

Fast Edge Orientation for Unweighted Graphs

Anand Bhalgat*

Ramesh Hariharan†

Abstract

We consider an unweighted undirected graph with n vertices, m edges, and edge-connectivity $2k$. The *weak edge orientation problem* requires that the edges of this graph be oriented so the resulting directed graph is at least k edge-connected. Nash-Williams proved the existence of such orientations and subsequently Frank [6], Gabow [7], and Nagamochi-Ibaraki [12] gave algorithmic constructions. All of these algorithms took time at least quadratic in n . We provide the first sub-quadratic (in n) algorithm for this problem. Our algorithm takes $\tilde{O}(nk^4 + m)$ time. This improves the previous best bounds of $\tilde{O}(n^2k^2 + m)$ by Gabow [7] and $\tilde{O}(n^2m)$ by Nagamochi-Ibaraki [12] when $k \leq \sqrt{n}$. Indeed, many real networks have $k \ll n$.

Our algorithm uses the fast edge splitting paradigm introduced by Bhalgat et al. [2]. We seek to split out a large fraction of the vertices, recurse on the resulting graph, and then put back the split-off vertices. The main challenge we face is that only vertices with even degree may be split-off in an undirected graph and there may not be any such vertex in the current graph. The edge orientation algorithms of Gabow and Nagamochi-Ibaraki as well as Frank's proof are based on showing the existence of at least two even degree vertices (in fact, vertices with degree $2k$) in a $2k$ minimally connected graph. We generalize this to show that in any edge minimal $2k$ edge-connected graph, there are at least $n/3$ even degree vertices. These vertices are then split-off.

Our next challenge is to drop edges from the given graph so it remains $2k$ connected and yet has $\Omega(n)$ even degree vertices. We provide an algorithm that discards edges specifically to produce $\Omega(n)$ even degree vertices while maintaining connectivity $2k$ and takes time $\tilde{O}(nk^4 + m)$. Note that this algorithm does not necessarily make the graph edge-minimally $2k$ edge-connected. We also briefly outline an $\tilde{O}(nk^5 + m)$ time algorithm that achieves edge-minimality which improves the previous best bound of $\tilde{O}(m + n^2k^2)$ by Gabow [7].

*University of Pennsylvania. bhalgat@seas.upenn.edu. Work partly done when at Google Inc. and IISc.

†Strand Life Sciences and House of Algorithms, Bangalore. ramesh@strandls.com.

1 Introduction

We consider an unweighted undirected graph with n vertices, m edges, and edge-connectivity $2k$. The *weak edge orientation problem* requires that the edges of this graph be oriented so the resulting directed graph is at least k edge-connected. Nash-Williams [13] proved the existence of such orientations by showing the following theorem.

THEOREM 1.1. *Any $2k$ edge-connected undirected graph G can be oriented by giving direction to its edges such that in the resulting directed graph G' is at least k connected.*

Frank [6] gave a constructive proof of weak orientation theorem based on edge splitting. Subsequently, Gabow [7] and Nagamochi-Ibaraki [12] gave algorithms based on the Frank's proof. Gabow's algorithm takes time $\tilde{O}(n^2k^2 + m)$ and the algorithm of Nagamochi-Ibaraki takes time $\tilde{O}(n^2m)$. In many real world networks $k \ll n$ and that motivates the following question: can the edge orientation problem be solved in time near-linear in n , possibly at the expense of factors polynomial in k . We address this question by providing such an algorithm. Our algorithm takes time $\tilde{O}(nk^4 + m)$.

Previous approaches for a problem use edge-splitting to remove vertices from the graph; edges incident on these vertices are paired up resulting in a new graph with edge connectivity $2k$. The problem is then solved recursively. The time taken is quadratic because vertices are split-off one at a time. Our algorithm uses the fast edge splitting paradigm introduced by Bhalgat et al. [2]. We seek to split out a large fraction of the vertices, recurse on the resulting graph, and then put back the split-off vertices. The main challenge we face is that only vertices with even degree may be split-off completely in an undirected graph and there may not be any such vertex in the current graph. The edge orientation algorithms of Gabow and Nagamochi-Ibaraki as well as Frank's proof are based on showing the existence of at least two even degree vertices (in fact, vertices with degree $2k$) in a minimally $2k$ edge-connected graph. Here, a graph is said to be minimally $2k$ edge-connected if removing even a single edge causes the global min-cut to drop below $2k$. We generalize the above fact to show that in any minimally $2k$ edge-connected graph, there are at least $n/3$ even degree vertices. These vertices are then split-off.

Our next challenge is to drop edges from the given graph so it remains $2k$ edge-connected and yet has $\Omega(n)$ even

degree vertices. We provide an algorithm that discards edges specifically to produce $\Omega(n)$ even degree vertices while maintaining connectivity $2k$ and takes time $\tilde{O}(nk^4 + m)$. Note that this algorithm does not necessarily make the graph minimally $2k$ edge-connected. We also briefly outline an $\tilde{O}(nk^5 + m)$ time algorithm that achieves edge-minimality.

2 Preliminaries

This section outlines definitions and consolidates all previous results from literature that we will need. Let the input undirected graph $G = (V, E)$ have n vertices and m edges. Let $c(G) \geq 2k$ denote the global min-cut for G , where k is a positive integer. We will use the following theorems from literature.

THEOREM 2.1. [5] *Consider any directed graph with global min-cut $c(G)$ (if the graph is undirected then convert it to a directed graph by directing each edge in both directions). The number of edge disjoint arborescences rooted at any specified vertex r equals $c(G)$.*

THEOREM 2.2. [2] *For an undirected graph G with global min-cut $c(G)$ and any given positive integer $h \leq c(G)$, h edge disjoint arborescences as in Theorem 2.1 can be found in time $\tilde{O}(nh^3 + m)$.*

THEOREM 2.3. [11] *Given undirected graph G , all but upto $n * c(G)$ edges can be discarded in $O(n + m)$ time so the remaining graph continues to be $c(G)$ connected.*

We also use the edge splitting technique introduced by Lovász [9],[10] (exercise 6.53). Edge splitting in a graph involves removing two edges (a, b) and (b, c) and replacing them by a single edge (a, c) . The goal of this operation is to reduce the graph size while retaining certain connectivity properties thus serving as an important inductive and recursive tool for proving connectivity properties on the graph. When all edges incident on a vertex v have been paired up and replaced by their single counterparts as above, we say that vertex v has been *split-off*. Lovász showed that in an undirected graph with edge connectivity more than 2, any even degree vertex can be *split off* maintaining the global edge-connectivity. We will need the following constructive result to split-off many vertices simultaneously.

THEOREM 2.4. [2] *Suppose we are given undirected graph G with $c(G) > 1$, a subset W of even degree vertices in G , and a number $h \leq \alpha$, where α is the minimum of the pairwise connectivities of vertices outside W . Then vertices in W can be split-off in time $\tilde{O}(nh^2 + m)$ to obtain a new graph G' with $c(G') \geq h$.*

The $2k$ -partial Gomory-Hu Tree. Given an undirected graph G with $c(G) \geq 2k$, the $2k$ -partial Gomory-Hu tree T of G is defined as a tree satisfying the following two properties:

- The nodes of T represent a partition of the vertex set of G . The set of vertices associated with a particular node x of G is denoted by $v(x)$ and vertices in this set have pairwise edge-connectivity at least $2k + 1$.
- Each edge e of T represents a cut of size $2k$; this cut separates vertices of G associated with nodes on one side of e from vertices of G associated with nodes on the other side.

Note that T can be obtained from the full Gomory-Hu tree by compressing all edges with weight more than $2k$. Our algorithm uses these $2k$ -partial Gomory-Hu trees, which can be constructed using the following theorem from [1].

THEOREM 2.5. [1] *Given undirected graph G with $c(G) \geq 2k$, the $2k$ -partial Gomory-Hu tree can be constructed in time $\tilde{O}(nk^2 + m)$.*

3 Edge Orientation Algorithm Overview

The algorithm for edge orientation appear below in Algorithm 3.1. In essence, it splits-off a subset of vertices, recursively orients the resulting graph, and then puts back the split-off vertices. There are challenges in identifying vertices to be split-off because these need to have even degrees and also need to be independent.

Algorithm 3.1 Algorithm overview for the edge orientation problem.

1. Prune edges via Nagamochi-Ibaraki's algorithm so $|E| \leq 2kn$ and $c(G) \geq 2k$.
2. Prune edges so $c(G) \geq 2k$ and there are $\Theta(n)$ even degree vertices, each with degree $O(k)$. Let W denote the set of even degree vertices with degree $O(k)$.

repeat

3. Find a maximal independent set comprising only vertices in W which have not yet been split-off.
4. Split-off the above vertices while maintaining connectivity $2k$ for the remaining vertices.

until all vertices in W are split-off

5. Recursively solve edge orientation on the current graph; the boundary case is when there are only 2 vertices a, b in which case orient half the edges between a and b towards a , the other half towards b , and orient self-loops arbitrarily.

for each independent set split-off above, in reverse order of splitting-off **do**

6. Put back vertices in this independent set into the current graph and adjust orientations.

end for

Time Complexity. Step 1 takes $O(n + m)$ time by Theorem 2.3. Step 2 is our key contribution and is described in

Theorem 3.1 below. An existential proof of this theorem appears in Sections 4 and the constructive proof appears in 5.

THEOREM 3.1. *Given an undirected graph G with $c(G) \geq 2k$ and number of edges $O(nk)$, a subset of $O(n)$ edges can be identified in $\tilde{O}(nk^4)$ time; removal of these edges maintains the property that $c(G) \geq 2k$ but leaves the graph with $\Theta(n)$ even degree vertices, each with degree $O(k)$.*

For Step 3, by Turan's theorem, W has an independent set of size $\Omega(n/k)$; further, such an independent set can be found using a simple greedy algorithm in time $O(n)$. The repeat loop thus iterates $O(k \log n)$ times, and Step 3 takes $\tilde{O}(nk)$ time over all these iterations. By Theorem 2.4, Step 4 takes $\tilde{O}(nk^2)$ time per iteration, giving $\tilde{O}(nk^3)$ time over all iterations. Step 5 performs recursion on a graph that has only a constant fraction of the vertices of the original graph, and therefore the depth of recursion is $O(\log n)$. Step 6 performs only the following operation: given a directed edge (u, v) with one hidden split-off vertex b (since we split-off an independent set at a time, there cannot be more than one hidden split-off vertex inside an edge), we replace this edge with two directed edges (u, b) and (b, v) ; clearly this takes time $O(nk)$ over all iterations of the for loop. The overall time complexity is thus $\tilde{O}(nk^4)$, with Step 2 being the time bottleneck.

Correctness. The correctness of the algorithm comes from the following lemma.

LEMMA 3.1. *The resulting directed graph has global min-cut at least k .*

Proof. Consider the recursion in Step 5; the input to this step is an undirected graph with global min-cut at least $2k$. Inductively, assume that orientation on this undirected graph produces a directed graph H with min-cut at least k (in the boundary situation with two vertices, there must be at least $2k$ edges between the two vertices and orienting half of them in each direction clearly achieves this). It now suffices to show that when we put back an independent set X in Step 6, the resulting directed graph continues to have min-cut k . To show this, we use the arborescences of Theorem 2.1. So inductively assume that there exist two sets of arborescences, S_1 and S_2 , in H . S_1 comprises k edge disjoint arborescences directed away from root and S_2 comprises k edge disjoint arborescences directed into the root. We now need to put back vertices in X and show how S_1 and S_2 can each be modified to obtain new arborescence sets for H containing vertices from X as well. Note that this needs to be only an existential proof so time complexity is unimportant. We will show how S_1 can be modified, the modifications required for S_2 are identical.

After running step 6, we run into the following problems on the arborescences. First, a particular vertex $b \in X$ could appear many times in the same arborescence, and second, not all arborescences need have an occurrence of X . The first problem can be solved by a transformation which combines multiple occurrences of b in an arborescence; this is done by taking subtrees hanging off these multiple occurrences and hanging them off one chosen occurrence. This leaves at most one non-leaf occurrence of b in each arborescence. Leaf occurrences can be moved across arborescences because X is an independent set. With these transformations, we can ensure at most one occurrence of b per arborescence. It remains to consider the case when there are still arborescences without an occurrence of b . To handle this, note that b has degree at least $2k$ in the undirected graph just before it was split-off; after splitting-off at least k edges were created, each carrying b as a hidden vertex. These edges are now oriented and Step 6 creates at least k directed edges incident into and going out of b from these edges. If all of these are used in the arborescences then b must occur in every arborescence. Otherwise, the unused edges directed into b can be hung off as leaves in whichever arborescences they are missing in.

4 A Key Property

In this section, we show a key lemma which will contribute to the proof of Theorem 3.1.

THEOREM 4.1. *Any edge minimal undirected graph with global min-cut $2k$ has $n/3$ even degree vertices.*

Proof. Consider the partial $2k$ -Gomory-Hu tree T of G and note the following properties:

- Each resulting node in T is associated with one or more vertices of G and each vertex of G maps to exactly one node in T .
- Each edge in T is crossed by exactly $2k$ edges of G .
- No edge of G connects vertices within the same node in T by edge minimality of G ; so every edge in G crosses one or more of the edges of T .
- If $l \geq 2$ vertices of G are associated with the same node x in T , then each of these vertices has degree at least $2k + 1$ in G . By properties 2 and 3, it follows that x has degree at least $l + 1$ in T . It also follows that a degree 1 or 2 node in T must have exactly one associated vertex from G .

By property 4, degree 1 and 2 nodes in T have exactly one associated vertex from G each. It readily follows that vertices of G associated with leaves in T have even degree in G , in fact degree exactly $2k$. That vertices of G associated with a degree 2 node x in T have even degree as well can be

shown as follows: exactly $2k$ edges in G must cross each of the 2 edges incident on x in T and edges of G which cross both edges in T do not contribute to the degree of x in G . As a corollary, the number of even degree vertices in G is at least the number of degree 1 and 2 nodes in T . To complete the proof, we need to show that the number of degree 1 and 2 nodes in T is at least $n/2$.

To see this, note that by property 4, the total number of vertices associated with internal nodes in T is at most the number of edges in T , which is $|T| - 1$ where $|T|$ is the number of vertices in T . Or in other words, $n - |L| \leq |T| - 1$ where $|L|$ is the number of leaves in T , so $|T| + |L| \geq n + 1$. Since $|T| \leq |L| + (|L| - 1) + |D|$, where $|D|$ denotes the number of degree 2 vertices, we have $3|L| + |D| - 1 \geq n + 1$, so $|L| + |D| \geq |L| + |D|/3 \geq n/3 + 2/3$, as required.

Corollary 4.1 now provides an existential proof of Theorem 3.1; the constructive proof will appear in Section 5.

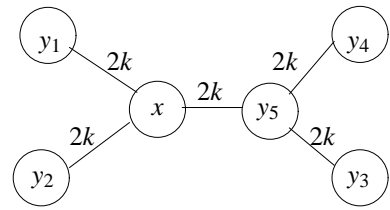
COROLLARY 4.1. *Given an undirected graph with global min-cut at least $2k$ and number of edges $O(nk)$, it is possible to remove edges so that the global min-cut continues to be at least $2k$ and there are $\Omega(n)$ even degree vertices, each with degree $O(k)$.*

Proof. By Theorem 4.1, removal of appropriate edges will result in $n/3$ even degree vertices while maintaining global min-cut at least $2k$. Since the total number of edges is $O(nk)$, a constant fraction of these even degree vertices must have degree $O(k)$.

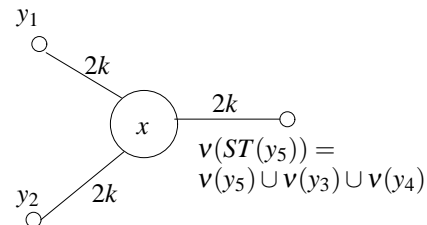
5 Algorithm Overview for Theorem 3.1

This section describes the framework of our algorithm for showing Theorem 3.1. Recall that we are given an undirected graph with global min-cut at least $2k$ and number of edges $O(nk)$ and we wish to identify a subset of $O(n)$ edges in $\tilde{O}(nk^4)$ time such that removal of these edges maintains the property that global min-cut is at least $2k$ but leaves the graph with $\Theta(n)$ even degree vertices, each with degree $O(k)$. Before explaining the algorithm, we introduce a concept of a *relevant graph* for a node in the $2k$ -partial Gomory-Hu tree.

The Relevant Graph for node x in the $2k$ -partial Gomory-Hu Tree. Let $y_1 \dots y_l$ denote the neighbours of x in Gomory-Hu tree T and let $ST(y_i)$ denote the subtree of T rooted at y_i . We extend the definition of $v(\cdot)$ to subtrees by defining $v(ST(y_i))$ as $\cup_{y \in ST(y_i)} v(y)$. We create a new graph G' by combining some vertices of G as follows: vertices of G present in $v(x)$ are retained as such in G' while vertices of G present in each $v(ST(y_i))$ are compressed together to yield l new vertices. Self-loops incident on these l vertices are then discarded. We call this graph G' the *relevant graph* for node x in T . The l vertices obtained by compressing vertices in $v(ST(y_1)) \dots v(ST(y_l))$ respectively are called *black*, while



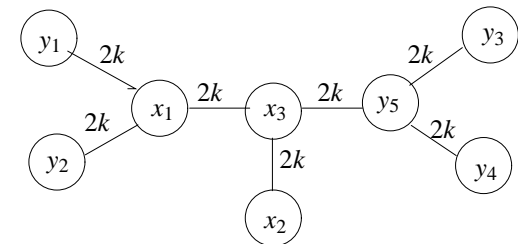
2k partial Gomory-Hu tree of G



Relevant graph G' for node x

Note : Each of $2k$ edges incident on y_i have other end point as $y_j, j \neq i$ or a vertex in x

Drop edges S such that both endpoints of edges in S are in $v(x)$ and G' is $2k$ connected after removal of S



Updated $2k$ partial Gomory-Hu tree of $G(V, E - S)$. x has been split into x_1, x_2, x_3

Figure 1: The Relevant Graph for a node in the $2k$ -partial Gomory-Hu tree and an overview of the algorithm for finding even degree vertices

the other vertices are called *white*. The number of vertices in G' is $|v(x)| + d_T(x)$ and the number of edges is at most $(d_G(v(x)) + d_T(x) * 2k)/2$, where $d_T(x)$ is the degree of node x in T and $d_G(v(x))$ is the sum of the degrees in G of vertices in $v(x)$. The relevant graph can be determined in linear time, i.e., in time $O((|v(x)| + d_T(x)) + d_G(v(x)) + d_T(x) * 2k)$.

Step 1 takes time $\tilde{O}(nk^2)$ using Theorem 2.5. Step 3 is the critical step; we show how to perform this step in Section 7; the time taken by this step for one iteration of the repeat loop will be $\tilde{O}(nk^3)$. Step 4 is straightforward. We describe how Step 5 can be performed below in Section 6; it will follow from Lemma 6.4 below that the time complexity of this step for one iteration of the repeat loop is $\tilde{O}(nk^2)$. The total time complexity now depends upon the time complexity

Algorithm 5.1 Algorithm Overview for Theorem 3.1.

-
0. Initialize R to the set of even degree vertices in G .
 1. Construct the $2k$ -partial Gomory-Hu tree T for G .
- repeat**
- for** each node x in T **do**
2. Obtain G' , the relevant graph for x as described in Section 2.
 3. Identify a set of edges S with both endpoints in $v(x)$ with the additional property that G' stays $2k$ edge-connected even after the removal of edges in S .
 4. Remove edges in S from G , and add vertices spanned by S to R .
- end for**
5. Update T to reflect the removal of edges above.
- until** $|R| > n/4$
-

of Step 2 and the number of iterations of the repeat loop. We claim the following lemma, whose proof will appear in Section 7; it follows that the total number of iterations of the repeat loop is $O(k \log n)$ and that the total time taken is therefore $\tilde{O}(nk^4)$, as required.

LEMMA 5.1. *Each iteration of the repeat loop increases $|T| + |R|$ by $\Omega(n/k)$ (where $|T|$ is the number of nodes in T). And if $|T| > n/4$ and $|R| < n/4$, the next iteration of the repeat loop causes either $|R|$ to become $\Omega(n)$ or alternatively, the number of even degree vertices becomes $\Omega(n)$.*

Finally, we need to show how to obtain $\Theta(n)$ even-degree vertices. If Lemma 5.1 terminates with $\Theta(n)$ even degree vertices then we are done. Otherwise we show that by removing only a subset and not all of the edges that got actually removed in the above algorithm, one can ensure $\Theta(n)$ even-degree vertices. To do this, restrict the graph to edges removed by the algorithm and find a spanning forest in this graph; this forest has $\Omega(n)$ vertices with non-zero degree (because $|R| = \Omega(n)$). In this spanning forest, find an independent set I of $\Omega(n)$ vertices with non-zero degree (such an independent set exists by Turan's theorem). For each vertex in I remove up to 1 edge so its degree becomes even. This results in $\Theta(n)$ even degree vertices. Since the total number of edges in the input graph is $O(nk)$, it follows that a constant fraction of these will have degree $O(k)$. This proves Theorem 3.1.

Two things remain. First the description of Step 5 for updating the $2k$ -partial tree, and second, the description of Step 3 for identifying S along with the proof of Lemma 5.1.

6 Updating the $2k$ -partial Gomory-Hu Tree T

Given a node x in T and a subset S of edges in G with both endpoints in $v(x)$, we show how T can be updated to reflect the removal of edges in S from G . Repeating this

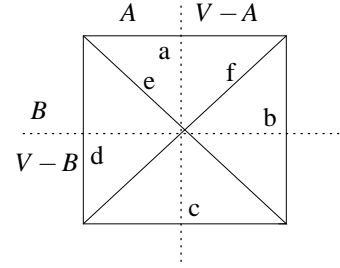


Figure 2: Submodularity

over all nodes x will yield the final updated tree. Recall that S satisfies the property that $c(G') \geq 2k$ even after removing edges in S . We need the following lemmas before describing the update process.

LEMMA 6.1. *In an undirected graph $H(V, E)$, consider a cut A of size z and another cut B of size less than z . Suppose none of $A \cap B, A \cap (V - B), (V - A) \cap B, (V - A) \cap (V - B)$ is empty. Then at least one of $A \cap B, A \cap (V - B), (V - A) \cap B, (V - A) \cap (V - B)$ is a cut of size less than z .*

Proof. Suppose each of the cuts $A \cap B, A \cap (V - B), (V - A) \cap B, (V - A) \cap (V - B)$ is of size at least z . Then from Figure 2, $2(a + b + c + d + e + f) \geq 4z$. Since the A is of size z and B is of size less than z , from Figure 2 $a + b + c + d + 2(e + f) < 2z$ which implies $(e + f) < 0$, a contradiction. Hence the proof.

LEMMA 6.2. *In an undirected graph $H(V, E)$, consider cuts A and B of size z . Suppose none of $A \cap B, A \cap (V - B), (V - A) \cap B, (V - A) \cap (V - B)$ is empty, and further, suppose H is z edge-connected. Then each of $A \cap B, A \cap (V - B), (V - A) \cap B, (V - A) \cap (V - B)$ is of size z .*

Proof. Suppose at least one of the cuts $A \cap B, A \cap (V - B), (V - A) \cap B, (V - A) \cap (V - B)$ has size more than z . Then since H is z connected, it follows from Figure 2 that $2(a + b + c + d + e + f) > 4z$. Since the A and B are of size z , $a + b + c + d + 2(e + f) = 2z$, which implies $(e + f) < 0$, a contradiction. Hence the proof.

LEMMA 6.3. *After edges in S are removed from G , $c(G) \geq 2k$; further, for all nodes $y \neq x$ of T , the pairwise connectivity between vertices in $v(y)$ continues to be least $2k + 1$.*

Proof. First, suppose $c(G) < 2k$ after removal of edges in S . The corresponding cut C must split $v(x)$ because edge in S have both endpoints in $v(x)$ and $c(G) \geq 2k$ before the removal of S . C must split at least one of the $v(ST(y_i))$'s as well, because $c(G') \geq 2k$ even after removal of S . Among all the cuts of size less than $2k$, select a cut C which splits the fewest number of $v(ST(y_i))$'s (black vertices). Let one of the $v(ST(y_i))$'s split by C be $v(ST(y_j))$. Now, by Lemma

6.1, at least one of $v(ST(y_j)) \cap C$, $(V(G) - v(ST(y_j))) \cap C$, $v(ST(y_j)) \cap (V(G) - C)$, $(V(G) - v(ST(y_j))) \cap (V(G) - C)$ should be a cut of size less than $2k$. Of these, $v(ST(y_j)) \cap C$ and $v(ST(y_j)) \cap (V(G) - C)$ cannot have size less than $2k$, because any cut of size less than $2k$ should split $v(x)$ as well. Also, $(V(G) - v(ST(y_j))) \cap C$ and $(V(G) - v(ST(y_j))) \cap (V(G) - C)$ split fewer $v(ST(y_i))$'s than C hence cannot be cuts of size less than $2k$, by the definition of C . Hence, after edges in S are removed from G , $c(G) \geq 2k$.

Second, suppose a new $2k$ cut splits $v(y)$ for some node $y \neq x$. Since both endpoints of an edge in S are in $v(x)$ the above cut must also split $v(x)$. Let C be such a cut. Now, by Lemma 6.2, cut $C \cap v(ST(y))$ is of size $2k$. Since C splits $v(y)$, cut $C \cap v(ST(y))$ separates two vertices of $v(y)$ whose connectivity before removing S was $2k + 1$. This is a contradiction since both endpoints of an edge in S are in $v(x)$.

LEMMA 6.4. *The $2k$ -partial Gomory-Hu tree T can be updated to reflect deletion of a specified edge set S comprising edges with both endpoints in $v(x)$ in time $\tilde{O}(|v(x)| + d_T(x)k^2 + d_G(v(x)) + d_T(x) * k)$, where $d_T(x)$ is the degree of node x in T and $d_G(v(x))$ is the sum of the degrees in G of vertices in $v(x)$.*

Proof. We describe the update algorithm. By Lemma 6.3, all nodes $y \neq x$ can be retained as such and only node x needs to be split further to identify the new $2k + 1$ edge-connected components, as follows. In the description below, G denotes the graph in which edges in S have been removed.

We now construct the $2k$ -partial Gomory-Hu tree on G' , the relevant graph for node x , using Theorem 2.5. The l black vertices in G' will correspond to leaf nodes in this new tree; we then put back the subtrees $ST(y_i)$ associated with each such leaf to get the final updated $2k$ -partial Gomory-Hu tree T' . The time complexity of this procedure follows from Theorem 2.5.

To show correctness, it suffices to show that:

- For each node y in T' , vertices in $v(y)$ are pairwise at least $2k + 1$ edge-connected in G .
- Each edge in T' represents a $2k$ cut in G .

First, for each node y in T' , we show that vertices in $v(y)$ are pairwise at least $2k + 1$ edge-connected in G . Let $x_1 \dots x_r$ be the nodes in T' obtained by splitting node x in T . All other nodes y in T' , $y \neq x_i$, $1 \leq i \leq r$, have corresponding nodes in T and $v(y)$ satisfies the $2k + 1$ edge-connectivity requirement by Lemma 6.3. For nodes $x_1 \dots x_r$, $v(x_i)$ satisfies this requirement in G' by construction. We need to show that this holds true for G as well. Suppose for a contradiction that there exists a $2k$ cut separating vertices in $v(x_i)$ in G , but not in G' . Then this cut must necessarily split one or more of the sets $v(ST(y_i))$, $1 \leq i \leq l$. Among all such

cuts of size $2k$ which split $v(x_i)$, and which exist in G but not in G' , select a cut C which splits the fewest $v(ST(y_i))$, $1 \leq i \leq l$. Let it split $v(ST(y_j))$. Since G is $2k$ connected and C and $v(ST(y_j))$ are cuts of size $2k$, by Lemma 6.2, the cut $(V(G) - v(ST(y_j))) \cap C$ is of size $2k$, splits $v(x_i)$, and violates the minimality in the definition of C . Hence the contradiction.

Second, we show that each edge in T' represents a $2k$ cut in G . This is clearly true for edges whose corresponding cuts do not split any $v(ST(y_i))$, $1 \leq i \leq l$, by virtue of construction on G' . It remains to consider edges which do split some $v(ST(y_i))$; the corresponding cut is present in T as well and since edges in S have both endpoints in $v(x)$, these cuts continue to have size $2k$.

7 Identifying edges for removal

Given a node x in T and G' the relevant graph for x , this section shows how to identify a subset S of edges in G' satisfying the following properties.

- Both endpoints of edges in S should be in $v(x)$.
- $c(G') \geq 2k$ even after the removal of edges in S .

We also prove Lemma 5.1 in this section. The algorithm for identifying S is given below in Algorithm 7.1.

Note that in Step 1, removal of a single edge will ensure $2k$ edge-connectedness of $v(x)$ because vertices in $v(x)$ are at least $2k + 1$ edge-connected to begin with. So consider Steps 2 through 8. Recall that the relevant graph has $|v(x)| + d_T(x)$ vertices and $(d_G(v(x)) + d_T(x) * 2k)/2$ edges (see Section 2). Step 1 takes time $O(|v(x)| + d_T(x) + d_G(v(x)) + d_T(x) * 2k)$. Step 2 takes time $\tilde{O}(|v(x)| + d_T(x)k^2 + d_G(v(x)) + d_T(x) * 2k)$ by Theorem 2.4. Step 3 takes time $\tilde{O}(|v(x)| + d_T(x))k^3 + d_G(v(x)) + d_T(x) * 2k$ by Theorem 2.2. Step 4, 5, 6 take $O(|v(x)| + d_T(x) * 2k)$ time. We will show Step 7 takes $\tilde{O}(|v(x)| + d_T(x))k^2 + d_G(v(x)) + d_T(x) * 2k$ time. Step 8 is straightforward. The total time taken amounts to $\tilde{O}(|v(x)| + d_T(x))k^3 + d_G(v(x)) + d_T(x) * k$ and is dominated by Step 3. Over all vertices x of T , this time adds up to $\tilde{O}(nk^3)$, as required. And vertices in S have both endpoints in $v(x)$ as required because only valid edges, i.e., those without any black endpoints, are candidates for addition to S .

We have the following lemma which will be needed in proving Lemma 5.1 later.

LEMMA 7.1. *Consider one iteration of the repeat loop in Algorithm 5.1 and all invocations of Algorithm 7.1 in this iteration (each iteration runs on a different node x of T). Also suppose $|T| < n/4$. Then, summed over all the above invocations, $|S| = \Omega(n/k)$ and edges in S_2 span $\Omega(n/k)$ vertices which do not belong to R .*

Proof. Note that the repeat loop iterates only as long as $|R| < n/4$. Since $|T| < n/4$, the number of invalid edges in

Algorithm 7.1 Identifying S for node x of T .

0. Set S to ϕ .
if $|T| > n/4$ **then**
 1. Add any one edge which has both endpoints white and which has both endpoints in $v(x)$ to S , if such an edge exists.
 else
 2. Split-off black vertices in G' retaining $2k+1$ edge-connectivity on the whites to obtain G'' . Denote edges which are a result of splitting-off as invalid; valid edges have no hidden black vertices and are present in both G'' and G' .
 3. Construct $2k+1$ Edmonds' arborescences on the directed version of G'' obtained by directing edges in both directions.
 4. Identify the arborescence A_1 which has the maximum number of valid directed edges incident on vertices outside R ; let S_1 denote the set of these edges.
 5. Identify the arborescence A_2 which has the maximum number of reverse edges for valid directed edges in S_1 .
 6. Consider the set of all valid undirected edges in G'' incident on vertices outside R with both forward and reverse versions in $A_1 \cup A_2$; let S_2 denote this set.
 7. Run a certification procedure to declare each edge in S_2 as redundant or essential.
 8. Add redundant edges from S_2 to S
 end if

G'' over all invocations in an iteration is at most $nk/2$. Over all invocations, the total number of edges in arborescences incident on vertices outside R is at least $3n(2k+1)/8$; of these there must be at least $3n(2k)/8 - nk/2 \geq nk/4$ valid edges. Of the $nk/4$ valid edges, a $\Theta(1/k^2)$ fraction gets added to S_2 in Steps 4, 5, and 6. The lemma follows.

Step 7: The Certification Procedure. Consider the graph G' with edges in S_2 removed. If the min-cut in G' is at least $2k$, then we declare all of S_2 as redundant. Otherwise, we have the following lemma, which is the basis for our certification algorithm described below.

LEMMA 7.2. *The global min-cut in G' with edges in S_2 removed is at least $2k-1$.*

Proof. At least $2k-1$ of the arborescences constructed for G'' in Step 3 are untouched by S_2 , i.e., neither forward nor reverse versions of edges in S_2 appear in these arborescences. It follows that G'' is $2k-1$ edge-connected even after removal of edges in S_2 . To show the corresponding result for G' , assume for a contradiction that G' with edges in S_2 removed has global min-cut less than $2k-1$, and consider such a $2k-2$ or smaller cut. This cut cannot split the whites, otherwise G'' with edges in S_2 removed would not be $2k-1$

connected (i.e., splitting blacks cannot increase connectivity). And cuts which have all whites on one side are unaffected by removal of S_2 because edges in S_2 contain only white vertices. The lemma follows.

The algorithm first removes edges in S_2 from G' and then obtains the cactus of all $2k-1$ cuts in G' . Since $2k-1$ is odd, the cactus is a tree [4]. Next it runs two passes. The first pass processes edges e in S_2 in arbitrary order. If e crosses edges in the cactus then the cactus is modified by compressing all such edges, otherwise e is marked as redundant. All cactus edges are reset by uncompressing next. The second pass processes edges e in S_2 in reverse first-pass order. Again, if e crosses edges in the cactus then the cactus is modified by compressing all such edges, otherwise e is marked as redundant. S comprises all edges marked redundant.

The time required to find all $(2k-1)$ -cuts and arrange these in a cactus is $\tilde{O}((|v(x)| + d_T(x))k^2 + d_G(v(x)) + d_T(x) * 2k)$ [1]. The time taken in the first and second passes for compressing cactus edges can be kept down to $O(\log n)$ per edge using centroid path decomposition, which amounts to $\tilde{O}(d_G(v(x)) + d_T(x) * k)$. We show correctness next.

LEMMA 7.3. *The global min-cut in G' with edges in S removed is at least $2k$.*

Proof. Since G' is $2k$ edge-connected, every edge in the cactus is crossed by some edge in S_2 . It suffices to show that every edge in the cactus is crossed by some edge in $S_2 - S$ as well. This is true by construction because an edge is marked redundant and added to S only if all the cactus edges it crosses are crossed by other edges not yet marked redundant at that instant.

It remain to prove Lemma 5.1.

LEMMA 7.4. *Each edge in $S_2 - S$ can be associated with a unique cactus edge, i.e., with a unique cut of size $2k$ in G' (with edges in S removed).*

Proof. At the end of the first pass, each edge in $S_2 - S$ crosses a cactus edge which no previous edge in $S_2 - S$ (i.e., in first pass order) crosses. At the end of the second pass, each edge in $S_2 - S$ crosses a cactus edge which no edge which follows later (i.e., in first pass order) crosses. The lemma follows.

Proof of Lemma 5.1. First, we show that each iteration of the repeat loop increases $|T| + |R|$ by $\Omega(n/k)$ (where $|T|$ is the number of vertices in T). This is done as follows. S_2 has size $\Omega(n/k)$ and spans $\Omega(n/k)$ vertices outside R by Lemma 7.1. By Lemma 7.4, each edge in $S_2 - S$ is associated with a unique new $2k$ cut in G' , i.e., a $2k$ cut created by the removal of edges in S . Each such $2k$ cut causes $|T|$ to increment by 1

when it is updated. If S spans $\Omega(n/k)$ vertices outside R then the lemma follows. Otherwise $S_2 - S$ spans $\Omega(n/k)$ vertices and therefore $|S_2 - S| = \Omega(n/k)$. The lemma follows.

Second, we need to show that if $|T| > n/4$ and $|R| < n/4$, the next iteration of the repeat loop causes either $|R|$ to become $\Omega(n)$ or alternatively, the number of even degree vertices becomes $\Omega(n)$. This is done as follows. Consider the next iteration of the repeat loop and take all invocations of Algorithm 7.1 in this iteration. Each invocation removes an arbitrary valid edge, if available, from the corresponding problem instance. If there are $\Omega(n)$ invocations corresponding to degree 1 or 2 nodes in T which have available valid edges then the lemma follows. Otherwise, there are $\Omega(n)$ invocations corresponding to degree 1 or 2 nodes in T which have no available valid edges; each of these invocations can be shown to have just one white vertex (if there are $x \geq 2$ white vertices, each must have degree at least $2k + 1$ because pairwise edge-connectivity for white vertices is $2k + 1$, and there must be $x(2k + 1)$ edges connecting these whites to the up to 2 available blacks, which is not possible since blacks have degree $2k$); these white vertices must therefore have even degree. The lemma follows.

8 Achieving Edge Minimality

The above algorithm can be modified to achieve edge minimality in $\tilde{O}(nk^5 + m)$ time by changing steps 4, 5 and 6 in Algorithm 7.1 to select S without regard to incidence on vertices in R . Each iteration of the repeat loop in Algorithm 5.1 then causes a $\Theta(1/k^2)$ fraction of the valid edges to be either declared as redundant or certified as necessary (as for edges in Lemma 7.4). The number of iterations of the repeat loop then increases to $O(k^2 \log n)$ from $O(k \log n)$ and the total time increases to $\tilde{O}(nk^5 + m)$.

9 Conclusion and Open problems

In this paper, we show that in an edge minimally $2k$ connected unweighted graph, there are at least $n/3$ even degree vertices. We also give first sub-quadratic in n algorithm for weak orientation problem. Our algorithm runs in time $\tilde{O}(m + nk^4)$ and improves previous best $\tilde{O}(m + n^2k^2)$ by Gabow [7]. We also give an edge minimalization algorithm which runs in time $\tilde{O}(m + nk^5)$ and improves previous best $\tilde{O}(m + n^2k^2)$ by Gabow [7]. This is the first algorithm for edge minimalization which does not check all edges one by one. One of the open questions is whether randomization can be used to find a large set of simultaneously droppable edges with large vertex span.

References

- [1] A. Bhalgat, R. Hariharan, T. Kavitha, D. Panigrahi. *An $\tilde{O}(mn)$ Gomory-Hu tree construction algorithm for unweighted graphs*, Proceedings of the thirty-ninth annual ACM symposium on Theory of computing, San Diego, pp. 605-614, 2007.
- [2] A. Bhalgat, R. Hariharan, T. Kavitha, D. Panigrahi. *Fast edge splitting and Edmond's arborescence construction for unweighted graphs*. Proceedings of the Nineteenth annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, pp. 455-464, 2008.
- [3] R. Cole, R. Hariharan. *A fast algorithm for computing steiner edge connectivity*, Proc. of the 35th Annual ACM Symposium on Theory of Computing, San Diego, pp. 167-176, 2003.
- [4] E.A. Dinitz, A.V. Karzanov, M.V. Lomonosov. *On the structure of a family of minimal weighted cuts in a graph*. Studies in Discrete Optimization, A.A. Fridman, Ed., pp. 240-306, 1976. (Original article in Russian, translation available from National Translation Center, Library of Congress, Cataloging Distribution Center, Washington D.C, 20541 (NTC 89-20265)).
- [5] J. Edmonds, *Submodular functions, matroids, and certain polyhedra*, Calgary International Conf. on Combinatorial Structures and their Applications, Gordon and Breach, New York, pp. 69-87, 1969.
- [6] A. Frank, *Applications of submodular functions*, in surveys in combinatorics. London Math Soc. Lecture note series 187, K. Walker, Ed., Cambridge University Press NY, 1993, pp. 86-136.
- [7] Harold N. Gabow, *Efficient splitting off algorithms for graphs*, Proc. of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, Montréal, Québec, Canada, pp. 696-705, 1994.
- [8] R. E. Gomory and T. C. Hu. *Multi-terminal network flows*, J. Soc. Indust. Appl. Math. 9(4), pp. 551-570, 1961.
- [9] L. Lovász, Lecture, Conference of Graph Theory, Prague, 1974.
- [10] L. Lovász, *Combinatorial Problems and Exercises*, 2nd Ed., North-Holland, New York, 1993.
- [11] Hiroshi Nagamochi and Toshihide Ibaraki, *A linear-time algorithm for finding a sparse k -connected spanning subgraph of a k -connected graph*, Algorithmica, 7, 1992, pp. 583-596.
- [12] Hiroshi Nagamochi and Toshihide Ibaraki, *Deterministic $\tilde{O}(mn)$ Time edge splitting in undirected graphs*, Proc. of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, Philadelphia, pp. 64-73, 1996.
- [13] C. St. J. A. Nash-Williams, *On orientations, connectivity and odd vertex pairings in finite graphs*, Canadian J. Math., 12, 1960, pp. 555-567.
- [14] W. Mader, *A reduction method for edge connectivity in graphs*, Ann. Discrete Math., 9, 1978, pp. 145-164.