach. In particular, an important motivating problem is to compute the global *(weighted) vertex connectivity* of a graph which has received substantial recent attention [17, 45]. In this setting we are given a graph $G = (V, E)$ with vertex weights $w : V \to \mathbb{R}_+$ and the goal is to find a minimum weight subset $S \subset V$ such that $G - S$ has at least two non-trivial connected components. However, as is well-known, vertex cuts/separators are not as easy to work with as edge cuts. Despite recent exciting progress via an approach based on local cuts and connectivity, the weighted case had not been addressed and the best known algorithms are from the work of Henzinger, Rao and Gabow [27]. Our starting point is the observation that the isolating cut approach of [37] relies only on the submodularity and symmetry of the edge-cut function of undirected graphs. Recall that a real-valued set function $f : 2^V \to \mathbb{R}$ is *submodular* iff $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$ for all $A, B \subseteq V$. A set function is symmetric if $f(A) = f(V \setminus A)$ for all $A \subseteq V$. The applicability of the isolation cut approach to symmetric submodular set functions already yields faster algorithms for hypergraph connectivity and several other problems that we describe subsequently. However, as we already remarked, vertex cuts do not lend themselves to this approach as vertex cuts, unlike undirected edge cuts, are simply not a symmetric submodular function.

When considering isolating cuts in the context of vertex connectivity one naturally encounters the notion of *element connectivity*, which has been found to have several important connections between edge and vertex connectivity. Element connectivity plays a key role in network design, and in fact, it was introduced by [28] to overcome the difficulty of working with vertex connectivity. We refer the reader to surveys and related papers on network design [16, 14, 15, 23, 33] for extensive literature on this topic. It also plays an important role in packing vertex disjoint Steiner trees and forests among others [13, 5, 1, 8]; [7] surveys this area. We now formally define element connectivity. The input is a graph $G = (V, E)$ and a partition of V into terminals T and non-terminals $N = V \setminus T$. The *elements* of G are the edges and non-terminals; that is, $E \cup N$. For two terminals s, t we define the element connectivity between s and t as the minimum number of elements whose removal disconnects s from t. We emphasize that element connectivity is defined *only* between the terminals. We can generalize this to the weighted setting where edges and non-terminals have non-negative weights. The global element connectivity of $G = (T \cup N, E)$ is the minimum element connectivity between any two terminals. [11] considered algorithms for computing (global) element connectivity. For global element connectivity they obtained an algorithm with running time $O(|T|)$ times the time for (s,t)-minimum cut computation.

Set-pairs and Bisubmodularity. Cuts for element and vertex connectivity do not fall into the setting of symmetric submodular set functions. A vertex separator S induces a partition of $V \setminus S$ into disjoint sets A, B that do not share an edge, and obviously $B \neq V \setminus A$ (for nonempty S). Nevertheless, one of the reasons for the tractability of element connectivity is that it does admit submodularity properties. The natural way to view its submodularity properties is via the more general notion of *bisubmodular* set functions. Given a ground set V a set-pair is (A, B) where $A, B \subseteq V$. Informally speaking a bisubmodular function f assigns a real-value to each set-pair (A, B) in a collection of set-pairs as to satisfy the inequality

$$f(X_1, Y_1) + f(X_2, Y_2) \geq f(X_1 \cup X_2, Y_1 \cap Y_2) + f(X_1 \cap X_2, Y_1 \cup Y_2)$$

for all set-pairs (X_1, Y_1) and (X_2, Y_2) on which it is defined. For this to make sense the collection of set-pairs needs to be closed under the above criss-crossed intersection and union operations for set-pairs. These binary operations can be understood more clearly as the meet and join of an appropriately defined lattice; we defer the formal definitions to Section 2. One can generalize the notion of cuts to set-pairs. Let (S, T) be a set-pair corresponding to a partition of a terminal set R. A set-pair (A, B) cuts (S, T) if $S \subseteq A$ and $T \subseteq B$. One can then define the f-minimum cut problem for (S, T): find the set-pair of minimum f value among all set-pairs that cut (S, T). With this definition in place the notions of global minimum cut for a terminal set $R \subseteq V$, and isolating cuts for R, naturally generalize. In this paper we show that the isolating cut approach of [37] generalizes to the class of symmetric bisubmodular set functions defined over appropriate collections of set-pairs.

1.1 Contributions and Results

We make two contributions at the high-level. The first is conceptual in generalizing the isolating cut approach to the (bi)submodular setting. The second is to apply this abstract framework with additional ideas to derive faster randomized algorithms for several fundamental problems. Together they yield a plethora of new running times for a diverse collection of connectivity problems, both abstract (optimizing over set functions in an oracle model) and concretely in graphs. The multiplicity of results is for the following combination of reasons. First, by implementing the isolating cut approach at a higher level of abstraction, and abstaining from concrete specificities, we not only expose the isolating cut approach to new problems, but allow for the substitution of different domain specific black box subroutines that, within a domain, can have interesting tradeoffs. Second, and unlike the case of graph edge connectivity, the second phase of the isolating cut approach can often benefit from additional problem specific ideas, especially if one wants to take advantage of certain domain-specific algorithms that can be very powerful if applied carefully.

An important aspect of the isolating cut approach is that it inherently gives an algorithm for the subset connectivity version. In the following we will use m, n to refer to the number of edges and vertices in a given graph and use $\mathrm{EC}(m, n)$ to refer to the running time for computing a minimum (s, t)-cut in an edge-weighted directed graph, and $\mathrm{VC}(m, n)$ for the running time for computing a minimum (s, t)-cut in a vertex-weighted directed graph. We instantiate concrete running times for special cases when needed.

Connectivity of Bisubmodular functions. The precise statement that captures the general isolation cut property in bisubmodular set functions requires stating several technical definitions. Our main results for this are captured by Lemma 11, Lemma 12 and Theorem 13 which are better understood after the technical definitions. Here we state an informal theorem that captures these results.

▶ **Theorem 1** (Informal). *Let $f : \mathcal{V} \to \mathbb{R}$ be a symmetric bisubmodular function defined over a collection of set-pairs $\mathcal{V}$ over V. Let $R \subseteq V$. Suppose one has an oracle that given a partition (S, T) of R finds the f-minimum set-pair $(A, B) \in \mathcal{V}$ that cuts (S, T). In $O(\log |R|)$ calls to this oracle one can find for each $r \in R$ a set-pair $(X_r, X'_r) \in \mathcal{V}$ such that the following properties hold: (i) for each $r \in R$, (X_r, X'_r) is a $(r, R - r)$ separating set-pair, (ii) there is an f-minimum set-pair (Y_r, Y'_r) separating $(r, R - r)$ such that $Y_r \subseteq X_r$, $X'_r \subseteq Y'_r$ and (iii) $X_r \cap X_q = \emptyset$ for $r \neq q$. The total run time for finding the f-minimum isolating cut (Y_r, Y'_r) for each $r \in R$ can thus be bounded by the $O(\log |R|)$ cut computations and the total time to find the cuts inside each (X_r, X'_r).*

Symmetric submodular functions. We derive the following theorem as a corollary.

▶ **Theorem 2.** *Let $f : 2^V \to \mathbb{R}$ be a symmetric submodular function and $R \subseteq V$ and let $n = |V|$. Suppose there is an algorithm for submodular function minimization in the value oracle model in time $\mathrm{SFM}(n) = g_1(n)EO + g_2(n)$ where EO is the time for the evaluation oracle. Assuming that $g_1(n) = \Omega(n)$ and $g_2(n) = \Omega(n)$, a minimum f-cut that separates some two terminals in R can be found in $O\big(\mathrm{SFM}(n) \log^2(n)\big)$ time.*

▶ **Corollary 3.** *Let f be an integer valued symmetric submodular function with $|f(S)| \leq M$. Using the submodular function minimization algorithms of [35] one can find the global minimum cut of f with high probability in time $\tilde{O}(n^2 \log(nM)EO + n^3 \log^{O(1)}(nM))$.*

The preceding corollary should be compared to Queyranne's well-known combinatorial algorithm that uses $O(n^3\mathrm{EO})$ time [47]. The algorithm from [35] is not strongly polynomial but uses a factor $\tilde{\Omega}(n)$ fewer evaluation calls. Further, our randomized algorithm can handle minimum f-cut for a subset of terminals while Queyranne's algorithm does not generalize. In addition, the black box reduction can take advantage of future improvements to $\mathrm{SFM}(n)$ as well as for special cases as we will see next.

Hypergraph connectivity. A hypergraph $H = (V, E)$ consists of vertices V and hyperedges E where each hyperedge $e \in E$ is a subset of nodes; that is, $e \subseteq V$. We let $p = \sum_{e \in E} |e|$ denote the total size of H and let m, n denote number of hyperedges and vertices. The rank r of a hypergraph is the maximum edge size; graphs are rank 2 hypergraphs. The cut function of a hypergraph is symmetric and submodular and the global minimum cut question for edge connectivity naturally generalizes to hypergraphs. The best deterministic algorithm for this problem runs in $O(pn + n^2 \log n)$ time [32, 47, 41]. The best randomized algorithm runs in time $\tilde{O}(n^r)$ time with high probability in rank r hypergraphs [18] and this is better than $\tilde{O}(pn)$ only for very dense hypergraphs. Via sparsification one can also get an algorithm in unweighted hypergraphs that runs in time $O(p + \lambda n^2)$ where λ is the minimum cut value [12]. We obtain the following theorem that gives significantly better bounds in most settings of interest, and new tradeoffs, while also generalizing to subset minimum cut.

▶ **Theorem 4.** *Let $H = (V, E)$ be a weighted hypergraph with m edges, n nodes and total size $p = \sum_{e \in E} |e|$. Let $R \subseteq V$. The global minimum cut for R in H can be found with high probability in time $\tilde{O}(\mathrm{EC}(p, m + n))$ or in time $\tilde{O}(\sqrt{pn(m + n)^{1.5}})$.*

Now we state our algorithmic results for element connectivity and vertex connectivity that follow via the bisubmodularity framework and problem specific optimizations.

Element connectivity. The fastest known algorithm so far for global element connectivity is from [11] and runs in time $O(|T| \,\mathrm{EC}(m, n))$ for terminal set T, which can be $\Omega(n \,\mathrm{EC}(m, n))$. We obtain the following.

▶ **Theorem 5.** *Let $G = (T \cup N, E)$ be an instance of weighted element connectivity with $|T| = k$ terminals. The global element connectivity can be computed in*

$$\tilde{O}\left(\mathrm{EC}(m, n) + \max_{m_1, \ldots, m_k} \left\{ \sum_{i=1}^{k} \mathrm{EC}(m_i, n) : m_1 + \cdots + m_k \leq 2m \right\}\right)$$

time with high probability. The algorithm generalizes to subset element connectivity.

In particular, for $\mathrm{EC}(m, n)$ of the form $\mathrm{EC}(m, n) = \tilde{O}(m \,\mathrm{poly}(m, n))$, the running time above is $\tilde{O}(\mathrm{EC}(m, n))$. For instance, via [34], one obtains an $\tilde{O}(m\sqrt{n})$ time algorithm. However, recent breakthrough work of [51] showed that $\mathrm{EC}(m, n) = \tilde{O}(m + n^{1.5})$. This running time bound cannot be directly used in the preceding theorem. Using further ideas we obtain an improved running times that are encapsualted in the following theorem.

▶ **Theorem 6.** *Let $G = (T \cup N, E)$ be an instance of element connectivity with n nodes and m edges. Let $w : V \cup E \to [1..U]$ assign integer (or infinite) weights to each vertex and edge. The global element connectivity can be computed in randomized $\tilde{O}\big(m^{1+o(1)} n^{3/8} U^{1/4} + n^{1.5}\big)$ time or in $\tilde{O}\big(m^{1/2} n^{5/4}\big)$ time where $\tilde{O}(\cdots)$ hides $\mathrm{poly}(\log(n), \log(U))$-factors.*

Vertex connectivity. We now consider global vertex connectivity of both weighted and unweighted graphs. For simplicity we consider the interesting setting where there is a vertex separator of size less than $0.99W$ where W is the total vertex weight. We obtain new and faster randomized $(1 + \epsilon)$-approximation algorithms that improves upon the randomized $\tilde{O}(mn)$ exact algorithm of Henzinger, Rao and Gabow [27]. The algorithms are based on reducing, via sampling, to computing isolating element cuts. The running times we obtain are captured by the following theorem.

▶ **Theorem 7.** *Let $G = (V, E)$ be a* weighted *instance of vertex connectivity. There is a randomized algorithm that gives a $(1+\epsilon)$-approximation with high-probability in $\tilde{O}(\mathrm{EC}(m, n)/\epsilon)$ time; in particular there is a randomized algorithm that runs in time $\tilde{O}(m\sqrt{n}/\epsilon)$. For dense graphs there is a randomized algorithm that runs in $\tilde{O}(m^{1/2}n^{5/4}/\epsilon)$ time.*

There has been exciting recent work on faster algorithms for vertex connectivity via a local connectivity approach [45, 17]. The algorithms are limited to unweighted graphs while our theorem above gives the first constant factor approximation for weighted vertex connectivity in $o(mn)$ time. For unweighted graphs and graphs with small integer capacities we can obtain *exact* algorithms by setting $\epsilon = 1/\kappa$ where κ is the vertex connectivity. We obtain several different tradeoffs depending on m, n, κ. These can be found in Section 4.

Recent related work. There have been several recent papers on the use of isolating cuts and other approaches for connectivity problems in undirected as well as directed graphs. We refer the reader to some of these papers [36, 9, 46, 6]. Others have also observed that the isolating cut approach generalizes to symmetric submodular functions; see [43] for instance.

Organization. Section 2 describes the bisubmodularity framework and the abstract results at a high level; a more detailed description with several examples and formal proofs of the lemmas and theorems stated in Section 2 can be found in the full version [10]. Section 3 describes the algorithms for element-connectivity and Section 4 describes our algorithms for vertex connectivity. Section 5 describes the results for hypergraph connectivity. The proofs for Section 5 have been omitted due to space constraints and can be found in the full version [10].

2 Isolating Cuts, Symmetric Bisubmodular Functions, and Lattices

Our goal in this section to define the relevant machinery to explore and make explicit the generality of the isolating cut idea. As discussed in the introduction, this framework is motivated by the necessity of going beyond symmetric submodular set functions to capture concrete applications of interest such as element and vertex connectivity. Given the abstract nature of this discussion, in constrast to the concrete algorithmic applications, we have elected to give a brief and minimal discussion of the bisubmodular framework here. A more comprehensive description, including many more examples as well as the proofs of all lemmas and theorems stated here, can be found in the full version [10].

Let V be a finite set of elements. An ordered pair $(A, B) \in 2^V \times 2^V$ is a *set-pair* over V. For a family of set-pairs $\mathcal{V} \subseteq 2^V \times 2^V$ over V, we say that $\mathcal{V}$ is a *crossing lattice*[2] over V if it is closed under the following two operators.

$$(X_1, Y_1) \vee (X_2, Y_2) = (X_1 \cup X_2, Y_1 \cap Y_2).$$
$$(X_1, Y_1) \wedge (X_2, Y_2) = (X_1 \cap X_2, Y_1 \cup Y_2).$$

If $\mathcal{V}$ is closed under these operations, then $\mathcal{V}$ is a lattice under the partial order

$$(X_1, Y_1) \preceq (X_2, Y_2) \iff X_1 \subseteq X_2,\ Y_2 \subseteq Y_1.$$

[2] This notion is analogous to the definition of a crossing family of sets.

The binary operator $\vee$ returns the unique least upper bound of its arguments (a.k.a. the *meet*) and the binary operator $\wedge$ returns the unique greatest lower bound of its arguments (a.k.a. the *join*).

For a pair of sets $(X,Y) \in 2^V \times 2^V$, the *transpose* of (X,Y), denoted $(X,Y)^T$, is the reversed pair of sets $(X,Y)^T \overset{\text{def}}{=} (Y,X)$. A crossing lattice $\mathcal{V} \subseteq 2^V \times 2^V$ is *symmetric* if is closed under taking the transpose. We have the following identities relating the transpose with the lattice operations $\vee$ and $\wedge$. Observe that for $\mathcal{X}, \mathcal{Y} \in \mathcal{V}$, we have $\left(\mathcal{X}^T\right)^T = \mathcal{X}$, $(\mathcal{X} \vee \mathcal{Y})^T = \mathcal{X}^T \wedge \mathcal{Y}^T$, and $(\mathcal{X} \wedge \mathcal{Y})^T = \mathcal{X}^T \vee \mathcal{Y}^T$. Lastly, A crossing lattice $\mathcal{V} \subseteq 2^V \times 2^V$ is *pairwise disjoint* if $X \cap Y = \emptyset$ for all $(X,Y) \in \mathcal{V}$.

We now define an abstract, lattice-based notion of cuts that unifies the various different families of cuts of interest in graphs. Let V be a set. For two set-pairs $\mathcal{S} = (S,T) \in 2^V \times 2^V$ and $\mathcal{X} = (X,Y) \in 2^V \times 2^V$, we denote

$$\mathcal{S} \subseteq \mathcal{X} \overset{\text{def}}{\iff} S \subseteq X, T \subseteq Y.$$

If $\mathcal{S} \subseteq \mathcal{X}$, then we say that $\mathcal{X}$ *cuts* $\mathcal{S}$ or that $\mathcal{X}$ is an $\mathcal{S}$*-cut*. If $\mathcal{V}$ is a crossing lattice over V, $R \subset V$ is a subset, and $\mathcal{R}$ is a crossing lattice over R, then we say that $\mathcal{V}$ *separates* $\mathcal{R}$ if for every $\mathcal{S} \in \mathcal{R}$, there is an $\mathcal{S}$-cut $\mathcal{X} \in \mathcal{V}$. The following lemma observes that cuts are closed under the two lattice operations.

▶ **Lemma 8.** *Let V be a set and let $R \subseteq V$. Let $\mathcal{V} \subseteq 2^V \times 2^V$ be a crossing lattice over V and let $\mathcal{R} \subseteq 2^R \times 2^R$ be a crossing lattice over R. Suppose that $\mathcal{V}$ separates $\mathcal{R}$. Let $\mathcal{S}_1, \mathcal{S}_2 \in \mathcal{R}$, let $\mathcal{X}_1 \in \mathcal{V}$ be an $\mathcal{S}_1$-cut, and let $\mathcal{X}_2 \in \mathcal{V}$ be an $\mathcal{S}_2$-cut. Then $\mathcal{X}_1 \vee \mathcal{X}_2$ is an $\mathcal{S}_1 \vee \mathcal{S}_2$-cut and $\mathcal{X}_1 \wedge \mathcal{X}_2$ is an $\mathcal{S}_1 \wedge \mathcal{S}_2$-cut.*

Now, let $\mathcal{V}$ be a lattice. A real-valued function $f : \mathcal{V} \to \mathbb{R}$ is *submodular* if for all $\mathcal{X}, \mathcal{Y} \in \mathcal{V}$,

$$f(\mathcal{X}) + f(\mathcal{Y}) \geq f(\mathcal{X} \vee \mathcal{Y}) + f(\mathcal{X} \wedge \mathcal{Y}).$$

Bisubmodular functions can be interpreted as submodular functions over particular crossing lattices. There are at least two definitions of bisubmodular function in the literature. These definitions are similar and we discuss both.

In one definition (e.g., in [48]), a function $f : 2^V \times 2^V \to \mathbb{R}$ is called *bisubmodular* if for all $X_1, Y_1, X_2, Y_2 \subseteq V$, we have

$$f(X_1, Y_1) + f(X_2, Y_2) \geq f(X_1 \cup X_2, Y_1 \cap Y_2) + f(X_1 \cap X_2, Y_1 \cup Y_2). \tag{1}$$

A bisubmodular function $f : 2^V \times 2^V \to \mathbb{R}$ is submodular over the crossing lattice of all set-pairs, $\mathcal{V} = 2^V \times 2^V$.

Another definition (e.g., [4, 3, 2, 19]) of a bisubmodular function f is that $f(X_1, Y_1)$ is only defined for disjoint sets X_1 and Y_1, and otherwise satisfies inequality (1) for these inputs. In this version, f is bisubmodular iff it is a submodular function over the lattice of disjoint sets, $\mathcal{V} = \{(X,Y) : X, Y \subseteq V, X \cap Y = \emptyset\}$.

Now, let $\mathcal{V} \subseteq 2^V \times 2^V$ be a symmetric crossing lattice. A function $f : \mathcal{V} \to \mathbb{R}$ is *symmetric* if for all $\mathcal{X} \in \mathcal{V}$, $f(\mathcal{X}) = f\left(\mathcal{X}^T\right)$. This is a different definition than for symmetric submodular *set* functions and generalizes the (more standard) set-based definition. Both undirected edge cuts and vertex cuts are examples of symmetric submodular functions over appropriate symmetric crossing lattices.

There is an important relationship between the sets of terminals being separated and *minimal* minimum cuts that separate them, highlighted in the following lemma.

▶ **Lemma 9.** *Let V be a set and $R \subset V$. Let $\mathcal{V} \subseteq 2^V \times 2^V$ be a symmetric crossing lattice over V and let $\mathcal{R} \subseteq 2^R \times 2^R$ be a symmetric crossing lattice over R, such that $\mathcal{V}$ separates $\mathcal{R}$. Let $f : \mathcal{V} \to \mathbb{R}$ be a symmetric bisubmodular function. Consider the function $h : \mathcal{R} \to \mathcal{V}$ where $h(\mathcal{S})$ is defined as $\preceq$-minimum, f-minimum $\mathcal{S}$-cut. Then h is well-defined and carries the partial orders on $\mathcal{R}$ to $\mathcal{V}$; that is, $\mathcal{S}_1 \preceq \mathcal{S}_2 \implies h(\mathcal{S}_1) \preceq h(\mathcal{S}_2)$.*

The following is a particularly convenient form of Lemma 9, and the one applied directly in the sequel.

▶ **Lemma 10.** *Let V be a set and $R \subset V$. Let $\mathcal{V} \subseteq 2^V \times 2^V$ be a symmetric crossing lattice over V and let $\mathcal{R} \subseteq 2^R \times 2^R$ be a symmetric crossing lattice over R, such that $\mathcal{V}$ separates $\mathcal{R}$. Let $f : \mathcal{V} \to \mathbb{R}$ be a symmetric bisubmodular function. Let $\mathcal{S}_1, \ldots, \mathcal{S}_k \in \mathcal{R}$ and $\mathcal{X}_1, \ldots, \mathcal{X}_k \in \mathcal{V}$ such that for all $i \in [k]$, $\mathcal{X}_i$ is an f-minimum $\mathcal{S}_i$-cut. Then for any $\mathcal{S} \in \mathcal{R}$ such that $\mathcal{S} \preceq \mathcal{S}_i$ for all i, there is an f-minimum $\mathcal{X}$-cut with $\mathcal{X} \preceq \mathcal{X}_1 \wedge \cdots \wedge \mathcal{X}_k$.*

We now come to the issue of computing isolating cuts. We formalize this as follows. Let V be a set and $R \subset V$. Let $\mathcal{V} \subseteq 2^V \times 2^V$ be a symmetric and *pairwise disjoint* crossing lattice over V and let $\mathcal{R} \subseteq 2^R \times 2^R$ be the symmetric and *pairwise disjoint* crossing lattice over R consisting of all partitions of R; i.e., $\mathcal{R} = \{(S,T) : S \cup T = R,\ S \cap T = \emptyset\}$. Let $f : \mathcal{V} \to \mathbb{R}$ be a symmetric bisubmodular function. For each $r \in R$ we wish to find an f-minimum cut $\mathcal{Y}_r$ for the set-pair $(\{r\}, R - \{r\})$ (which we abbreviate as $(r, R - r)$ for notational simplicity). The main property that leads to efficiency is captured by the next lemma.

▶ **Lemma 11.** *Let V be a set and $R \subset V$. Let $\mathcal{V} \subseteq 2^V \times 2^V$ be a symmetric and* pairwise disjoint *crossing lattice over V and let $\mathcal{R} \subseteq 2^R \times 2^R$ be the symmetric and* pairwise disjoint *crossing lattice over R consisting of all partitions of R; i.e., $\mathcal{R} = \{(S,T) : S \cup T = R,\ S \cap T = \emptyset\}$. Let $f : \mathcal{V} \to \mathbb{R}$ be a symmetric bisubmodular function. Suppose we had access to an oracle that, given $\mathcal{S} \in \mathcal{R}$, returns a minimum $\mathcal{S}$-cut $\mathcal{W} \in \mathcal{V}$. Let $k = \lceil \log |R| \rceil$. Then with k calls to the oracle, one can compute k cuts $\mathcal{W}_1, \ldots, \mathcal{W}_k \in \mathcal{V}$ such that the following holds.*

For each $r \in R$, let $\mathcal{X}_r = \left(\bigwedge_{i:(r,V-r) \preceq \mathcal{W}_i} \mathcal{W}_i\right) \wedge \left(\bigwedge_{i:(r,V-r) \preceq \mathcal{W}_i^T} \mathcal{W}_i^T\right)$ be the intersection of cuts transposed to always include r in the first component. Then we have the following. (1) For all $r \in R$, $\mathcal{X}_r$ is an $(r, R-r)$-cut. (2) For all $r \in R$, there is a minimum $(r, R-r)$-cut $\mathcal{Y}_r$ such that $\mathcal{Y}_r \preceq \mathcal{X}_r$. (3) For any two distinct elements $r, q \in R$, $\mathcal{X}_r \wedge \mathcal{X}_q \preceq (\emptyset, R)$. (That is, the first components of the set pairs $\mathcal{X}_r$ are pairwise disjoint.)

Using the preceding lemma the problem of computing the f-minimum r-isolating cuts is reduced to finding such a cut in $\mathcal{X}_r$. The advantage, in terms of running time, is captured by the disjointness property: for distinct $r, q \in R$ we have $\mathcal{X}_r \wedge \mathcal{X}_q \preceq (\emptyset, R)$. For each r let $\mathcal{X}_r = (A_r, B_r)$. Thus we have $\sum_r |A_r| \leq |V|$. Given r and $\mathcal{X}_r$, the problem of computing the f-minimum cut $\mathcal{Y}_r \preceq \mathcal{X}_r$ can in several settings be reduced to solving a problem that depends only on $|A_r|$ and $|V|$. We capture this in the following lemma.

▶ **Lemma 12.** *Let V be a set and $R \subset V$. Let $\mathcal{V} \subseteq 2^V \times 2^V$ be a symmetric and* pairwise disjoint *crossing lattice over V and let $\mathcal{R} \subseteq 2^R \times 2^R$ be the symmetric and* pairwise disjoint *crossing lattice over R consisting of all partitions of R; i.e., $\mathcal{R} = \{(S,T) : S \cup T = R,\ S \cap T = \emptyset\}$. Let $f : \mathcal{V} \to \mathbb{R}$ be a symmetric bisubmodular function. Suppose we had access to an oracle that, given $\mathcal{S} \in \mathcal{R}$, returns a minimum $\mathcal{S}$-cut $\mathcal{W} \in \mathcal{V}$ and let $\mathrm{SM}(n)$ denote its running time where $n = |V|$. Moreover, suppose we have an oracle that given any $u \in R$ and $(A_u, B_u) \in \mathcal{V}$ with $u \in A_u$ outputs an f-minimum cut $\mathcal{Y}_u \preceq (A_u, B_u)$ in time $\mathrm{SMI}(|A_u|, n)$. Let $k = \lceil \log |R| \rceil$. Then, one can compute for each $r \in R$ an f-minimium r-isolating cut in in total time $O(k\,\mathrm{SM}(n) + \max_{0 \leq n_1, n_2, \ldots, n_{|R|} : \sum_i n_i = n} \sum_{i=1}^{|R|} \mathrm{SMI}(n_i, n))$.*

A simple random sampling approach combined with isolating cuts, as shown in [37] for edge cuts in graphs, yields the following theorem in a much more abstract setting.

▶ **Theorem 13.** *Let V be a set and $R \subset V$. Let $\mathcal{V} \subseteq 2^V \times 2^V$ be a symmetric and pairwise disjoint crossing lattice over V and let $\mathcal{R} \subseteq 2^R \times 2^R$ be the symmetric and pairwise disjoint crossing lattice over R consisting of all disjoint subsets of R; i.e., $\mathcal{R} = \{(S,T) : S,T \subseteq R, S \cap T = \emptyset\}$. Let $f : \mathcal{V} \to \mathbb{R}$ be a symmetric bisubmodular function. Suppose we had access to an oracle that, given $\mathcal{S} \in \mathcal{R}$, returns a minimum $\mathcal{S}$-cut $\mathcal{W} \in \mathcal{V}$ and let $\mathrm{SM}(n)$ denote its running time where $n = |V|$. Moreover, suppose we have an oracle that given any $u \in R$ and $(A_u, B_u) \in \mathcal{V}$ with $u \in A_u$ outputs an f-minimum cut $\mathcal{Y}_u \preceq (A_u, B_u)$ in time $\mathrm{SMI}(|A_u|, n)$. Then one can compute the minimum (nontrivial) $\mathcal{R}$-cut with constant probability in $O\Big(\mathrm{SM}(n)\log^2|R| + \max_{0 \le n_1, n_2, \ldots, n_{|R|} : \sum_i n_i = n} \log(|R|) \sum_{i=1}^{|R|} \mathrm{SMI}(n_i, n)\Big)$ time.*

We derive the following corollary for symmetric submodular set functions.

▶ **Corollary 14.** *Let $f : 2^V \to \mathbb{R}$ be a symmetric submodular function and $R \subseteq V$ and let $n = |V|$. Suppose there is an algorithm for submodular function minimization in the value oracle model in time $\mathrm{SFM}(n) = g_1(n)EO + g_2(n)$ where EO is the time for the evaluation oracle. Assuming that $g_1(n) = \Omega(n)$ and $g_2(n) = \Omega(n)$, a minimum f-cut that separates some two terminals in R can be found in $O\big(\mathrm{SFM}(n)\log^2(n)\big)$ time.*

3 Element connectivity

Let $G = (V, E)$ be an undirected graph with m edges and n vertices. Let $T \subseteq V$ be a set of terminals and let $N = V \setminus T$ be the non-terminal set. For any two distinct terminals $u, v \in T$, the element connectivity between u and v is defined as the maximum number of paths from u to v that are edge-disjoint and vertex-disjoint in the non-terminal vertices $V \setminus T$. That is, only terminal vertices may be reused across paths. This notion can be easily generalized to the weighted setting where edges and non-terminals have non-negative weights/capacities. For any two terminals $s, t \in T$, we denote by $\kappa'(s,t)$ the element connectivity between them. One can compute $\kappa'(s,t)$ via a simple reduction to s-t maximum flow in a directed graph which takes $\mathrm{EC}(m,n)$ time. In this section we are concerned with the problem of computing the global element connectivity which is defined as $\kappa' = \min_{s,t \in T, s \neq t} \kappa'(s,t)$. In fact we are also interested in computing the more general problem of computing $\kappa'(R) = \min_{s,t \in R, s \neq t} \kappa'(s,t)$ where $R \subseteq T$; note that $\kappa' = \kappa'(T)$. Here we apply our general framework that obtains a randomized algorithm with running time $O(\mathrm{EC}(m,n)\log^2|R|)$. In addition to the global minimum cut for R, as we will see in the next section, finding all the isolating cuts can be used with other ideas for vertex connectivity.

Let $G = (T \uplus N, E)$ be an instance of a weighted element connectivity problem. Let $w : N \cup E \to \mathbb{R}_{\ge 0}$ assign weights to the elements. Let $R \subseteq T$ be a subset of terminals with $|R| \ge 2$. We reduce the problem of computing $\kappa'(R)$ to Theorem 13 as follows.

For ease of notation, let $\overline{V} = V \cup E$ denote the elements. Consider the family of pairs of sets $\mathcal{V} \subseteq 2^{\overline{V}} \times 2^{\overline{V}}$ defined as the set of pairs $(X, Y) \in \overline{V} \times \overline{V}$ with the following properties: (i) X and Y are disjoint, (ii) no edge in X is adjacent to a vertex in Y, and no edge in Y is adjacent to a vertex in X, and (iii) $T \subseteq X \cup Y$. $\mathcal{V}$ describes the disjoint sets that are element-wise disconnected and cover T. Clearly $\overline{V}$ is symmetric and pairwise disjoint. It is also straightforward to verify that $\overline{V}$ is a crossing lattice.

We define a function $f : \mathcal{V} \to \mathbb{R}$ by $f(X,Y) = \sum_{x \in \overline{V} - (X \cup Y)} w(x)$. $f(\mathcal{X})$ gives the total weight of elements that are not a member of either of the two sets in $\mathcal{X}$. This function f is submodular and in fact it is modular. One can easily verify that $f(X_1, Y_1) + f(X_2, Y_2) = f(X_1 \cup X_2, Y_1 \cap Y_2) + f(X_1 \cap X_2, Y_1 \cup Y_2)$. Thus $\mathcal{V}$ is a symmetric and pairwise disjoint crossing lattice, and $f : \mathcal{V} \to \mathbb{R}$ is a symmetric submodular function over $\mathcal{V}$.

Isolating (weighted) element cuts and global connectivity Let $R \subseteq T$ and let $\mathcal{R}$ be the crossing lattice consisting of all pairwise disjoint subsets of R. Given a partition of R into two sets (A, B), an f-minimum (A, B)-cut, which corresponds to the minimum element cut separating A from B, can be computed via directed (s,t)-maxflow, in $\mathrm{EC}(m, n)$-time. By Lemma 11, we can compute disjoint sets of elements $\left\{\overline{U}_r \subset \overline{V} : r \in R\right\}$ where for each $r \in R$, $\overline{U}_r$ contains the r-side component of a minimum $(r, R - r)$-element cut. Moreover, because the $\overline{U}_r$'s are obtained as intersections of sides of element cuts, for any distinct $r, q \in R$, there is no edge from $\overline{U}_r$ incident to a vertex from $\overline{U}_q$ and (symmetrically) vice-versa.

For each r, let $\overline{U}'_r \subset \overline{V}$ be the set of vertices outside $\overline{U}_r$ and incident to an edge in $\overline{U}_r$, and the edges outside $\overline{U}_r$ incident to vertices in $\overline{U}_r$. Informally speaking, $\overline{U}'_r$ is the "boundary" of $\overline{U}_r$ in an element connectivity sense. Let $n_r = \left|V \cap \left(\overline{U}_r \cup \overline{U}'_r\right)\right|$ be the number of vertices in $\overline{U}_r \cup \overline{U}'_r$ and let $m_r = \left|E \cap \left(\overline{U}_r \cup \overline{U}'_r\right)\right|$ be the number of edges in $\overline{U}_r \cup \overline{U}'_r$. Note that $\sum_r m_r \leq 2m$ since each edge can either appear in $\overline{U}_r$ for a unique choice of r or in $\overline{U}'_r$ for two choices of r.

To find an isolating cut for r we need to find the cheapest element cut contained in $\overline{U}_r$. We can do this via a flow computation as described below. For each r, consider the graph G_r where we first take the graph $\overline{U}_r \cup \overline{U}'_r$ and introduce an auxiliary vertex $\bar{t}$. We connect $\bar{t}$ to all vertices in $\overline{U}'_r$ with infinite capacity. For every edge $e \in \overline{U}'_r$ with exactly one endpoint in $\overline{U}_r$, we replace the opposite endpoint with $\bar{t}$. Observe that the minimum $(r, \bar{t})$-element cut in G_r coincides with the minimum $(r, R - r)$-element cut in G. G_r has $O(m_r)$ edges and $O(n_r)$ vertices, and the element $(r, \bar{t})$-cut problem can be solved in $\mathrm{EC}(m_r, n_r)$ time. Summing over all $r \in R$ gives the following theorem.

▶ **Theorem 15.** *Let $G = (T \cup N, E)$ be an instance of element connectivity with n nodes and m edges and let $R \subseteq T$. Let $w : V \cup E \to (-\infty, \infty]$ assign positive weight to each vertex and edge. Let $k = |T|$. Then one can compute, for all $r \in R$, the minimum weight element $(r, R - r)$-cut in $O\left(\mathrm{EC}(m, n) \log k + \max_{m_1, \ldots, m_k} \left\{\sum_{i=1}^k \mathrm{EC}(m_i, n) : m_1 + \cdots + m_k \leq 2m\right\}\right)$, where $\mathrm{EC}(m, n)$ is the running time for element (S, T)-cut with m edges and n vertices.*

With Theorem 15 in place we can reduce the global mincut problem for R to the isolating cut computation via sampling [37], and captured in the abstract setting Theorem 13, to obtain the following theorem to compute $\kappa'(R)$.

▶ **Theorem 16.** *Let $G = (T \cup N, E)$ be an instance of element connectivity with n nodes and m edges and let $R \subseteq T$. Let $w : V \cup E \to (-\infty, \infty]$ assign positive weight to each vertex and edge. Let $k = |R|$. Then one can compute $\kappa'(R)$ with constant probability in time $O((\mathrm{EC}(m, n) \log(k) + \max_{m_1, \ldots, m_n} \{\sum_{i=1}^n \mathrm{EC}(m_i, n) : m_1 + \cdots + m_n \leq 2m\}) \log n)$.*

3.1 Refined running times for element connectivity

Until recently, the leading running times for $\mathrm{EC}(m, n)$ (e.g., $\mathrm{EC}(m, n) = \tilde{O}(m\sqrt{n})$ [34]) plug directly into Theorem 16 to give running times of the form $\tilde{O}(\mathrm{EC}(m, n))$ to compute the global element connectivity. A recent breakthrough work by [50] has obtained a running

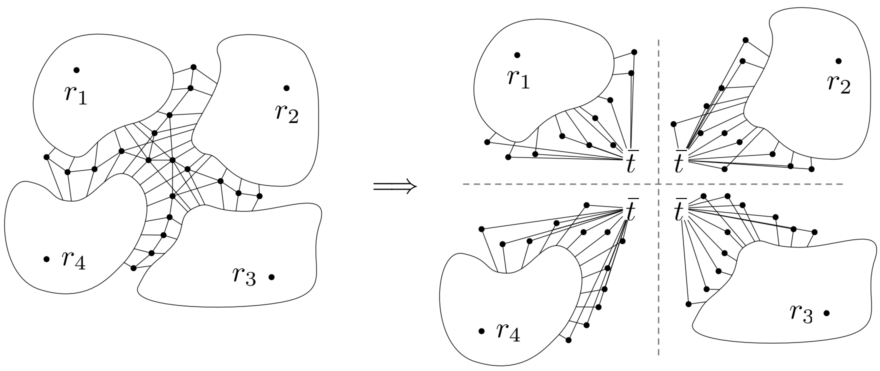

Figure 1 Applying the uncrossing framework to vertex connectivity reduces vertex isolating cuts to edge disjoint cut problems. Note that the separating vertices may appear in multiple subproblems, which is an obstruction towards a direct $\tilde{O}(\mathrm{EC}(m,n))$ overall running time for isolating vertex cuts.

time of $\mathrm{EC}(m,n) = \tilde{O}\big(m + n^{1.5}\big)$ for polynomially bounded and integral capacities. However, Theorem 16 does not directly benefit from this running time because the vertices are not partitioned across subproblems. See Figure 1 for an illustration in the concrete setting of vertex cuts. Consequently, plugging $\mathrm{EC}(m,n) = \tilde{O}\big(m + n^{1.5}\big)$ directly into Theorem 16 generates a running time of $\tilde{O}\big(m + n^{1.5}k\big)$, where $k = |T|$. The additional factor of k (to a certain extent) defeats the purpose of the isolating cuts framework.

In this section, we develop more advanced algorithms that take the isolating cut framework as a starting point, and incorporates additional ideas to take advantage of $\mathrm{EC}(m,n) = \tilde{O}\big(m + n^{1.5}\big)$. In addition to obtaining faster algorithms, these results point to a general algorithm design space where additional ideas can be introduced to obtain even better running times. The first algorithm we present leverages the fact that the edges are partitioned across subproblems, even if the vertices are not.

▶ **Theorem 17.** *Let $G = (T \cup N, E)$ be an instance of element connectivity with n nodes and m edges and let $R \subseteq T$. Let $w : V \cup E \to [1..U]$ assign integer (or infinite) weights to each vertex and edge. For $R \subseteq T$, the minimum R-isolating vertex cut can be computed in*

$$\tilde{O}\Big(m^{1+o(1)} n^{3/8} U^{1/4} + n^{1.5}\Big)$$

time.

Proof. Let $k = |R|$. We apply Theorem 15 and give concrete upper bounds using known upper bounds for $\mathrm{EC}(m,n)$. Let $m_1, \ldots, m_k \in \mathbb{N}$ with $m_1 + \cdots + m_k \leq m$. Recall that $\mathrm{EC}(m,n) = \tilde{O}\big(m^{4/3+o(1)} U^{1/3}\big)$ by [31] and $\mathrm{EC}(m,n) = \tilde{O}\big(m + n^{3/2}\big)$ by [50]. Let $\alpha > 0$ be a parameter to be determined. We apply the first running time when $m_i < m\alpha/k$ and the second running time then $m_i \geq m\alpha/k$. At most k/α indices i have $m_i \geq m\alpha/k$. Thus,

$$\begin{aligned}
\sum_{i=1}^{k} \mathrm{EC}(m_i, n) &= \sum_{i : m_i \geq m\alpha/k} \mathrm{EC}(m_i, n) + \sum_{i : m_i < m\alpha/k} \mathrm{EC}(m_i, n) \\
&\leq \tilde{O}\left(m + \frac{k}{\alpha} n^{1.5} + \sum_{i : m_i < m\alpha/k} m_i^{4/3+o(1)} U^{1/3} \right) \\
&\overset{\text{(a)}}{\leq} \tilde{O}\left(m + \frac{k}{\alpha}\left(n^{1.5} + \left(\frac{\alpha m}{k}\right)^{4/3+o(1)} U^{1/3} \right)\right).
\end{aligned}$$

Here (a) is by convexity: the quantity $\sum_{i:m_i<m\alpha/k} m_i^{4/3+o(1)}$ subject to the condition that $\sum_i m_i \leq m$ is at most $(k/\alpha)(\alpha m/k)^{4/3+o(1)}$. Balancing terms at $\alpha = kn^{9/8}/m$, this gives an upper bound of $\tilde{O}\big(m^{1+o(1)}n^{3/8}U^{1/4}\big)$, hence the claimed running time. ◀

We point out that other running time tradeoffs between m and U can be obtained by instead applying the flow alogrithms from [39, 40].

The next theorem, which is particularly good for dense graphs, leverages the fact that while the vertices are not necessarily partitioned across subproblems, at least the "inner" vertex sets $\overline{U}_r \cap V$ are disjoint and all of the repeating "boundary" vertices are guaranteed to be outside the r-component of each $(r, R-r)$-minimum cut. The following algorithm balances a tradeoff between the recent algorithm with [51] with blocking flows [22]. In the application of blocking flows, we argue that with an appropriate construction of the auxiliary graph in the components given by the decomposition by isolating cuts, the maximum length of any augmenting paths is proportional to the number of inner vertices (rather than the total number of vertices) for that component.

▶ **Theorem 18.** *Let $G = (T \cup N, E)$ be an instance of element connectivity with n nodes and m edges and let $R \subseteq T$. Let $w : V \cup E \to [0, U]$ assign positive (or infinite) weights to each vertex and edge. For $R \subseteq T$, the minimum R-isolating cut can be computed in*

$$\tilde{O}\Big(m^{1/2}n^{5/4}\Big)$$

randomized time, where $\tilde{O}(\cdots)$ hides $\operatorname{poly}(\log(n), \log(U))$*-factors.*

Proof. We recall the construction from Theorem 15, adopting the same notation. In addition, for each r, let $\tilde{n}_r$ be the number of vertices in $\overline{U}_r$. Note that as the $\bar{U}_r$'s are disjoint, we have $\sum_{r\in R} \tilde{n}_r \leq n$.

For each r, we employ two different approaches to computing the minimum $(r,\bar{t})$-element cut. On one hand we can apply any max flow algorithm in $\mathrm{EC}(m_r, n_r)$ time. As remarked above we have $\mathrm{EC}(m_r, n_r) \leq \tilde{O}\big(m_r + n_r^{1.5}\big)$ by [50]. The second approach is to apply blocking flows with the following additional observations. Element connectivity can be modeled as maximum flow in undirected graphs with edge and vertex capacities, which in turn can be reduced to maximum flow in edge capacitated directed graphs. Recall the directed graph representation of vertex capacities, sometimes called the "split graph". We remind the reader that in the split graph, each non-terminal vertex $v \in V_r \setminus \{r, \bar{t}\}$ is split into two vertices – an "in-vertex" v^- and an "out-vertex" v^+ – and there is an edge (v^-, v^+) with capacity equal to $w(v)$. Each edge $(u, v) \in E_r$ is replaced with an edge (u^+, v^-) with the same capacity. From this split graph, we contract the edges $(v^+, \bar{t})$ for all $v \in V \cap \overline{U}'_r$, which is safe because $\bar{t}$ is the sink and each edge $(v^+, \bar{t})$ has infinite capacity. Now, in this directed auxiliary graph, we have $O(m_r)$ edges and $O(n_r)$ vertices. We now observe that the auxiliary vertices corresponding to $\overline{U}'_r \cap V$, $\Big\{v^- : v \in \overline{U}'_r \cap V\Big\}$, do not have any edges between them. Then any $(s, \bar{t})$ path in this graph or in any residual graph that may arise cannot have consecutive auxiliary vertices from $\overline{U}'_r$. Therefore, every augmenting path has length at most $2\tilde{n}_r$. In turn, $O(\tilde{n}_r)$ iterations of blocking flows suffice to find the minimum $(r, \bar{t})$ cut in G_r, which takes $O(m_r \log(m_r/n_r))$ time per iteration [22] and $O(m_r \tilde{n}_r \log(m_r/n_r))$ time overall.

Let $\alpha > 0$ be a parameter to be determined. We have $\tilde{n}_r \geq \alpha n/k$ for at most k/α vertices $r \in R$. We have

$$\begin{aligned} O\left(\sum_r \min\{\mathrm{EC}(m_r, n_r), m_r \tilde{n}_r \log(m)\}\right) &\leq \tilde{O}\left(\sum_{r:\tilde{n}_r \leq \alpha n/k} m_r \tilde{n}_r + \sum_{r:\tilde{n}_r \geq \alpha n/k} \big(m_r + n_r^{1.5}\big)\right) \\ &\leq \tilde{O}\Big(m + \Big(\frac{\alpha}{k}\Big) mn + \Big(\frac{k}{\alpha}\Big) n^{1.5}\Big) \\ &\overset{(a)}{\leq} \tilde{O}\big(m + m^{1/2} n^{5/4}\big) = \tilde{O}\big(m^{1/2} n^{5/4}\big), \end{aligned}$$

as desired. Here, in step (a), we substituted $\alpha = kn^{1/4}/m^{1/2}$. ◀

4 Vertex connectivity

In this section we consider the problem of computing the vertex connectivity in weighted and unweighted graphs. Let $G = (V, E)$ be an undirected graph with m edges and n vertices. Let $w : V \to [1, U]$ be positive vertex weights. Given distinct nodes $s, t \in V$ such that $st \notin E$, the minimum weight vertex separator between s and t can be computed via flow techniques. Recently there has been significant improvement in the running time of vertex capacitated flow to $\tilde{O}(m+n^{1.5})$ [51]. We use $\mathrm{VC}(m, n)$ to denote the complexity of computing such a separator. We let $\kappa(s, t)$ denote the weight of the separator between s, t with the understanding that $\kappa(s, t) = \infty$ if $\{s, t\} \in E$. Here we are interested in the minimum vertex weight separator of G which can be defined as $\min_{s,t \in V, s \neq t} \kappa(s, t)$.

Let $R \subset V$ such that R is an *independent set* in G; that is, no two vertices in R share an edge. One can then define $\kappa(R)$ to be $\min_{s,t \in R, s \neq t} \kappa(s, t)$. We observe that $\kappa(R)$ is the same as the element connectivity of R in the graph where R is the set of terminals and $V \setminus R$ are the non-terminals and edge weights are set to ∞; i.e., only vertices are allowed to be removed. We have already seen algorithms for element connectivity, which immediately convert to isolating cut algorithms for vertex connectivity. For instance, one can compute the minimum isolating cut in $\tilde{O}\big(m^{1+o(1)} n^{3/8} U^{1/4} + n^{1.5}\big)$ time with integral vertex weights between 1 and U, or in $\tilde{O}\big(\sqrt{m} n^{5/4}\big)$ time for polynomially bounded weights.

These running times for isolating cuts do not, however, immediately convert to running times for vertex cuts. To obtain the minimum vertex cut as an isolating cut, we must initialize the algorithm with a set of vertices R for which the minimum vertex cut is also an isolating cut. Let (S, T) be opposite sides of a minimum vertex cut $N(S) = N(T)$. Without loss of generality suppose S has weight less than or equal to T. We would like a set R that samples exactly one point from S, at least one point from T, and avoids $N(S)$ altogether. Even in the unweighted setting, uniform sampling is thwarted by the fact that $N(S)$ may be much larger than S, and it is difficult to hit S without hitting $N(S)$ too. In the following lemma, we observe that if we relax our problem to an $(1 + \epsilon)$-approximately minimum vertex cut, then we can sample a useful set R with reasonably good probability.

▶ **Lemma 19.** *Let $\epsilon > 0$ be fixed. Let $G = (V, E)$ be an undirected graph with m edges and n vertices. Let $w : V \to [1, U]$ be positive vertex weights and let $W = \sum_{v \in V} w(v)$ be the total weight. Let κ be the weight of the minimum weight vertex cut. Suppose the minimum weighted degree is greater than $(1 + \epsilon)\kappa$. Then one can compute a randomized independent set $R \subset V$ such that the minimum vertex cut is an R-isolating set with probability at least $\Omega((\epsilon/\log(nU)) \max\{\epsilon, (1 - \kappa/W)\})$.*

Proof. For ease of notation, let

$$\epsilon_0 = \max\left\{\epsilon, \frac{1}{2}\left(1 - \frac{\kappa}{W}\right)\right\}$$

Let (S, T) be opposite sides of the minimum vertex cut $N(S) = N(T)$. Without loss of generality suppose $w(S) \leq w(T)$ where we use the notation $w(A)$ to denote the total weight of vertices in A, that is, $w(A) = \sum_{v \in A} w(v)$. Since $N(S)$ is the minimum weight vertex separator we have $w(N(S)) = \kappa$. We would like a independent set $R \subset V$ that has exactly one point from S, at least one point from T, and avoids $N(S)$ altogether. Then (S, T) would isolate the lone vertex in $R \cap S$ from $R - r \subseteq T$, as desired. We can achieve this via a sampling procedure that we described below.

First we claim that $w(S) \geq \epsilon\kappa$. Fix an arbitrary vertex $v \in S$. By assumption, $w(N(v)) \geq (1 + \epsilon)\kappa$. Since $N(v) \subseteq S \cup N(S)$ and $S \cap N(S) = \emptyset$, we have $w(S) \geq \epsilon\kappa$.

Let μ be any value in the range $[2\max\{w(S), \kappa\}, 4\max\{w(S), \kappa\}]$. Since $\max\{\sum_{v \in S} w(v), \kappa\}$ lies in the range $[1, \text{poly}(n, U)]$, we can sample a value μ that lies in the above range with probability $\Omega(1/\log(nU))$ by randomly picking a power of 2 in the range $[1, \text{poly}(n, U)]$. Once we fix μ, let R be a random subset of vertices obtained by independently sampling each vertex v with probability $w(v)/\mu$. Then, as long as R has an adjacent pair of vertices, we remove one of them from R. We claim that the intial sample for R has one point from S, no points from $N(S)$, and at least one point from T with probability $\geq \Omega(\epsilon\epsilon_0)$. If so, then since S and T are independent from one another, dropping vertices in the second phase will not remove any vertices from S, and retain at least one vertex in T, as desired. Observe that the three events are independent.

The probability that R avoids $N(S)$ is $\prod_{v \in N(S)}(1 - w(v)/\mu)$. Since $w(N(S)) = \kappa$ and $\mu \geq 2\kappa$, for any $v \in N(S)$, $w(v)/\mu \leq 1/2$. For $x \in (0, 1/2]$ the inequality $(1 - x) \geq e^{-2x}$ holds. Hence, $\prod_{v \in N(S)}(1 - w(v)/\mu) \geq \prod_{v \in N(S)} e^{-2w(v)/\mu} \geq e^{-2\kappa/\mu} \geq 1/e$.

Recall that $w(S) \geq \epsilon\kappa$ and hence the probability that R samples exactly one vertex from S is

$$\sum_{v \in S} \frac{w(v)}{\mu} \prod_{u \in S - \{v\}} (1 - \frac{w(u)}{\mu}) \geq \sum_{v \in S} \frac{w(v)}{\mu} e^{-2(w(S) - w(v))/\mu} \geq \frac{1}{e} \sum_{v \in S} \frac{w(v)}{\mu} \geq \frac{\epsilon}{4e}.$$

In the preceding set of inequalities we used the fact that $1 - x \geq e^{-2x}$ for $x \in [0, 1/2]$ since $w(S)/\mu \leq 1/2$. In the final inequality we used the fact that $w(S) \geq \epsilon\kappa$ which implies that $w(S) \geq \epsilon\mu/4$.

We claim that $w(T) \geq \epsilon_0\mu/4$. Assuming the claim, the probability that R samples at least one vertex from T is $\geq 1 - e^{-w(T)/\mu} \geq 1 - e^{\epsilon_0/4} = \Omega(\epsilon_0)$. To see the claim, recall that $w(T) \geq w(S) \geq \epsilon\kappa$. We also have $w(S) + w(T) + w(N(S)) = W$ which implies that $w(T) \geq \frac{1}{2}(W - \kappa) \geq \frac{1}{2}(1 - \frac{\kappa}{W})W \geq \frac{1}{2}(1 - \frac{\kappa}{W})w(S)$. Since $\mu \leq 4\max\{w(S), \kappa\}$, we have $w(T) \geq \epsilon_0\mu/4$.

Thus, given μ lies in the range $[2\max\{w(S), \kappa\}, 4\max\{w(S), \kappa\}]$ which happens with probability $\Omega(1/\log(nU))$ we have the desired sample R with probability $\Omega(\epsilon \cdot \epsilon_0)$. ◀

4.1 Approximate vertex connectivity

By combining the isolating cut algorithms with the sampling lemma above (for the case where no singleton already induces a good enough vertex cut), we obtain the following approximation algorithm for vertex connectivity. We point out that in the running time below, the trailing factor $(\min\{1/\epsilon, W/(W - \kappa)\})$ is simply a constant except in the relatively uninteresting setting where the minimum weight vertex cut is almost all of the weight of the graph. In the regime of interest, the following is a $\tilde{O}(1/\epsilon)$ factor greater than the running time to compute an isolating vertex cut.

▶ **Theorem 20.** *Let $\epsilon > 0$ be fixed. Let $G = (V, E)$ be an undirected graph with m edges and n vertices. Let $w : V \to \mathbb{R}_{>0}$ be positive vertex weights and let $W = \sum_{v \in V} w(v)$ be the total weight. Let κ be the weight of the minimum weight vertex cut. Then a minimum vertex cut can be computed with high probability in $\tilde{O}((1/\epsilon)\,\mathrm{IsoVC}(m,n)\min\{(1/\epsilon), W/(W-\kappa)\})$ randomized time, where $\mathrm{IsoVC}(m,n)$ is the running time to compute the minimum isolating vertex cut in a weighted graph of m edges and n vertices.*

Proof. Let

$$\ell = \tilde{O}\left(\frac{1}{\epsilon}\min\left\{\frac{1}{\epsilon}, \frac{W}{W-\kappa}\right\}\right)$$

The algorithm first repeats the following subroutine $O(\ell)$ times. This subroutine first generates an set $R \subset V$ by Lemma 19, and then it computes a minimum R-isolating cut. It compares the ℓ isolating cuts generated above with the singleton cuts in the graph, returning the minimum overall.

We argue that the algorithm returns a $(1+\epsilon)$-approximate minimum weight cut with high probability by the following simple analysis. In one case, some singleton cut is an approximate minimum cut, in which case the algorithm always succeeds. In the second case, the minimum weighted degree is at least an $(1+\epsilon)$-multiplicative factor greater than the vertex connectivity. In that case, the minimum weight vertex cut is a minimum R-isolating cut for at least one of the random sets R with high probability, in which case we return the minimum weight vertex cut. ◀

We briefly compare our bound above to previous work. As mentioned previously Henzinger, Rao and Gabow [27] obtain a randomized algorithm that gives the exact vertex connectivity in $\tilde{O}(mn)$ time for weighted graphs. We obtain a $(1+\epsilon)$-approximation in $\tilde{O}(m\sqrt{n}/\epsilon)$ time or in $\tilde{O}(m^{1/2}n^{5/4}/\epsilon)$ time; other bounds are outlined in previous subsection. We are thus able to obtain substantially faster algorithm if we settle for a small approximation. There have been past works on approximation for vertex connectivity but as far as we know they have been limited to unweighted graphs. Henzinger [26] obtained a 2-approximation in $O(n^2\min(\sqrt{n},\kappa))$. Forster et al. obtained a $(1+\epsilon)$-approximation in randomized time $\tilde{O}(m + n\kappa^2/\epsilon)$ which is near-linear for small connectivity, and combining various other results they improve upon Henzinger's result. We refer the reader to [17] for detailed bounds. Our running times are useful for the larger connectivity regime and we can obtain improved bounds in various other regimes of interest. We leave a more detailed comparison to a future version of the paper.

4.2 Exact vertex connectivity

Now, for integral weights, the approximation algorithm above gives the following exact algorithm for vertex connectivity by suitable choice of ϵ. Again we highlight that in the running time below, the trailing factor $(\min\left\{\kappa, \frac{W}{W-\kappa}\right\})$ is simply a constant except in the relatively uninteresting setting where κ is almost $\sum_{v \in V} w(v)$, in which case the remaining factors of $O(\kappa\,\mathrm{IsoVC}(m,n))$ are not as compelling anyway.

Table 1 A table of running times for finding the minimum vertex cut in an *unweighted* and undirected graph. $\mathrm{VC}(m,n)$ denotes the running time of computing (s,t)-vertex connectivity. $\mathrm{EC}(m,n)$ denotes the running time computing (s,t)-edge connectivity. See also [48, Section 15.2a].

$O\big(n^2\,\mathrm{VC}(\kappa n, n)\big)$	Combines trivial algorithm with sparsification [44].
$O(n\,\mathrm{VC}(\kappa n, n))$	Combines randomized trivial algorithm with sparsification [44]. $\kappa \leq .99n$
$O(n^\omega + n\kappa^\omega)$.	[38].
$\tilde{O}\big(\kappa n^2\big)$	[27]. Randomized.
$O\big(\min\{n^{3/4}, \kappa^{3/2}\}\kappa^2 n + \kappa n^2\big)$	[20].
$\tilde{O}\big(m + \kappa^{7/3} n^{4/3}\big)$	[45]. Randomized.
$\tilde{O}\big(m + n\kappa^3\big)$	[17]. Randomized.
$\tilde{O}\big(m + \kappa^{7/3} n^{4/3}\big)$	Corollary 22. Randomized. $\kappa \leq .99n$
$\tilde{O}\big(m + \kappa^2 n^{11/8+o(1)} + \kappa n^{3/2}\big)$	Corollary 22. Randomized. $\kappa \leq .99n$
$\tilde{O}\big(m + \kappa^{1.5} n^{7/4}\big)$	Corollary 22. Randomized. $\kappa \leq .99n$.

▶ **Theorem 21.** *Let $G = (V, E)$ be an undirected graph with m edges and n vertices. Let $w : V \to \mathbb{N}$ be integer vertex weights and let $W = \sum_{v \in V} w(v)$ be the total weight. Let κ be the weight of the minimum weight vertex cut. Then the minimum vertex cut can be computed with high probability in $\tilde{O}(\kappa\,\mathrm{IsoVC}(m,n)\min\{\kappa, W/(W-\kappa)\})$ randomized time, where $\mathrm{IsoVC}(m,n)$ is the running time to compute the minimum isolating vertex cut in a weighted graph of m edges and n vertices.*

Proof. For integral capacities, a $(1 + 1/(\kappa+1))$-approximation is an exact solution. Thus the result follows from Theorem 20. ◀

For the unweighted case, combining the above with sparsification [44] gives the following.

▶ **Corollary 22.** *Let $G = (V, E)$ be a simple unweighted graph. Then the minimum vertex cut can be computed with high probability in $\tilde{O}(m + \kappa\,\mathrm{IsoVC}(n\kappa, n)\min\{\kappa, n/(n-\kappa)\})$ randomized time, where $\mathrm{IsoVC}(m,n)$ is the running time to compute the minimum isolating vertex cut in a graph of m edges and n vertices.*

Proof. For unweighted graphs we can assume we know κ (via exponential search which adds an additional $O(\log\kappa)$ overhead). We apply the well-known linear-time sparsification algorithm of Nagamochi and Ibaraki [44] to reduce the number of edges to $O(n\kappa)$ and then run the algorithm in the preceding theorem on the sparsified graph which gives the claimed bound. ◀

The reduction from exact vertex connectivity to isolating vertex cut above, mixed with the algorithms for isolating vertex cuts, and optionally including the sparsification step from Corollary 22, produces a number of new running times that are optimal for different ranges of κ. In general, the running times obtained here have a lower dependence on κ then other algorithms for vertex connectivity with a poly(κ) dependence (which is common for the unweighted setting), so the running times here are particularly good for moderate to large κ. For a more detailed comparison between the literature and new running times for the unweighted setting (where we restrict to unweighted for simplicity), see Table 1.

5 Hypergraph Connectivity

Let $H = (V, E)$ be a weighted hypergraph and let $R \subseteq V$. The cut function of a hypergraphs is symmetric and submodular. Given disjoint sets $S, T \subset V$ the minimum S-T cut in H can be computed in $\mathrm{EC}(p, m+n)$ time via standard reductions[3]. We can use Lemma 11 and Corollary 14 to understand the running time to compute R-connectivity in H. Up to logarithmic factors it suffices to estimate the time to find R-isolating cuts. Recall that the running time consists of two parts. The first part is $O(\log |R|)$ calls to S-T cut problem in H. After this we have the following situation. For each $r \in R$ we obtain a set $U_r \subset V$ such that $r \in R$ and $U_r \cap (R - r) = \emptyset$. Furthermore the sets U_r over $r \in R$ are pairwise disjoint. For each r the goal is to find a set $Y_r \subseteq U_r$ with minimum $w(\delta(Y_r))$ where $\delta(Y_r)$ is set of hyperedges crossing Y_r. Let $n_r = |U_r|$. We can compute Y_r by solving a cut problem in an auxiliary hypergraph G_r on $n_r + 1$ vertices obtained by shrinking $V \setminus U_r$ into a single vertex. Let p_r be the total size of the hyperedges in G_r. It is not hard to see that $\sum_{r \in R} p_r = O(p)$. Thus each cut problem in G_r can be computed in either $\mathrm{EC}(p_r, m + n_r + 1)$. This implies the following.

▶ **Theorem 23.** *The minimum isolating cuts over a set of vertices R of size $k = |R|$ in a hypergraph with m edges, n vertices, and total size p can be computed*

$$\tilde{O}\left(\mathrm{EC}(p, m+n) + \max_{n_1,\ldots,n_k,p_1,\ldots,p_k} \left\{\sum_{i=1}^{k} \mathrm{EC}(p_i, m+n_i) : n_1 + \cdots + n_k \leq n,\ p_1 + \cdots + p_k \leq 2p\right\}\right)$$

time with high probability.

In particular $\mathrm{EC}(p, m+n)$ is $\tilde{O}(p\sqrt{m+n}\log U)$ [34] and for unweighted case we have $\mathrm{EC}(p, m+n) = \tilde{O}(p^{4/3})$ [39]. We can obtain two other run times for hypergraphs that provide different tradeoffs. These are obtained by more carefully solving the second part of the isolating cut framework, and transfer ideas from vertex connectivity to hypergraphs.

1. $\sqrt{pn(m+n)^{1.5}}$.
2. $\tilde{O}\left(p(m+n)^{\frac{3\alpha}{2(1+\alpha)}}\beta^{\frac{1}{1+\alpha}}\right)$ for any α, β where $\mathrm{EC}(m, n) \leq m^{1+\alpha}\beta$ (e.g., [31] gives $\mathrm{EC}(m, n) \leq \tilde{O}(m^{4/3}U^{1/3})$, which we interpret as $\alpha = 1/3$ and $\beta = U^{1/3}$).

We sketch the proofs of theorems that obtain the preceding bounds.

▶ **Theorem 24.** *The minimum isolating cut in a hypergraph can be computed in*

$$\tilde{O}\left(\sqrt{pn(m+n)^{1.5}}\right)$$

randomized time.

The proof is omitted due to space constraints, and can be found in [10]. We mention that the approach is similar to the the algorithm for element isolating cuts that had a running time of $\tilde{O}(\sqrt{m}n^{5/4})$.

[3] One can also reduce to computing s-t cut in a vertex capacitated undirected graph with p edges and $m + n$ nodes, although there does not seem to be any particular advantage with current running time bounds for $\mathrm{EC}(p, m+n)$.

▶ **Theorem 25.** *Suppose* $\mathrm{EC}(m,n) \leq \tilde{O}\big(m^{1+\alpha}\beta\big)$ *for fixed* $\alpha, \beta > 0$*. Then minimum isolating cuts can be computed in* $\tilde{O}\Big(p(m+n)^{\frac{3\alpha}{2(1+\alpha)}}\beta^{\frac{1}{1+\alpha}} + p + (m+n)^{1.5}\Big)$.

The proof is omitted due to space constraints, and can be found in [10]. We mention that the approach is similar to the algorithm for element isolating cuts that obtained a running time of $\tilde{O}\big(m^{1+o(1)}n^{3/8}U^{1/4} + n^{1.5}\big)$.

References

1 Ashkan Aazami, Joseph Cheriyan, and Krishnam Raju Jampani. Approximation algorithms and hardness results for packing element-disjoint steiner trees in planar graphs. *Algorithmica*, 63(1-2):425–456, 2012.

2 Kazutoshi Ando and Satoru Fujishige. On structures of bisubmodular polyhedra. *Math. Program.*, 74:293–317, 1996.

3 Kazutoshi Ando, Satoru Fujishige, and Takeshi Naitoh. A characterization of bisubmodular functions. *Discret. Math.*, 148(1-3):299–303, 1996.

4 André Bouchet. Greedy algorithm and symmetric matroids. *Math. Program.*, 38(2):147–159, 1987.

5 Gruia Călinescu, Chandra Chekuri, and Jan Vondrák. Disjoint bases in a polymatroid. *Random Structures & Algorithms*, 35(4):418–430, 2009.

6 Ruoxu Cen, Jason Li, Danupon Nanongkai, Debmalya Panigrahi, and Thatchaphol Saranurak. Minimum cuts in directed graphs via $\sqrt{n}$ max-flows. *CoRR*, abs/2104.07898, 2021. `arXiv:2104.07898`.

7 Chandra Chekuri. Some open problems in element connectivity, September 2015. URL: `http://chekuri.cs.illinois.edu/papers/elem-connectivity-open-probs.pdf`.

8 Chandra Chekuri and Nitish Korula. A graph reduction step preserving element-connectivity and packing steiner trees and forests. *SIAM Journal on Discrete Mathematics*, 28(2):577–597, 2014.

9 Chandra Chekuri and Kent Quanrud. Faster algorithms for rooted connectivity in directed graphs, 2021. To appear in ICALP,. `arXiv:2104.07205`.

10 Chandra Chekuri and Kent Quanrud. Isolating cuts, (bi-)submodularity, and faster algorithms for global connectivity problems. *CoRR*, abs/2103.12908, 2021. `arXiv:2103.12908`.

11 Chandra Chekuri, Thapanapong Rukkanchanunt, and Chao Xu. On element-connectivity preserving graph simplification. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 313–324. Springer, 2015.

12 Chandra Chekuri and Chao Xu. Minimum cuts and sparsification in hypergraphs. *SIAM Journal on Computing*, 47(6):2118–2156, 2018.

13 Joseph Cheriyan and Mohammad R Salavatipour. Packing element-disjoint steiner trees. *ACM Transactions on Algorithms (TALG)*, 3(4):47–es, 2007.

14 Joseph Cheriyan, Santosh Vempala, and Adrian Vetta. Network design via iterative rounding of setpair relaxations. *Combinatorica*, 26(3):255–275, 2006.

15 Julia Chuzhoy and Sanjeev Khanna. An $o(k^3 \log n)$-approximation algorithm for vertex-connectivity survivable network design. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 437–441. IEEE, 2009.

16 Lisa Fleischer, Kamal Jain, and David P Williamson. Iterative rounding 2-approximation algorithms for minimum-cost vertex connectivity problems. *Journal of Computer and System Sciences*, 72(5):838–867, 2006.

17 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 Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2046–2065. SIAM, 2020.

18 Kyle Fox, Debmalya Panigrahi, and Fred Zhang. Minimum cut and minimum k-cut in hypergraphs via branching contractions. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 881–896. SIAM, 2019.

19 Satoru Fujishige and Satoru Iwata. Bisubmodular function minimization. *SIAM J. Discret. Math.*, 19(4):1065–1073, 2005.

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

21 Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Minimum cut in $(m \log^2 n)$ time. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 57:1–57:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

22 Andrew V. Goldberg and Robert E. Tarjan. Finding minimum-cost circulations by successive approximation. *Math. Oper. Res.*, 15(3):430–466, 1990.

23 Anupam Gupta and Jochen Könemann. Approximation algorithms for network design: A survey. *Surveys in Operations Research and Management Science*, 16(1):3–20, 2011.

24 Jianxiu Hao and James B. Orlin. A faster algorithm for finding the minimum cut in a directed graph. *J. Algorithms*, 17(3):424–446, 1994.

25 Ramesh Hariharan, Telikepalli Kavitha, Debmalya Panigrahi, and Anand Bhalgat. An $\tilde{O}(mn)$ gomory-hu tree construction algorithm for unweighted graphs. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 605–614, 2007.

26 Monika Rauch Henzinger. A static 2-approximation algorithm for vertex connectivity and incremental approximation algorithms for edge and vertex connectivity. *Journal of Algorithms*, 24(1):194–220, 1997.

27 Monika Rauch Henzinger, Satish Rao, and Harold N. Gabow. Computing vertex connectivity: New bounds from old techniques. *J. Algorithms*, 34(2):222–250, 2000.

28 Kamal Jain, Ion Măndoiu, Vijay V Vazirani, and David P Williamson. A primal–dual schema based approximation algorithm for the element connectivity problem. *Journal of Algorithms*, 45(1):1–15, 2002.

29 Stephen Jue and Philip N. Klein. A near-linear time minimum steiner cut algorithm for planar graphs, 2019. `arXiv:1912.11103`.

30 David R Karger. Minimum cuts in near-linear time. *Journal of the ACM (JACM)*, 47(1):46–76, 2000.

31 Tarun Kathuria, Yang P. Liu, and Aaron Sidford. Unit capacity maxflow in almost $o(m^{4/3})$ time. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 119–130. IEEE, 2020.

32 Regina Klimmek and Frank Wagner. A simple hypergraph min cut algorithm. Technical Report B 96-02, Bericht FU Berlin Fachbereich Mathematik und Informatik, 1996.

33 Guy Kortsarz and Zeev Nutov. Approximating minimum cost connectivity problems. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2010.

34 Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $\tilde{O}(\sqrt{\text{rank}})$ iterations and faster algorithms for maximum flow. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 424–433. IEEE Computer Society, 2014.

35 Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1049–1065. IEEE, 2015.

36 Jason Li, Danupon Nanongkai, Debmalya Panigrahi, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Vertex connectivity in poly-logarithmic max-flows, 2021. To appear in ACM STOC,.

37 Jason Li and Debmalya Panigrahi. Deterministic min-cut in poly-logarithmic max-flows. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 85–92. IEEE, 2020.

38 Nathan Linial, László Lovász, and Avi Wigderson. Rubber bands, convex embeddings and graph connectivity. *Comb.*, 8(1):91–102, 1988.

39 Yang P. Liu and Aaron Sidford. Faster energy maximization for faster maximum flow. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 803–814. ACM, 2020.

40 Aleksander Madry. Computing maximum flow with augmenting electrical flows. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 593–602. IEEE Computer Society, 2016.

41 Wai-Kei Mak and D.F. Wong. A fast hypergraph min-cut algorithm for circuit partitioning. *Integration, the VLSI Journal*, 30(1):1–11, 2000.

42 Sagnik Mukhopadhyay and Danupon Nanongkai. Weighted min-cut: sequential, cut-query, and streaming algorithms. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 496–509. ACM, 2020.

43 Sagnik Mukhopadhyay and Danupon Nanongkai. A note on isolating cut lemma for submodular function minimization, 2021. arXiv:2103.15724.

44 Hiroshi Nagamochi and Toshihide 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.

45 Danupon Nanongkai, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Breaking quadratic time for small vertex connectivity and an approximation scheme. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 241–252, 2019.

46 Kent Quanrud. Fast approximations for rooted connectivity in weighted directed graphs. *CoRR*, abs/2104.06933, 2021. arXiv:2104.06933.

47 Maurice Queyranne. Minimizing symmetric submodular functions. *Mathematical Programming*, 82(1-2):3–12, 1998.

48 Alexander Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.

49 Mechthild Stoer and Frank Wagner. A simple min-cut algorithm. *Journal of the ACM (JACM)*, 44(4):585–591, 1997.

50 Jan van den Brand, Yin Tat Lee, Yang P. Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Minimum cost flows, mdps, and ℓ_1-regression in nearly linear time for dense instances. *CoRR*, abs/2101.05719, 2021. arXiv:2101.05719.

51 Jan van den Brand, Yin Tat Lee, Danupon Nanongkai, Richard Peng, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Bipartite matching in nearly-linear time on moderately dense graphs. *CoRR*, abs/2009.01802, 2020. arXiv:2009.01802.