Data structure for node connectivity queries

Zeev Nutov
The Open University of Israel
nutov@openu.ac.il

Abstract

Let \( \kappa(s, t) \) denote the maximum number of internally disjoint paths in an undirected graph \( G \). We consider designing a data structure that includes a list of cuts, and answers in \( O(1) \) time the following query: given \( s, t \in V \), determine whether \( \kappa(s, t) \leq k \), and if so, return a pointer to an \( st \)-cut of size \( \leq k \) in the list. A trivial data structure includes a list of \( n(n - 1)/2 \) cuts and requires \( \Theta(kn^2) \) space. We show that \( O(kn) \) cuts suffice, thus reducing the space to \( O(k^2n + n^2) \). In the case when \( G \) is \( k \)-connected, we show that \( O(n) \) cuts suffice, and that these cuts can be partitioned into \( O(k) \) laminar families; this reduces the space to \( O(kn) \). The latter result slightly improves and substantially simplifies a recent result of Pettie and Yin [6].

1 Introduction

Let \( \kappa(s, t) = \kappa_G(s, t) \) denote the maximum number of internally disjoint \( st \)-paths in an undirected graph \( G = (V, E) \). An \( st \)-cut is a subset \( C \subseteq V \cup E \) such that \( G \setminus C \) has no \( st \)-path. By Menger’s Theorem, \( \kappa(s, t) \) equals to the minimum size of an \( st \)-cut, and there always exists a minimum \( st \)-cut that contains no edge except of \( st \). We consider designing a compact data structure that given \( s, t \in V \) answers the following queries.

- \( \text{CON}_k(s, t) \) (connectivity query): Determine whether \( \kappa(s, t) \geq k + 1 \).
- \( \text{CUT}_k(s, t) \) (cut query): If \( \kappa(s, t) \leq k \) then return an \( st \)-cut of size \( \leq k \).

The query \( \text{CUT}_k(s, t) \) requires \( \Theta(k) \) time just to write an \( st \)-cut. However, by slightly relaxing the definition, we allow the data structure to include a list of cuts, and to return just a pointer to an \( st \)-cut of size \( \leq k \) in the list. How short can this list be? By choosing a minimum \( st \)-cut for each pair \( \{s, t\} \), one gets a list of \( n(n - 1)/2 \) cuts. This gives a trivial data structure, that answers both queries in \( O(1) \) time, but requires \( \Theta(kn^2) \) space. For edge connectivity, the Gomory-Hu Cut-Tree [2] shows that there exists such a list of \( n - 1 \) cuts that form a laminar family. However, no similar result is known for the node connectivity case considered here.

Hsu and Lu [3] gave an \( O(kn) \) space\(^1\) data structure that determines whether \( \kappa(s, t) \geq k + 1 \) in \( O(1) \) time. They proved that there exists an auxiliary directed graph \( H = (V, F) \) and an ordered partition \( K_1, K_2, \ldots \) of \( V \), such that every \( K_i \) has at most \( 2k - 1 \) neighbors in \( H \), and all of them are in \( K_{i+1} \cup K_{i+2} \cup \cdots \). Then \( \kappa(s, t) \geq k + 1 \) iff \( s, t \) belong to the same part, or \( st \in F \), or \( ts \in F \). Since the outdegree of every node in \( H \) is at most \( 2k - 1 \), this data structure can be implemented using \( O(kn) \) space. They also gave a simple polynomial time algorithm for constructing such \( H \). A collection of data structures as in Theorem 2 for each \( k' = 1, \ldots, k \) enables to find \( \min\{\kappa(s, t), k + 1\} \) in \( O(log k) \) time, using binary search. However, their data structure cannot answer \( \text{CUT}_k(s, t) \) queries.

\(^1\) As in previous works, we ignore the unavoidable \( O(log n) \) factor invoked by storing the indexes of nodes.
A graph is \( k \)-connected if \( \kappa(s,t) \geq k \) for all \( s,t \in V \). Recently, Pettie and Yin \([6]\), and earlier in the 90’s Cohen, Di Battista, Kanevsky, and Tamassia \([1]\), considered the above problem in \( k \)-connected graphs. Pettie and Yin \([6]\) suggested for \( n \geq 4k \) an \( O(kn) \) space data structure, that answers \( \text{CON}_k(s,t) \) in \( O(1) \) time and \( \text{CUT}_k(s,t) \) in \( O(k) \) time; they showed that it can be constructed in \( \tilde{O}(m + \text{poly}(k)n) \) time. The data structure and the arguments in \([6]\) are complex, and here by simple proof we obtain the following slightly better result.

**Theorem 1.** For any \( k \)-connected graph, there exists an \( O(kn) \) space data structure, that includes a list of \( O(n) \) cuts, and answers each of \( \text{CON}_k(s,t) \) and \( \text{CUT}_k(s,t) \) in \( O(1) \) time.

We will not discuss the construction time of our data structure, but since it is substantially simpler than that of Pettie and Yin \([6]\), the construction time is not expected to be high.

For arbitrary graphs, a trivial data structure that answers \( \text{CON}_k(s,t) \) and \( \text{CUT}_k(s,t) \) queries in \( O(1) \) time uses \( \Theta(kn^2) \) space. We slightly improve on this as follows.

**Theorem 2.** There exists an \( O(k^2n + n^2) \) space data structure that includes a list of \( O(kn) \) cuts, that answers \( \text{CON}_k(s,t) \) and \( \text{CUT}_k(s,t) \) in \( O(1) \) time.

2 \( k \)-connected graphs (Theorem \([1]\))

In this section let \( G = (V,E) \) be a \( k \)-connected graph. We first explain how to answer the queries for pairs \( s,t \) with \( st \in E \). Let \( K \) denote the set of nodes of degree \( k \) in \( G \). If \( s \in K \) then the set of edges incident to \( s \) is a minimum \( st \)-cut for all \( t \). There are \( |K| \leq n \) such minimum cuts. This situation can be recognized in \( O(1) \) time, hence we may omit such pairs from our analysis and assume that each of \( s,t \) has degree \( \geq k + 1 \).

We say that \( st \in E \) is a **critical edge** if \( \kappa(s,t) = k \). Let \( F \) be the set of critical edges \( st \in E \) such that \( s,t \in V \setminus K \). Mader’s Critical Cycle Theorem \([\text{Mader}]\) states that any cycle of critical edges contains a node of degree \( k \), hence \( F \) is a forest. Thus just specifying the edges in \( F \) and a list of \( |F| \leq n - |K| - 1 \) minimum \( st \)-cuts for every \( st \in F \), gives an \( O(kn) \) space data structure that answers the relevant queries in \( O(1) \) time.

Henceforth assume that \( s,t \in V \setminus K \) and \( st \notin E \). We will show that then there exists a list of \( n - |K| \) cuts, such that whenever \( \kappa(s,t) = k \), there exists a minimum \( st \)-cut in the list. However, this is not enough to answer the relevant queries in \( O(1) \) time, since we still need to choose the right minimum cut from the list.

For a node subset \( A \subseteq V \) let \( \partial A \) denote the set of neighbors of \( A \) in \( G \). For \( A \subseteq V \) let \( A^* = V \setminus (A \cup \partial A) \) denote the "node complement" of \( A \). We say that \( A \) is: a **tight set** if \( |\partial A| = k \) and \( A^* \neq \emptyset \), an **st-set** if \( s \in A \) and \( t \in A^* \), and a **small set** if \( |A| \leq \frac{n-k}{2} \). Note that \( A \) is tight if and only if \( \partial A \) is a minimum cut, and \( A \) is a union of some, but not all, connected components of \( G \setminus C \). The following statement is a folklore.

**Lemma 3.** Let \( A,B \) be tight sets. If the sets \( A \cap B^*, B \cap A^* \) are both nonempty then they are both tight. If \( A,B \) are small and \( A \cap B \neq \emptyset \) then \( A \cap B \) is tight.

Let \( S = \{ s \in V \setminus K : \text{there exist a small tight set containing } s \} \). For \( s \in S \) let \( R_s \) be the (unique, by Lemma 3) inclusion minimal small tight set that contains \( s \). Let \( \mathcal{R} = \{ R_s : s \in S \} \). The following lemma shows that the family \( \{ \partial R : R \in \mathcal{R} \} \) is a "short" list of \( n - |K| \) minimum cuts, that for every \( s,t \in V \setminus K \) with \( st \notin E \) includes a minimum \( st \)-cut.

**Lemma 4.** Let \( s,t \in V \setminus K \) with \( st \notin E \). Then \( \kappa(s,t) = k \) if an only if (i) \( s \in S \) and \( R_s \) is an st-set, or (ii) \( t \in S \) and \( R_t \) is a t-set. Consequently, \( \kappa(s,t) = k \) if and only if the family \( \{ \partial R : R \in \mathcal{R} \} \) contains a minimum st-cut.
Proof. If (i) holds then \( \partial R_s \) is a minimum \( st \)-cut, while if (ii) holds then \( \partial R_t \) is a minimum \( st \)-cut. Thus \( \kappa(s,t) = k \) if (i) or (ii) holds.

Assume now that \( \kappa(s,t) = k \) and we will show that (i) or (ii) holds. Let \( C \subseteq V \) be a minimum \( st \)-cut. Then one component \( A \) of \( G \setminus C \) contains \( s \) and the other \( B \) contains \( t \). Since \( |A| + |B| \leq n - |C| = n - k \), one of \( A, B \), say \( A \) is small. Thus \( s \in S \). Since \( R_s \subseteq A \) and since \( t \in A^* \), we have \( t \in R_s^* \). Consequently, \( \partial R_s \) is a minimum \( st \)-cut, as required. \(\blacktriangleleft\)

Two sets \( A, B \) are \textit{laminar} if they are disjoint or one of them contains the other. A set family is laminar if its members are pairwise laminar. A laminar family \( \mathcal{L} \) can be represented by a rooted tree \( T = (\mathcal{L} \cup \{V\}, J) \) and a mapping \( \psi : V \rightarrow \mathcal{L} \cup \{V\} \). Here \( B \) is a child of \( A \) in \( T \) if \( B \) is a maximal set in \( \mathcal{L} \) distinct from \( A \), and \( \psi(v) \) is a minimal set in \( \mathcal{L} \) that contains \( v \).

\(\blacktriangleright\) **Lemma 5.** \( \mathcal{R} \) can be partitioned in polynomial time into at most \( 2k + 1 \) laminar families.

Proof. Consider two sets \( A = R_a \) and \( B = R_b \) that are not laminar. Then \( a \notin A \cap B \) since otherwise by Lemma 2, \( A \cap B \) is a tight set that contains \( a \), contradicting the minimality of \( A = R_a \). By a similar argument, \( b \notin A \cap B \). We also cannot have both \( a \in A \cap B^* \) and \( b \in B \cap A^* \), as then by Lemma 2, \( A \cap B^* \) is a tight set that contains \( a \), contradicting the minimality of \( A = R_a \). Consequently, \( a \notin \partial B \) or \( b \notin \partial A \).

Construct an auxiliary directed graph \( H \) on node set \( S \) and edges set \{\( ab : a \in \partial R_b \)\}. The indegree of every node in \( H \) is at most \( k \). This implies that every subgraph of the underlying graph of \( H \) has a node of degree \( 2k \). A graph is \( d \)-degenerate if every subgraph of it has a node of degree \( d \). It is known that any \( d \)-degenerate graph can be colored with \( d + 1 \) colors, in polynomial time. Hence \( H \) is \( (2k + 1) \)-colorable. Consequently, we can compute in polynomial time a partition of \( S \) into at most \( 2k + 1 \) independent sets. For each independent set \( S' \), the family \{\( R_s : s \in S' \)\} is laminar. \(\blacktriangleleft\)

Our data structure for pairs \( s, t \in V \setminus K \) with \( st \notin E \) consists of:

- A family \( \mathcal{T} \) of at most \( 2k + 1 \) trees, where each tree \( T \in \mathcal{T} \) with a mapping \( \psi_T : V \rightarrow V(T) \) represents one of the at most \( 2k + 1 \) laminar families of tight sets as in Lemma 5, the total number of edges in all trees in \( \mathcal{T} \) is at most \( n - |K| \).
- For each tree \( T \in \mathcal{T} \), a linear space data structure that answers ancestor/descendant queries in \( O(1) \) time. This can be done by assigning to each node of \( T \) the in-time and the out-time in a DFS search on \( T \).
- A list \{\( \partial R_s : s \in S \)\} of \( |S| = n - |K| \) minimum cuts.

For every \( s \in S \) let \( T_s \) be the (unique) tree in \( \mathcal{T} \) where \( R_s \) is represented. The next statement, that is a direct consequence of Lemma 4, specifies how we answer the queries.

\(\blacktriangleright\) **Lemma 6.** Let \( s, t \in V \setminus K \) with \( st \notin E \).

(i) If in \( T_s \), \( \psi_{T_s}(t) \) is not a descendant of \( \psi_{T_s}(s) \) and \( t \notin \partial R_s \), then \( \partial R_s \) is a minimum \( st \)-cut.

(ii) If in \( T_t \), \( \psi_{T_t}(s) \) is not a descendant of \( \psi_{T_t}(t) \) and \( s \notin \partial R_t \), then \( \partial R_t \) is a minimum \( st \)-cut. Furthermore, if none of (i),(ii) holds then \( \kappa(s,t) \geq k + 1 \).

It is easy to see that with appropriate pointers, we get an \( O(kn) \) space data structure that checks the two conditions in Lemma 6 in \( O(1) \) time. If one of the conditions is satisfied, the data structure return a pointer to one of \( \partial R_s \) or \( \partial R_t \). Else, it reports that \( \kappa(s,t) \geq k + 1 \).
3 Arbitrary graphs (Theorem 2)

To avoid dealing with mixed cuts that contain both nodes and edges, we first explain how to answer the queries for pairs \( s, t \) with \( st \in E \). Nagamochi and Ibaraki [5] proved that every undirected graph has a spanning subgraph with at most \((k + 1)n\) edges, that for all \( s, t \in V \) with \( \kappa(s, t) \leq k \) has the same minimum \( k \)-cuts as \( G \), and that such a subgraph can be computed in linear time. Consequently, we may assume that \(|E| \leq (k + 1)\). Thus just providing a list of \(|E| \leq n(k + 1)\) minimum \( k \)-cuts for every \( st \in E \), gives an \( O(k^2 n) \) space data structure that answers the relevant queries in \( O(1) \) time whenever \( st \in E \).

Henceforth assume that \( st \notin E \). Here we will say that \( A \) is an \( st \)-tight set if \( A \) is an \( st \)-set and \( \partial A = \kappa(s, t) \). Then there exists an \( st \)-tight set \( A \) such that \( \partial A \) is a minimum \( st \)-cut. We will need the following “uncrossing” lemma.

Lemma 7. Let \( A \) be \( s \)-tightly and \( B \) \( sb \)-tightly, where \( \kappa(s, a) \geq \kappa(s, b) \). Then exactly one of the following holds.

(i) \( a \in A^* \cap B^* \), \( \kappa(s, a) = \kappa(s, b) \), \( A \cup B, \{ s, A \} \) are both \( sa \)-tight, and \( A \cap B \) is a \( sb \)-tight.

(ii) \( a \notin A^* \cap B^* \) or \( b \notin A^* \cap B^* \), \( A \cap B \) is \( sa \)-tight, and \( A \cup B \) \( sb \)-tight.

(iii) \( a \in \partial B \) and \( b \in A^* \cap B^* \), or \( b \in \partial A \) and \( a \in A^* \cap B^* \), or \( a \in \partial A \) and \( b \in \partial A \).

Proof. If \( a \in A^* \cap B^* \) (see Fig. 1(a)), then (for any location of \( b \in B^* \)) both \( A \cap B, A \cup B \) are \( sa \)-sets, and \( A \cap B \) is an \( sb \)-set. Therefore

\[ \kappa(s, a) + \kappa(s, a) \geq \kappa(s, a) + \kappa(s, b) = \kappa(A \cap B) + \kappa(A \cup B) \geq \kappa(s, a) + \kappa(s, a) . \]

Thus equality holds everywhere, and (ia) follows.

If \( a \notin A^* \cap B^* \) and \( b \in A^* \cap B^* \) (see Fig. 1(b)), then \( A \cap B \) an \( sa \)-set and \( A \cup B \) an \( sb \)-set. Therefore

\[ \kappa(s, a) + \kappa(s, b) = \kappa(A \cap B) + \kappa(A \cup B) \geq \kappa(s, a) + \kappa(s, b) . \]

Thus equality holds everywhere, and (ib) follows.

If \( a \in A^* \cap B \) and \( b \in A^* \cap A \) (see Fig. 1(c)), then \( A^* \cap B \) is \( as \)-set and \( B^* \cap A \) is a \( bs \)-set. Therefore

\[ \kappa(s, a) + \kappa(s, b) = \kappa(A \cap B) + \kappa(A \cup B) \geq \kappa(s, a) + \kappa(s, b) . \]

Thus equality holds everywhere, and (ii) follows.

The remaining case is (iii), see Fig. 1(d,e). Since all the cases are exclusive, the lemma follows.

It is known that if \( A, B \) are both \( st \)-tight then so are \( A \cap B, A \cup B \). Let \( R_{st} \) denote the inclusion minimal (unique) \( st \)-tight set. From Lemma 7 we get the following.

Corollary 8. Let \( A = R_{sa} \) and \( B = R_{sb} \), where \( \kappa(s, a) \geq \kappa(s, b) \). Then one of the following holds: (i) \( A \subseteq B \); (ii) \( R_{sa} \subseteq B \) and \( R_{sb} \subseteq A \); (iii) \( a \in \partial B \) or \( b \in \partial A \).

Proof. Consider the four cases in Lemma 7. If (i) or (ii) holds, then \( A = A \cap B \), by the minimality of \( A = R_{sa} \). If (iii) holds, then \( R_{sa} \subseteq A \cap B \) and \( R_{sb} \subseteq B \cap A \). If (iii) holds, then \( a \in \partial B \) or \( b \in \partial A \).

Fix some \( s \in V \). Let \( T = \{ t \in V : st \notin E, \kappa(s, t) \leq k \} \). Let \( \mathcal{R}_s \) be the family of inclusion minimal members in \( \{ R_{st} : t \in T, |R_{st}| \leq |R_{st}| \} \). Let \( \mathcal{R} = \cup_{s \in V} \mathcal{R}_s \). Note that \( \mathcal{R} \) includes a minimum \( st \)-cut for every \( s, t \in V \) with \( st \notin E \). We will show that \( |\mathcal{R}| \leq 2k + 1 \), which implies \( |\mathcal{R}| \leq (2k + 1)n \).
Lemma 9. $|R_s| \leq 2k + 1$ for all $s \in V$.

Proof. Consider the cases in Corollary for distinct sets in $A = R_{sa}$ and $B = R_{sb}$ in $R_s$, where $|R_{sa}| \leq |R_{ss}|$, $|R_{sb}| \leq |R_{bs}|$, and $\kappa(s, a) \geq \kappa(s, b)$. Case (i) is not possible by the minimality of $A$. Case (ii) is also not possible; otherwise, $|R_{bs}| < |R_{sa}|$ and $|R_{as}| < |R_{sb}|$, and we get the contradiction $|R_{sa}| + |R_{sb}| > |R_{bs}| + |R_{as}|$. Thus case (iii) holds, namely, $a \in \partial B$ or $b \in \partial A$.

Construct an auxiliary directed graph $H$ on node set $T$ and edge set $\{ab : a \in \partial R_{sb}\}$. Note that if $H$ has no edge between $a$ and $b$ then $R_{sa} = R_{sb}$. The indegree of every node in $H$ is at most $k$. Thus by the same argument as in Lemma we obtain that the underlying graph of $H$ is $(2k + 1)$-colorable, and thus $T$ can be partitioned into at most $2k + 1$ independent sets. For each independent set $T'$, the family $\{R_{st} : t \in T'\}$ consists of a single set.

The data structure as in Theorem will include a $V \times V$ matrix, a list of $O(\kappa n)$ cuts, and for each pair $(s, t)$ with $\kappa(s, t) \leq k$, a pointer from each matrix entry $(s, t)$ to an $st$-cut in the list.

References

1. R. F. Cohen, G. Di Battista, A. Kanevsky, and R. Tamassia. Reinventing the wheel: an optimal data structure for connectivity queries. In STOC, pages 194–200, 1993.
2. R. E. Gomory and T. C. Hu. Multi-terminal network flows. Journal of the Society for Industrial and Applied Mathematics, 9, 1961.
3. T-H. Hsu and H-I. Lu. An optimal labeling for node connectivity. In ISAAC, pages 303–310, 2009.
4. W. Mader. Ecken von grad $n$ in minimalen $n$-fach zusammenhängenden graphen. Archive der Mathematik, 23:219–224, 1972.
5. 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.
6. S. Pettie and L. Yin. The structure of minimum vertex cuts. In ICALP, pages 105:1–105:20, 2021.