Hardness of Cut Problems in Directed Graphs

[Extended Abstract]

Julia Chuzhoy
CSAIL MIT
and Dept. of CIS
University of Pennsylvania
cjulia@theory.csail.mit.edu

Sanjeev Khanna
Dept. of Comp. & Inf. Sci.
University of Pennsylvania
Philadelphia, PA 19104
sanjeev@cis.upenn.edu

ABSTRACT

We study the approximability of the multicut and the (non-bipartite) sparsest cut problems in directed graphs. In the multicut problem, we are a given a graph G along with k source-sink pairs, and the goal is to find a smallest subset of edges whose deletion separates all source-sink pairs. The sparsest cut problem has the same input, but the goal is to find a subset of edges to delete so as to minimize the ratio of deleted edges to the number of source-sink pairs that are separated by this deletion. Study of algorithms for cut problems is intimately connected to the dual notion of flows in networks, and many approximation algorithms for cut problems use a flow solution as a starting point. The best known approximation algorithm for directed multicut is based on this approach and gives an $O(\sqrt{n})$-approximation. On the other hand, the gap between the maximum multicommodity flow and the minimum multicut is known to be $\Omega(\min\{k, \log n\})$. While this flow-cut gap may be interpreted as an evidence of inherent difficulty in designing good approximation algorithms for directed multicut, the strongest hardness result known is an APX-hardness. Even assuming the Unique Games Conjecture, only an $\omega(1)$-hardness is known. Similar bounds hold for the directed sparsest cut problem.

Our main result is that directed multicut is $\Omega(\log n/\log\log n)$-hard to approximate unless $\mathsf{NP} \subseteq \mathsf{DTIME}\left(n^{\mathrm{polylog(n)}}\right)$. We show that this hardness result holds even when we allow a bicriteria relaxation, where the approximate solution is required to separate only a constant fraction of the pairs. This bicriteria hardness allows us to infer an $\Omega(\log n/\log\log n)$-hardness for the directed (non-bipartite) sparsest cut problem.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Algorithms, Theory.

Keywords

Directed Multicut, Sparsest cut, Hardness of Approximation.

1. INTRODUCTION

Cut problems are fundamental to combinatorial optimization and arise as intermediate problems in design of approximation algorithms for many graph problems. In this paper, we study the approximability of the directed multicut problem and the closely related (non-bipartite) directed sparsest cut problem. An instance of the *directed multicut* problem consists of a directed graph $G(V, E)$ and a collection of k source-sink pairs $\{(s_1, t_1), ..., (s_k, t_k)\}$. The goal is to delete the smallest possible number of edges so as to separate all source-sink pairs; a pair (s, t) is considered separated if in the resulting graph there is no path connecting s to t. The input to the *directed sparsest cut* problem is the same, but the objective now is to find a subset E' of edges so as to minimize the ratio $|E'|/|S_{E'}|$ where $S_{E'}$ is the set of (s, t) pairs, which are disconnected in the graph $G(V, E \setminus E')$. The parameter k is also referred to as the *number of commodities* in the instance. Vertices in the set $T = \{s_1, t_1, s_2, t_2, ..., s_k, t_k\}$ are referred to as *terminals.*

Directed Multicut: For the single-commodity case, the celebrated max-flow min-cut theorem [12] shows that the size of minimum (s_1, t_1) cut equals the maximum flow from s_1 to t_1. It is no surprise then, that our study of cut problems is intimately connected with the dual notion of flows, and that algorithms for cut problems often use a flow solution (poly-time computable) as a starting point. However, as we go beyond the single commodity, the tight duality between cuts and flows breaks down even in undirected graphs. The gap between maximum flow and the minimum multicut is well-understood in undirected graphs and is known to be $\Theta(\log k)$ [19, 13]. In a sharp contrast, Saks *et al.* [21] have shown that the flow-cut in directed graphs can be as large as $k - \epsilon$ for any $\epsilon > 0$ [21]. Since it is easy to see that the flow-cut gap cannot exceed k, it may seem that the flow-cut gaps are well-understood in the directed case as well. Unfortunately (or fortunately), the size of the Saks *et. al* construction grows exponentially in k, and the gap realized by these instances is only $O(\log n)$, where n is the number of vertices in G. As a function of n, this is the strongest gap known.

Our lack of understanding of the directed flow-cut gaps is reflected in the large separation between the upper and lower bounds on the approximability threshold of directed multicut. The best hardness known is an APX-hardness [9]; this hardness holds even on undirected star trees [14]. On the algorithmic front, until recently, no non-trivial approximation algorithm was known. Cheriyan, Karloff, and Rabani [8] gave the first non-trivial approximation algorithm for this problem. They achieve an approximation ratio of $O(\sqrt{n \log n})$ which was subsequently further improved by Gupta [15] to $O(\sqrt{n})$. Thus, until recently, our understanding of the directed multicut problem allowed its approximability threshold to be anywhere between a constant and a polynomial function. A recent result [7, 18] shows that, assuming the Unique Games Conjecture of Khot [17], the multicut problem is hard to approximate to within any constant factor even on undirected graphs.

Sparsest Cut: The notion of a sparsest cut in a directed graph can be defined in two distinct ways. In one version of the problem, which we refer to as the *bipartite sparsest cut*, sparsest cut in a graph is a bipartition of vertices into two sets S and $\bar{S}$ that minimizes the ratio of $|\delta(S, \bar{S})|$[1] to $|\{(s_i, t_i) \mid s_i \in S,\ t_i \in \bar{S}\}|$. In the second version, which we refer to as the *non-bipartite sparsest cut* or simply as *sparsest cut*, we relax the bipartition requirement and simply minimize the ratio of edges deleted to the resulting number of pairs separated. In undirected graphs, it is easily seen that the two notions are equivalent to within a factor of two. However, in directed graphs, as highlighted in the very recent work of Charikar *et al.* [6], these version seem to behave quite differently. In particular, using a result of Feige and Kogan [11], it is shown in [6] that bipartite sparsest cut is hard to approximate to within $2^{\Omega((\log n)^{\delta})}$ for some $\delta > 0$ unless 3SAT has subexponential-time algorithms. Furthermore, this hardness can be strengthened to an n^{δ}-hardness for some $\delta > 0$ assuming a hypothesis concerning hardness of random 3SAT, as described by Feige [10]. The strongest hardness known for the directed non-bipartite sparsest cut, in contrast, is an APX-hardness. The best known approximation ratio is an $O(\sqrt{n})$-approximation due to Hajiaghayi and Räcke [16]. The work of [7, 18] also shows that assuming the Unique Games Conjecture, the non-bipartite sparsest cut problem is hard to approximate to within any constant factor even on undirected graphs. We note that for the undirected version of sparsest cut it is easy to prove APX-hardness (without assuming the Unique Games Conjecture). We provide a simple proof in the Appendix. Thus the approximability status of directed multicut and directed non-bipartite sparsest cut is the same.

Our focus in this paper is stronger hardness of approximation results for directed multicut and directed non-bipartite sparsest cut.

Our Results: We make some progress towards closing the gaps in our understanding of the directed multicut and the sparsest cut problem. We show that directed multicut is $\Omega(\log n / \log \log n)$-hard to approximate unless $\mathsf{NP} \subseteq \mathsf{DTIME}\left(n^{\mathrm{polylog(n)}}\right)$. In particular, we show that this hardness holds even for the following bicriteria relaxation. A solution to a multicut instance is called an (α, β) bicriteria approximation for some $0 \le \alpha \le 1$ and $\beta \ge 1$, if it disconnects at least an α fraction of the pairs and deletes at most $\beta\mathsf{OPT}$ edges, where OPT denotes the cost of the optimal solution for the multicut instance. We show the following:

Theorem 1. *The directed multicut problem is* $\Omega\left(\frac{\log n}{\log \log n}\right)$*-hard to approximate, unless* $\mathsf{NP} \subseteq \mathsf{DTIME}\left(n^{\mathrm{polylog(n)}}\right)$*. Moreover, there is no* $\left(0.99, O\left(\frac{\log n}{\log \log n}\right)\right)$ *bicriteria approximation for directed multicut, under the same complexity assumption.*

Given Theorem 1, we can use an argument similar to the one given in [7] for undirected cut problems, to obtain the following result:

Theorem 2. *The directed (non-bipartite) sparsest cut problem is* $\Omega\left(\frac{\log n}{\log \log n}\right)$*-hard to approximate unless* NP *is contained in* $\mathsf{DTIME}\left(n^{\mathrm{polylog(n)}}\right)$.

For sake of completeness, we briefly sketch the proof of Theorem 2. Let $\alpha_0 = 0.99$ and let $\beta_0 = \Theta(\log n / \log \log n)$ be the hardness factor in Theorem 1. Let OPT denote the optimal value for a given multicut instance. Suppose we have a β'-approximation algorithm for sparsest cut. Then as long as at least $(1 - \alpha_0)k$ pairs remain to be separated, we can use the β'-approximation to find a subset E' of edges of size at most $\beta' \left(\frac{\mathsf{OPT}}{(1-\alpha_0)k}\right) p$ that separates at least p pairs for some integer $p \ge 1$. We delete the edges in E' and repeat this process until the number of remaining pairs falls below $(1 - \alpha_0)k$. It is easy to see that the total number of edges deleted over all iterations is at most $\left(\frac{\beta'}{(1-\alpha_0)}\right) \mathsf{OPT}$. Thus assuming Theorem 1, we must have $\beta' \ge (1 - \alpha_0)\beta_0 = \Omega(\frac{\log n}{\log \log n})$.

Organization: The remainder of this paper is devoted to proving Theorem 1. We start with some preliminaries in Section 2, and present our hardness construction for Theorem 1 in Section 3, which is based on a reduction from 3SAT. Our construction is based on the Raz verifier for 3SAT. In Sections 4 and 5, we present an analysis of the construction for the case when the 3SAT formula is satisfiable (a Yes-Instance) and non-satisfiable (a No-Instance), respectively.

2. PRELIMINARIES

For the sake of convenience, we consider the vertex version of the directed multicut problem. In this version, the input is the same as in directed multicut, and the goal is to remove a subset $\mathcal{S}$ of **non-terminal** vertices of smallest cardinality, that disconnects all the source-sink pairs. We prove Theorem 1 for the vertex version of the multicut problem. In order to obtain the same result for the multicut problem itself, we use a standard procedure to convert our instance to the instance of directed multicut, as follows. Each vertex v is replaced by a directed edge $(v^+ \to v^-)$. Each edge $e = (u \to v)$ in the original graph is replaced by an edge $(u^- \to v^+)$. We can assume w.l.o.g., that any optimal solution of the new instance only contains edges of the form $(v^+ \to v^-)$, and thus solution costs for both instances are the same.

The reduction is performed from the gap version of Max 3SAT(5). The input to the problem is a CNF formula φ

[1] $\delta(S, \bar{S})$ refers to all edges (x, y) in G where $x \in S$ and $y \in \bar{S}$.

with n variables and $\frac{5n}{3}$ clauses. Each clause consists of 3 literals and each variable participates in 5 clauses, appearing in each clause at most once. Let $\epsilon : 0 < \epsilon < 1$ be a constant, and let φ be an instance of Max 3SAT(5). Then φ is called a YES-INSTANCE, if there is an assignment to its variables that satisfies all the clauses, and it is called a NO-INSTANCE (with respect to ϵ), if any assignment satisfies at most a fraction $(1-\epsilon)$ of the clauses. Following is one of the several equivalent statements of the PCP theorem [5, 3].

THEOREM 3. *There is a constant $\epsilon : 0 < \epsilon < 1$, such that it is NP-hard to distinguish between* YES-INSTANCES *and* NO-INSTANCES *(defined with respect to ϵ) of the Max 3SAT(5) problem.*

We use the Raz verifier for 3SAT(5) with ℓ parallel repetitions. This is an interactive proof system, in which two provers try to convince the verifier that the input 3SAT(5) formula φ is satisfiable. The verifier chooses, independently at random, ℓ clauses $C_1, \ldots, C_\ell$, and for each $i : 1 \leq i \leq \ell$, a variable x_i participating in clause C_i is chosen. The verifier then sends one query to each one of the two provers, while the query to the first prover consists of the indices of the clauses $C_1 \ldots, C_\ell$, and the query to the second prover contains the indices of the variables $x_1, \ldots, x_\ell$. The first verifier is expected to return an assignment to all the variables in clauses $C_1, \ldots, C_\ell$, which must satisfy the clauses. The second prover returns an assignment to variables $x_1, \ldots, x_\ell$. Finally, the verifier checks that the answers of the two provers are consistent, i.e., for each $i : 1 \leq i \leq \ell$, the assignment to x_i, returned by the second prover, is identical to the assignment to x_i, obtained by projecting the assignment to the variables of C_i, returned by the first prover, onto x_i. (We assume that the answers sent by the first prover always satisfy the clauses appearing in its query). The following theorem is obtained by combining the PCP theorem with the parallel repetition theorem [20].

THEOREM 4. *There exists a constant $\gamma > 0$, such that:*

- *If φ is a* YES-INSTANCE, *then there is a strategy of the provers, for which the acceptance probability is* 1.
- *If φ is a* NO-INSTANCE, *then for any strategy of the provers, the acceptance probability is at most $2^{-\gamma\ell}$.*

We denote the set of all the random strings of the verifier by R , $|R| = (5n)^\ell$, and the sets of all the possible queries of the first and the second prover by Q_1 and Q_2 respectively, $|Q_1| = (5n/3)^\ell$, $|Q_2| = n^\ell$, and set $Q = Q_1 \cup Q_2$. For each query $q \in Q$, let $A(q)$ be the collection of all the possible answers to q (if q is a query of the first prover, then $A(q)$ only contains answers that satisfy all the clauses of the query). Let $A = 7^\ell$, $A' = 2^\ell$. Then for each $q \in Q_1$, $|A(q)| = A$, and for each $q' \in Q_2$, $|A(q')| = A'$. Given a random string $r \in R$, let $q_1(r), q_2(r)$ be the queries sent to the first and the second prover respectively, when the verifier chooses r.

3. THE CONSTRUCTION

Given an input 3SAT(5) formula φ, we construct an instance of directed multicut recursively. The first step is to construct the level-1 instance. Next we define a slight generalization of the level-1 instance, called the *basic instance*, which will help us construct higher-level instances. In general, level-i instance is obtained by composing together level-$(i-1)$ instances with the basic instance. More specifically, we construct a graph G' containing several copies of level-$(i-1)$ instance and graph G'' containing several copies of the basic instance. In order to obtain level-i instance G_i, we add source-sink pairs and edges of G'' to graph G'. The edges of G'' are added to G' as follows. We define a bijection f between the vertices of G' and G''. Let $(v \to v')$ be any edge of G'', and let u, u' be two vertices of G', such that $f(u) = v$ and $f(u') = v'$. Then we add the edge $(u \to u')$ to graph G'.
We start by defining level-1 instances, and then show how to generalize them to obtain the basic instance. Finally, we show how level-i instances are constructed, by composing level-$(i-1)$ instances with the basic instance.

3.1 Level-1 Instances

For each possible random string $r \in R$ of the verifier, we construct a graph $G(r)$, corresponding to string r. Level-1 instance G_1 is simply the union of $G(r)$ for all $r \in R$.
Fix some $r \in R$, and let $q = q_1(r), q' = q_2(r)$. Graph $G(r)$ is a layered graph, containing $L = 32A = 32 \cdot 7^\ell$ layers. In each layer $i : 1 \leq i \leq L$, for each answer $a \in A(q) \cup A(q')$, there is a vertex $v(i, a)$ representing it. Let $a_1, \ldots, a_{A'}$ be all the answers to query q' of the second prover, ordered in some arbitrary way. For each $j : 1 \leq j \leq A'$, we define S_j to be the subset of $A(q)$, containing all the answers to q which are consistent with a_j. Notice that each $a \in A(q)$ belongs to exactly one such set S_j, and $|S_j| \leq 4^\ell$. We assume that all the answers in S_j are ordered in some arbitrary way, $b_1^j, b_2^j, \ldots, b_{|S_j|}^j$, and we will refer to b_1^j and $b_{|S_j|}^j$ as the first and the last answers of S_j, respectively.
We now describe the edges of $G(r)$. Let i, i' be two layers, $1 \leq i < i' \leq L$. We define the edges between the two layers, all of them directed from layer i to layer i'. For each $j : 1 \leq j \leq A'$, for each pair b_k^j, b_{k+1}^j of consecutive answers in S_j (where $1 \leq k < |S_j|$), we add the edge $(v(i, b_k^j) \to v(i', b_{k+1}^j))$. Consider now some $j : 1 \leq j < A'$. Let b be the last answer in S_j, and let b' be the first answer in S_{j+1} (i.e., $b = b_{|S_j|}^j$ and $b' = b_1^{j+1}$). We add the edges: $(v(i, b) \to v(i', b'))$, $(v(i, b) \to v(i', a_{j+1}))$, $(v(i, a_j) \to v(i', b'))$, $(v(i, a_j) \to v(i', a_{j+1}))$. Thus, we add all the edges from level-i vertices representing b and a_j to level-i' vertices representing b' and a_{j+1}.
Let B be the first answer in S_1, and let B' be the last answer in $S_{A'}$. We add a source $s(r)$, which connects to all the copies of B and a_1 (in all the layers), and we add a corresponding sink $t(r)$, to which all the copies of B' and $a_{A'}$ (from all the layers) connect. The pair $(s(r), t(r))$ is the only source-sink pair for graph $G(r)$. This concludes the description of $G(r)$. We now establish its properties.

LEMMA 1. *Let a, a' be a pair of consistent answers to q and q' respectively. Then the removal of all the vertices $v(i, a)$, $v(i, a')$ for $1 \leq i \leq L$ from $G(r)$ separates $s(r)$ from $t(r)$.*

PROOF. Let a, a' be a pair of consistent answers, and assume that $a' = a_i$ for some i, so $a \in S_i$. Consider any source-sink path P. We disregard the layers in which the vertices of P appear and only refer to the answers to which they correspond. Then P has to look as follows: $s(r) \to P_1 \to \cdots \to P_{A'} \to t(r)$, where for each $j : 1 \leq j \leq A'$, P_j either contains a single vertex representing a_j, or it is a path containing all the answers in S_j (in the order in which the answers appear in S_j). Clearly, if a, a' are removed from the graph, then each such path is disconnected. $\square$

Graph $G(r)$ contains $L(A+A')$ vertices, and from the above lemma, there is a collection of vertices of size $2L$ which disconnects $s(r)$ and $t(r)$.

LEMMA 2. *Consider any set $\mathcal{S}$ of non-terminal vertices, whose removal disconnects $s(r)$ and $t(r)$. Then there are at least $L - 7^\ell$ layers i, for which there is a pair of vertices $v(i,a), v(i,a') \in S$, where a, a' are consistent answers to queries q and q', respectively.*

PROOF. Assume otherwise. Let $\mathcal{L} = \{l_1 \leq \cdots \leq l_{7^\ell}\}$ be the layers such that for each l_i, $1 \leq i \leq 7^\ell$, there is no pair (a, a') of consistent answers to q, q', where both $v(i,a)$ and $v(i,a')$ belong to S. We build a path P from $s(r)$ to $t(r)$. Path P is defined to be $P = s(r) \rightarrow P_1 \rightarrow P_2 \rightarrow \cdots \rightarrow P_{2^\ell} \rightarrow t(r)$, where for each $j : 1 \leq j \leq 2^\ell$, P_j either contains a copy of a_j, or it is a path traversing all the answers in S_j. We build path P iteratively. Consider the first $|S_1|$ layers in $\mathcal{L}$. If, for any layer l_i of these layers, a copy of a_1 is not in $\mathcal{S}$, then we set $P_1 = v(l_i, a_1)$. Otherwise, for each one of these layers, none of the answers of S_1 belongs to the solution $\mathcal{S}$. Thus, we can set $P_1 = v(l_1, b_1^1) \rightarrow v(l_2, b_2^1) \rightarrow \cdots, v(l_{|S_1|} \rightarrow b^1_{|S_1|})$. We now consider the next $|S_2|$ layers of $\mathcal{L}$, and build P_2 similarly. $\square$

Level-1 construction is the union of $G(r)$ for all $r \in R$. It is a layered graph with L layers, where for each $i : 1 \leq i \leq L$, layer i is the union of layer-i vertices of all $G(r)$ for all $r \in R$. We note that we do not reuse the vertices, i.e., level-1 graph is actually not connected, and when a vertex representing the same layer-answer pair $v(i,a)$ appears in several graphs $G(r)$, we use fresh copies of this vertex. We have $|R|$ source-sink pairs, which are the union of the source-sink pairs belonging to graphs $G(r)$, for $r \in R$. From Lemma 1, in the YES-INSTANCE, there is a solution of cost $2L|R|$.
We now give some intuition. It is clear that even in the NO-INSTANCE, level-1 construction can be solved by using $2L|R|$ vertices: for each random string $r \in R$, we can select any pair a, a' of consistent answers to $q_1(r), q_2(r)$, and remove all the vertices representing these answers from $G(r)$, thus disconnecting $s(r)$ from $t(r)$. However, the solutions to the YES-INSTANCE and to the NO-INSTANCE look quite different: Let $q \in Q$ be some query, and let $r_1(q), r_2(q), ..., r_p(q)$ be the random strings, such that for each $j : 1 \leq j \leq p$, $q = q_1(r_j)$ if $q \in Q_1$, and $q = q_2(r_j)$ if $q \in Q_2$. In the YES-INSTANCE solution, for each $r_j(q)$, $j : 1 \leq j \leq p$, the same answer $a \in A(q)$ is chosen in each graph $G(r_j(q))$, and the vertices $v(i,a)$ representing this answer are removed. However, in the NO-INSTANCE, for most queries q, we will have to choose many different answers. Moreover, we can show that for most queries q, for most layers i, the solution will contain vertices representing many different answers to query q at layer i. Though this does not affect the solution cost for the level-1 instance, we will make the NO-INSTANCE pay for it at higher levels. The second level graph will contain a union of several level-1 constructions (with some new edges and source-sink pairs added). The source-sink pairs belonging to the level-1 instances are called level-1 source-sink pairs, and the newly added pairs are called level-2 source-sink pairs. In the YES-INSTANCE, the solution cost will remain the sum of the solution costs of the level-1 instances (so we will not have to pay any additional cost for disconnecting level-2 source-sink pairs). In the NO-INSTANCE, if we fix any minimal set $\mathcal{S}$ of vertices, that disconnects level-2 source-sink pairs, then $\mathcal{S}$ has a specific structure with respect to the level-1 instances. More precisely, consider one of the copies of the level-1 instance that is used in building the level-2 instance. Let a be an answer to some query $q \in Q$, and let $i : 1 \leq i \leq L$ be some layer. Then either all the vertices representing a at layer i in this level-1 instance belong to S, or none of them. Since we are in a NO-INSTANCE, a solution of this type cannot be a good solution for the level-1 instances, and thus we will need to pick additional vertices to disconnect level-1 source-sink pairs. Thus, the solution cost of the NO-INSTANCE will accumulate across the levels.

3.2 The Basic Instance Construction

The basic instance is a slight generalization of the level-1 construction, and it will be used when building higher-level instances. For each $r \in R$, we build a graph $G'(r)$, which is defined similarly to $G(r)$, with the following difference: for each layer $i : 1 \leq i \leq L$, for each answer $a \in A(q) \cup A(q')$ (where $q = q_1(r)$, $q' = q_2(r)$), instead of a single vertex $v(i,a)$ we have in $G(r)$, the new graph contains a set $V(i,a)$ of vertices. The cardinality of $V(i,a)$ is X, which is a parameter of our construction, and will be set to different values when we use the basic instance to construct different level instances. If there is an edge $v(i,a) \rightarrow v(i',a')$ in graph $G(r)$ between some pair $v(i,a), v(i',a')$ of vertices, then it is replaced in $G'(r)$ by the following collection of edges: we add an edge from each vertex of $V(i,a)$ to each vertex of $V(i',a')$. Additionally, for each vertex $v(i,a)$, for which there is an edge $(s(r) \rightarrow v(i,a))$ in the original graph, we add edges from $s(r)$ to all the vertices in $V(i,a)$. Similarly, if edge $(v(i,a) \rightarrow t(r))$ belongs to the original graph, we add all the edges from vertices in $V(i,a)$ to $t(r)$ in the new graph. It is easy to see that the following versions of Lemmas 1 and 2 still hold:

LEMMA 3. *Given a pair of consistent answers a, a' to queries q and q' respectively, where $q = q_1(r), q' = q_2(r)$, if we remove from $G'(r)$ all the vertices in $V(i,a)$ and $V(i,a')$, for all $i : 1 \leq i \leq L$, then $s(r)$ is disconnected from $t(r)$.*

LEMMA 4. *Consider any subset $\mathcal{S}$ of non-terminal vertices, whose removal disconnects $s(r)$ from $t(r)$ in $G'(r)$. Then there are at least $L - 7^\ell$ layers i, where for each such i there is a pair of consistent answers a, a' to queries q and q' respectively, such that* **all** *the vertices of $V(i,a)$ and $V(i,a')$ belong to the solution.*

The basic instance is defined to be the union of $G'(r)$ for all $r \in R$, exactly as the level-1 instance, with the same layered structure as in the level-1 instance. Notice that if $X = 1$, then the basic instance is exactly the level-1 instance.
Let $q \in Q$ be some query, $a \in A(q)$ be some answer to this query, and $i : 1 \leq i \leq L$ be some layer. We define $U(i,q,a)$ to be the set of all the vertices in the ith layer of the basic instance representing the answer a to query q, i.e., for each random string $r \in R$, such that $q \in \{q_1(r), q_2(r)\}$, the set $V(i,a)$ of vertices of $G'(r)$ is contained in $U(i,q,a)$. Thus, if q is a query of the first prover, then $U(i,q,a)$ is a union of 3^ℓ such sets (as each query $q \in Q_1$ participates in 3^ℓ random strings), and $|U(i,q,a)| = 3^\ell X$, and if q is a query of the second prover, then $U(i,q,a)$ is a union of 5^ℓ

such sets, and $|U(i,q,a)| = 5^\ell X$. Notice that sets $U(i,q,a)$ define a partition of the vertices of the basic instance, and each vertex belongs to exactly one such set.

3.3 Level-2 Instances

To motivate our general construction, we first describe how we go to level-2 instances starting from level-1 instances. We start by constructing two auxiliary graphs: $G' = (V', E')$ is a union of $L \cdot 15^\ell$ level-1 instances, and $G'' = (V'', E'')$ is a union of L basic instances with parameter $X_2 = 15^\ell$. The level-2 graph $G_2 = (V_2, E_2)$ is obtained by combining these two graphs, in the following fashion. Let T', T'' denote the sets of the the terminal vertices in G', G'' respectively, and let $\mathcal{P}', \mathcal{P}''$ denote the collections of source-sink pairs in these graphs. The vertex set of G_2 is $V_2 = V' \cup T''$, and the source-sink pairs are $\mathcal{P}' \cup \mathcal{P}''$. We will refer to $\mathcal{P}'$ and $\mathcal{P}''$ as level-1 and level-2 pairs, respectively. The edge set E_2 of G_2 consists of two subsets. The first subset is called level-1 edges, and these are all the edges in E'. The second subset is called level-2 edges. In order to define this set of edges, we first build a bijection $f : (V_2 \setminus T') \rightarrow V''$. Level-2 edges are added as follows: let $v, v' \in V''$, such that edge $(v \rightarrow v')$ belongs to G''. Let $u, u' \in V_2$, such that $f(u) = v, f(u') = v'$. Then we add edge $(u \rightarrow u')$ to E_2. When defining function f, we need to be very careful, so that adding level-2 edges to the graph does not create any cheating paths for either level-1 or level-2 source-sink pairs (a cheating path for any source-sink pair contains edges of more than one type. The presence of the cheating paths in the graph might create a problem in the YES-INSTANCE analysis, as the standard solution that contains all the vertices representing the "correct" answers to the queries, may no longer be a feasible solution, since it does not necessarily disconnects the cheating paths). The fact that level-1 instance is a layered construction, and all the edges are directed from lower-indexed to higher-indexed layers helps to avoid this problem.

Function f is defined as follows. Recall that graph G' is a union of $L \cdot 15^\ell$ level-1 instances. We partition them into L subsets $\mathcal{H}_i$ ($i : 1 \le i \le L$) of 15^ℓ level-1 instances each. The vertices in level-1 instances of set $\mathcal{H}_i$ are mapped to the vertices of the ith layer of the basic instances in G''. Moreover, each copy $H \in \mathcal{H}_i$ of level-1 instance, for each $j : 1 \le j \le L$, the vertices of the jth layer of H are mapped to vertices in the jth basic instance of G'', (in its ith layer). Thus, after adding level-2 edges to graph G_2, we obtain L^2 "layers", which are referred to from now on as groups, of vertices, $G_{j,i}$ where $1 \le i \le L, 1 \le j \le L$. Group $G_{j,i}$ contains, for each $H \in \mathcal{H}_i$, all the layer j vertices of H. Vertices in group $G_{j,i}$ are mapped to the vertices of the ith layer of the jth basic instance of G''. The important feature of this construction is that for any pair of groups $G_{j,i}, G_{j',i'}$, there are level-1 edges from $G_{j,i}$ to $G_{j',i'}$ only if $j < j'$, $i = i'$, and there are level-2 edges from $G_{j,i}$ to $G_{j',i'}$ only if $i < i'$ and $j = j'$. Figure 1 shows how level-2 instances are constructed.

We now complete the description of f. Consider the jth basic instance in G'', $1 \le j \le L$, its ith layer, for $1 \le i \le L$. The set of vertices $G_{j,i}$ includes all the layer-j vertices of all $H \in \mathcal{H}_i$. Recall that in the basic instance, for each layer i, for each prover-1 answer a, we have a set $U(i,q,a)$ of vertices that consists of a union of 3^ℓ sets $V(i,a)$ of vertices (one set for each random string in which the corresponding query participates), and for each prover-2 answer a', we have 5^ℓ sets $V(i,a')$. We also know that $\mathcal{H}_i$ contains 15^ℓ instances of level 1 construction. Fix some answer a to some prover 1 query q. In each $H \in \mathcal{H}_i$, let $W(H,j,q,a)$ be the set of all the vertices representing a in jth layer of H (i.e., $W(H,j,q,a)$ is actually set $U(j,q,a)$ of instance H). Since we have 15^ℓ graphs $H \in \mathcal{H}_i$, we have 15^ℓ such sets. We partition them arbitrarily into collections containing 5^ℓ such sets each, thus obtaining a collection of 3^ℓ equal-sized sets of vertices representing a, as desired. The vertices in each such set are arbitrarily mapped to the vertices of one of the sets $V(i,a)$ in the jth basic instance of G''. Recall that in the level-1 instance, for each layer $j : 1 \le j \le L$, for each answer a to a prover-1 query, there are 3^ℓ layer-j vertices representing a. Thus, for each H, $|W(H,j,q,a)| = 3^\ell$, and the number of vertices we map to each set $V(i,a)$ is $X_2 = 3^\ell \cdot 5^\ell = 15^\ell$. We do the same for each prover-2 answer a' to each query q', only this time we will partition the sets $W(H,j,q'a')$ into a collection of 5^ℓ equal-sized sets, each containing 3^ℓ of such sets. For prover-2 answers a', we have that $|W(H,a',j)| = 5^\ell$, and thus the number of vertices mapped to each set $V(i,a')$ is again $X_2 = 15^\ell$. Finally, for the set T'' of terminals, function f is defined to be the identity function.

3.4 Level-h Instances

We now formally describe level-h construction. An invariant that we will keep throughout the construction:

Invariant For each h, the vertices of the level-h instance are partitioned into L^h groups $G_{z_1,z_2,\dots,z_h}$, where $1 \le z_j \le L$ for each $j : 1 \le j \le h$. Given two groups $Y = G_{z_1,z_2,\dots,z_h}$ and $Z = G_{z'_1,z'_2,\dots,z'_h}$, for each $j : 1 \le j \le h$, there are level-j edges from Y to Z only if $z_j < z'_j$ and for all $k \neq j$, $z_k = z'_k$. For each query $q \in Q_1 \cup Q_2$, for each answer $a \in A(q)$, group $G_{z_1,z_2,\dots,z_h}$ contains a set $U(z_1, z_2, \dots, z_h, q, a)$ of vertices representing a, and the sizes of these sets are identical for all q, a, for all groups $G_{z_1,z_2,\dots,z_h}$.

In order to create a level-h instance, we again start by building two auxiliary graphs $G' = (V', E')$ and $G'' = (V'', E'')$, where G' is a union of $L \cdot 15^\ell$ level-$(h-1)$ instances, and G'' is a union of L^{h-1} basic instances with parameter X_h, which will be specified later. Let $T', T'', \mathcal{P}', \mathcal{P}''$ denote the terminal sets and the sets of the source-sink pairs of the two graphs, respectively. The level-h graph $G_h = (V_h, E_h)$ is defined as follows. The vertex set is $V_h = V' \cup T''$, and the set of source-sink pairs is $\mathcal{P}' \cup \mathcal{P}''$, where $\mathcal{P}'$ contains level-$1, \dots, h-1$ pairs, while pairs in $\mathcal{P}''$ are called level-h source-sink pairs. There are two sets of edges in E_h. The first set is exactly the set E' of edges from graph G', and it contains level-$1, \dots, h-1$ edges from the previous recursion levels. The second set contains level-h edges, and it is defined by the means of a bijection $f : (V_h \setminus T') \rightarrow V''$, as follows. For each edge $(v \rightarrow v') \in E''$, there is an edge $(u \rightarrow u') \in E$, where $u, u' \in V_h \setminus T'$, such that $f(u) = v, f(u') = v'$. In order to finish the description of level-h construction, it now only remains to define the bijection f. We partition the collection of level-$(h-1)$ instances in G' into L sets $\mathcal{H}_1, \dots, \mathcal{H}_L$, each containing 15^ℓ level-$(h-1)$ instances. We also denote the L^{h-1} basic instances of G'' by $B_{z_1,\dots,z_{h-1}}$, where for $j : 1 \le j \le h-1$, $1 \le z_j \le L$. For each h-tuple $(z_1, \dots, z_{h-1}, i)$, where $1 \le z_j \le L$ for all $1 \le j \le h-1$, and for each $i : 1 \le i \le L$, group $G_{z_1,\dots,z_{h-1},i}$ is defined as follows. It is the union of all the groups $G_{z_1,z_2,\dots,z_{h-1}}$ of instances belonging to $\mathcal{H}_i$. We map the vertices of this group

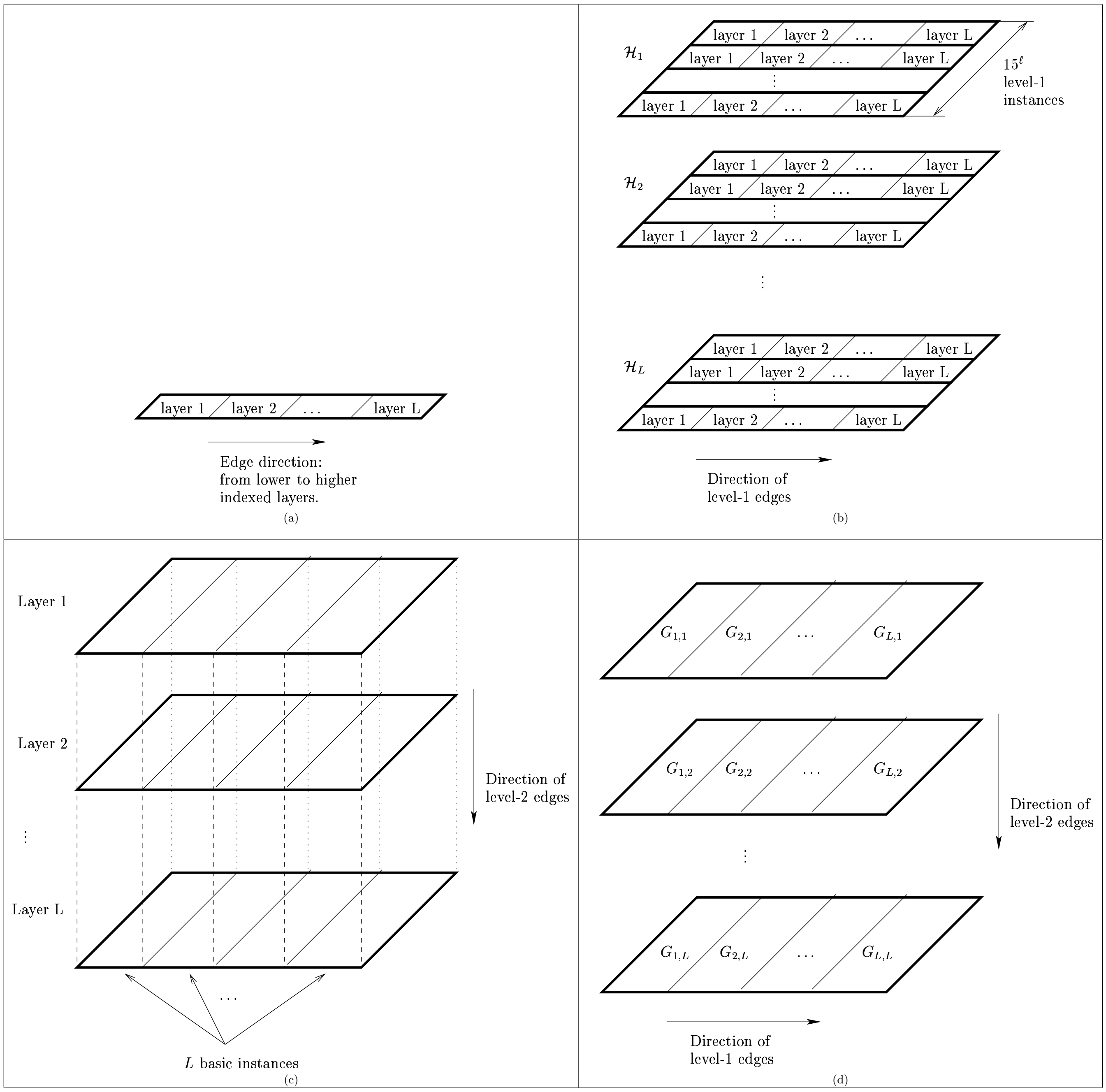

Figure 1: Level-2 instance construction. (a) Schematic view of level-1 instance; (b) Graph G'; (c) Mapping vertices of G' to vertices of G''; (d) Level-2 instance.

to the vertices of the ith layer of basic instance $B_{z_1,\ldots,z_{h-1}}$. Since we add level-h edges according to bijection f, and since, in the basic instance, the edges are directed from lower-indexed to higher-indexed coordinates, the invariant still holds.

Consider now the ith layer of instance $B_{z_1,\ldots,z_{h-1}}$, and the group $G_{z_1,z_2,\ldots,z_{h-1},i}$ of vertices. We show how to map vertices in $G_{z_1,z_2,\ldots,z_{h-1},i}$ to layer-i vertices of basic instance $B_{z_1,\ldots,z_{h-1}}$. Let a be some answer to some prover-1 query q. Then each original group $G_{z_1,z_2,\ldots,z_{h-1}}$ of each graph $H \in \mathcal{H}_i$ contains a collection $W(H, z_1, z_2, \ldots, z_{h-1}, a, q)$ of vertices representing a (i.e., $W(H, z_1, z_2, \ldots, z_{h-1}, a, q)$ is exactly the set $U(z_1, z_2, \ldots, z_{h-1}, a, q)$ for instance H). Thus, we have 15^{ℓ} such sets belonging to different graphs. We arbitrarily partition these sets into 3^{ℓ} collections of 5^{ℓ} sets each. Each such collection is arbitrarily mapped to $V(i, a)$ in some $G'(r)$ in the basic instance $B_{z_1,\ldots,z_{h-1}}$, where $q = q_1(r)$. Answers to second prover queries are treated similarly, except that we partition them into 5^{ℓ} collections, of 3^{ℓ} sets each. Finally, on the vertices of T'', function f is defined to be the identity function.

The total number of levels in our graph is $H = 2^{\gamma\ell/3}$. The final construction is obtained from G_H, with the following change: recall that a basic instance contains $|R|$ source-sink pairs, and that level-h instance contains $L\cdot 15^{\ell}$ copies of level-$(h-1)$ instances. Therefore, for each level $h : 1 \le h \le H$, graph G_h contains $|R| \left(L \cdot 15^{\ell}\right)^{H-h}$ source-sink pairs from level h. We need the number of source-sink pairs to be identical for all levels. Thus, for each source-sink pair (s, t) from level h (for all $h : 1 \le h \le H$), we replace s by $\left(L \cdot 15^{\ell}\right)^{h}$ sources that connect to all the vertices to which s has been connected. Similarly, t is replaced by the same number of copies, and all the vertices that previously connected to t now connect to each one of the copies. Finally, we define an arbitrary matching between the copies of s and the copies of t, and add the corresponding pairs to the collection of source-sink pairs that need to be disconnected. Thus, for each level h, $1 \le h \le H$, the final instance contains $|R| \left(L \cdot 15^{\ell}\right)^{H}$ source-sink pairs of level h. This completes the construction description.

Graph Size: It is easy to see that the size N_1 of level-1 construction is $L|R|(A + A') = n^{O(\ell)}$, and the recursive formula for the size N_h of level-h construction is: $N_h = N_{h-1} \cdot 15^{\ell} \cdot L$. Thus, $N_h = L^h|R|(A + A')15^{\ell(h-1)} = n^{O(\ell)} \cdot 2^{O(\ell h)}$.

Recall that a basic instance is a union of $G'(r)$ for $r \in R$, and that in each layer i of $G'(r)$, for each answer a to $q_1(r)$ or $q_2(r)$, we have a set $V(i, a)$ of vertices. We denote by X_h the size of this set for level-h instances (we will see that this size is the same for all answers to all queries). Clearly, $X_1 = 1$. In order to calculate X_h for $h > 1$, let a be some answer to some query $q \in Q_1$. Recall that for each $i : 1 \le i \le L$, for each basic instance we construct when building a level h graph, set $V(i, a)$ is a union of 5^{ℓ} sets, each of which contains all copies of a in a single group of a level $h-1$ instance. It is not hard to see that each such group of a single instance contains 3^{ℓ} sets $S(i', a)$ whose size is X_{h-1}. Thus, when a is an answer to a query of prover 1, we get that $X_h = 15^{\ell}X_{h-1}$. When a is an answer to a query of prover 2, we get a similar formula. Therefore, $X_h = 15^{\ell(h-1)}$.

Hardness Factor: The total number of levels in our graph is $H = 2^{\gamma\ell/3}$. In Section 4, we will show that in case of a YES-INSTANCE, there is subset of $2L^H|R|X_H$ vertices that disconnects all the source-sink pairs. On the other hand, in case of a NO-INSTANCE, we will show in Section 5 that the solution cost is $\Omega(HL^H|R|X_H)$, even if only a 0.99 fraction of the source-sink pairs need to be disconnected. Thus, the gap between the YES-INSTANCE and the NO-INSTANCE is $\Omega(H)$. Since the construction size is $N = n^{O(\ell)}2^{O(\ell H)}$, by choosing $\ell = \Theta(\log\log n)$, we get that the gap factor is $H = \Omega(\log N/\log\log N)$, while the construction size is bounded by $N = n^{\text{poly}\log n}$.

4. YES-INSTANCE ANALYSIS

When the input formula φ is a YES-INSTANCE, there is a strategy of the provers for which the verifier always accepts. For each $q \in Q$, let $g(q) \in A(q)$ be the answer to query q under this strategy. Define $\mathcal{S}$ to be the set of all the vertices of graph G_H that represent answers $g(q)$ for all $q \in Q$. The cost of $\mathcal{S}$ is $2L^H|R|X_H$: Graph G_H consists of L^{H-1} basic instances, each containing $|R|$ graphs $G'(r)$. For each $r \in R$, the number of vertices representing $g(q_1(r))$ and $g(q_2(r))$ is $2LX_H$, and thus the number of vertices in $\mathcal{S}$ is $2L^H|R|X_H$. The lemma below shows that $\mathcal{S}$ is indeed a feasible solution to G_H.

LEMMA 5. *The removal of $\mathcal{S}$ from the graph disconnects all the source-sink pairs.*

PROOF. Fix any $j : 1 \le j \le H$, and let $s(r), t(r)$ be any level-j source-sink pair in G_H. Let $C_j, C_{j+1}, \ldots, C_H$ denote the copies of the level-$j, j+1, \ldots, H$ instances, respectively, that have been used to construct the final instance, which contain $s(r)$ and $t(r)$. We prove by induction that for each such instance, any path connecting $s(r)$ to $t(r)$ contains only level-j edges belonging to instance C_j, and thus it is disconnected by Lemma 3 (if we look at the vertices of instance C_j and the level-j edges connecting them, we obtain a collection of graphs $G'(r)$ which are not connected to each other).

We start from C_j. Since $s(r), t(r)$ is a level-j source-sink pair, it belongs to some basic instance $B_{z_1,\ldots,z_{j-1}}$ of graph G'' used to construct the level-j instance. Then for each $i : 1 \le i \le L$, the vertices that are mapped to the ith layer of $B_{z_1,\ldots,z_{j-1}}$ are the vertices of the group $G_{z_1,\ldots,z_{j-1},i}$. Suppose there is a path from $s(r)$ to $t(r)$ in C_j that contains edges from levels $1, \ldots, j-1$. When using such an edge for the first time, the path moves to some group $G_{z'_1,\ldots,z'_{j-1},i'}$, such that for some $y : 1 \le y \le j-1$, $z'_y > z_y$. Since all the edges go from lower to higher indexed groups, this path can never return to a group $G(z_1, \ldots, z_{j-1}, i'')$ for any i'', i.e., it is impossible that the path eventually reaches $t(r)$.

We now assume that the lemma is true for some C_y, $y \ge j$, and prove it for C_{y+1}. Recall that all the vertices of C_y belong to some layer x of the new level-$(y+1)$ construction. The only edges inside this layer are the edges that are contained in the level y instances, i.e., contained in C_y in our case. By the induction hypothesis, any path connecting $s(r)$ to $t(r)$, that uses only these edges, consists of level-j edges belonging to C_j only. Assume now there is a path that leaves layer x. Since all the edges in the new instance are directed from lower indexed to higher indexed layers (or are contained inside a layer), it is impossible for the path to return to layer x, thus it cannot reach $t(r)$. □

5. NO-INSTANCE ANALYSIS

We start by analyzing the basic instance. Suppose we have a basic instance construction $B = (V, E)$ with parameter X. Let $\mathcal{S}' \subseteq V$ be any subset of vertices. We call this subset *canonical*, if for each layer $i : 1 \leq i \leq L$, for each query $q \in Q$ and for each answer $a \in A(q)$, either all the vertices representing a in layer i belong to $\mathcal{S}'$, or none of them does. In other words, either $U(i, q, a) \subseteq \mathcal{S}'$, or $U(i, q, a) \cap \mathcal{S}' = \emptyset$. The intuitive idea is that any minimal subset of vertices, that disconnects source-sink pairs of higher levels, is a canonical subset of vertices for the current level. We show that even if this subset is quite large, we still need to remove many vertices to disconnect the source-sink pairs of the current level.

THEOREM 5. *Let $\mathcal{S} \subseteq V$ be a subset of vertices whose removal disconnects at least a fraction $\frac{15}{16}$ of the source-sink pairs, where $\mathcal{S} = \mathcal{S}' \cup \mathcal{S}''$, and $\mathcal{S}'$ is a canonical subset containing less than $L|R|XH$ vertices. Then $\mathcal{S}''$ contains at least $\frac{L|R|X}{2^9}$ vertices.*

PROOF. We say that layer i is good, if at most $4|R|X \cdot H$ of its vertices belong to $\mathcal{S}'$. Let $\mathcal{L}$ denote the set of all the good layers. Then $|\mathcal{L}| \geq \frac{3}{4}L$.
Consider now some good layer i. For a query $q \in Q$, we denote by $\mathcal{S}(q, i)$ the set of all the answers $a \in A(q)$, such that $U(i, q, a) \subseteq \mathcal{S}'$. We say that q is good, if $|\mathcal{S}(q, i)| \leq 16H$.

CLAIM 1. *For any good layer $i : 1 \leq i \leq L$, at least $3/4$ of the queries Q_1 and at least $3/4$ of the queries Q_2 are good.*

PROOF. If we assume that less than $3/4$ of queries Q_1 are good, then we have $\frac{|Q_1|}{4}$ queries q, for which $|\mathcal{S}(q, i)| > 16H$. Since each query $q \in Q_1$ participates in 3^ℓ random strings, we have that the number of vertices corresponding to each (not-good) query that belong to $\mathcal{S}'$ is more than $3^\ell X \cdot 16H$. Since $|Q_1| = (5n/3)^\ell$, we have that more than $\frac{1}{4}(5n/3)^\ell \cdot 3^\ell X \cdot 16H = 4|R|XH$ vertices of layer i are in $\mathcal{S}'$, which is impossible since i is good.
The calculation for $q \in Q_2$ is similar, except that now each query participates in 5^ℓ random strings, and $|Q_2| = n^\ell$, yielding the number of vertices corresponding to bad queries belonging to $\mathcal{S}'$ is more than: $\frac{1}{4}|Q_2|5^\ell X \cdot 16H = (5n)^\ell X \cdot 4H = 4|R|XH$. $\square$

Given a good layer i, we define $R' \subseteq R$ to be a subset of random strings r for which both queries $q_1(r)$ and $q_2(r)$ are good. It is easy to see that $|R'| \geq \frac{1}{2}|R|$, for example by choosing $r \in R$ at random and calculating the probability that both its queries are good.
Let $R'' \subseteq R'$ be the subset of random strings r, such that there is a pair (a, a') of consistent answers to $q_1(r), q_2(r)$, for which both $U(i, q, a) \subseteq \mathcal{S}'$ and $U(i, q, a') \subseteq \mathcal{S}'$.

CLAIM 2. $|R''| \leq \frac{1}{2}|R'|$

PROOF. Assume otherwise. We define a strategy of the provers, for which the acceptance probability of the verifier is more than $2^{-\gamma\ell}$, which is impossible for a NO-INSTANCE. For each good query q, we choose one of its $16H$ answers whose corresponding vertices are contained in $\mathcal{S}'$. For a non-good query q, an arbitrary answer is chosen. The probability of choosing a random string in R'' is at least $\frac{1}{4}$, and if the verifier chooses $r \in R''$, then the probability that the provers choose a pair a, a' of matching answers to $q_1(r)$, $q_2(r)$ is at least $1/(16H)^2$ (since $r \in R''$, we know that there is a pair of matching answers (a, a'), such that $U(i, q_1(r), a), U(i, q_2(r), a') \in \mathcal{S}'$). In total, the probability that the verifier accepts is at least $\frac{1}{4(16H)^2} > 2^{-\gamma\ell}$ by the choice of H. $\square$

We say that r is good for layer i (when i is a good layer itself), if $r \in R' \setminus R''$. By the above discussion, for each good layer, there are at least $|R|/4$ good random strings.
Let P be the set of all the pairs (r, i) where i is a good layer, (i.e., $i \in \mathcal{L}$) and r is good for layer i. Then clearly $P \geq |\mathcal{L}||R|/4$. We say that a random string r is interesting, if there are at least $\frac{|\mathcal{L}|}{8}$ good layers i, such that r is good for i.

CLAIM 3. *The number of interesting strings r is at least $\frac{|R|}{8}$.*

PROOF. Assume otherwise. An interesting random string r is good for at most $|\mathcal{L}|$ good layers, while a non-interesting r is good for at most $|\mathcal{L}|/8$ good layers. Thus, we can bound $|P| < \frac{|R|}{8} \cdot |\mathcal{L}| + |R| \cdot \frac{|\mathcal{L}|}{8} < \frac{|R||\mathcal{L}|}{4}$, which is a contradiction. $\square$

We now complete the proof of the theorem. Let $J \subseteq R$ be the set of all the random strings r, such that r is interesting, and $s(r)$, $t(r)$ are disconnected by $\mathcal{S}$. Since the fraction of interesting strings is at least $\frac{1}{8}$, and the fraction of the source-sink pairs disconnected by $\mathcal{S}$ is at least $\frac{15}{16}$, we have that $|J| \geq \frac{1}{16}$. For each $r \in J$, we have at least $|\mathcal{L}|/8 \geq 3L/32 \geq L/16 \geq 2A$ layers i, for which $\mathcal{S}'$ does not contain $U(i, q_1(r), a), U(i, q_2(r), a')$ for any pair of consistent answers (a, a') to $q_1(r)$ and $q_2(r)$. Therefore, following Lemma 4, we need to remove at least $AX = LX/32$ vertices from these layers in $G'(r)$ to disconnect the $s(r)$-$t(r)$ pair. Thus in total, $|\mathcal{S}''| \geq \frac{LX}{32} \cdot \frac{|R|}{16} = \frac{L|R|X}{2^9}$.

Consider any solution $\mathcal{S}$ to the NO-INSTANCE, that disconnects at least a fraction 0.99 of the source-sink pairs. Let $\mathcal{F}$ denote the collection of levels h, for which the fraction of level-h source-sink pairs disconnected by $\mathcal{S}$ is at least 0.95. Also, let $F = |\mathcal{F}|$. Clearly, $F \geq \frac{4H}{5}$. Assume that $\mathcal{F} = \{h_1, h_2, \ldots, h_F\}$, where $h_1 \leq h_2 \leq \cdots \leq h_F$. We derive from $\mathcal{S}$ a disjoint collection $S_{h_1}, \ldots, S_{h_F}$ of subsets as follows. Define S_{h_F} to be a minimal subset of $\mathcal{S}$ that disconnects a fraction 0.95 of the level-h_F source-sink pairs (i.e., if we remove any vertex from S_{h_F}, then the fraction of source-sink pairs that remain disconnected is less than 0.95). Set $S_{h_{(F-1)}}$ contains the minimal subset of $\mathcal{S}$ that needs to be removed in addition to S_F to disconnect a fraction 0.95 of level-$h_{(F-1)}$ source-sink pairs and so on. For all $i : 0 \leq i \leq F$, we define $C_{h_i} = \sum_{j=i+1}^{F} |S_{h_j}|$. Clearly, $C_{h_0} \leq |\mathcal{S}|$. Our goal now is to show that $C_{h_0} = \Omega(H|R|L^H X_H)$. The next theorem shows that whenever C_{h_i} is small, $|S_{h_i}|$ is large.

THEOREM 6. *For each $i = 1, \ldots, F$, if $C_{h_i} < \frac{1}{10}H|R|L^H X_H$, then $|S_{h_i}| \geq \frac{1}{2^{13}}|R|L^H X_H$.*

PROOF. Consider our final level H instance. Let $\tilde{G}_{h_i}$ be the graph obtained from the final instance when we remove edges and source-sink pairs of levels higher than h_i from it. In other words, $\tilde{G}_{h_i}$ contains all the copies of level-h_i instance that we use in the final construction. It is easy to see that the number of these copies in $\tilde{G}_{h_i}$ is $(15^\ell L)^{(H-h_i)}$. Let $\overline{G}_{h_i}$ be the graph defined on the same set of vertices as $\tilde{G}_{h_i}$, where only level-h_i edges are present. Then $\overline{G}_{h_i}$ is actually a union of basic instances (with multiple copies of the sources and the sinks of the basic instance). Since each level-h_i construction uses L^{h_i-1} copies of the basic instance, the total number of basic instances in $\overline{G}_{h_i}$ is $Z_{h_i} = (15^\ell L)^{(H-h_i)}L^{h_i-1} = 15^{\ell(H-h_i)} \cdot L^{H-1}$. Let $\mathcal{S}' = \bigcup_{j=i+1}^{F} S_{h_j}$. For each basic instance B, let $\mathcal{S}_B \subseteq \mathcal{S}'$ be the vertices of $\mathcal{S}'$ that belong to B. By the definition of $\mathcal{S}'$, it is easy to see that $\mathcal{S}_B$ is a canonical set for B. Recall that for each level h_i, $X_{h_i} = 15^{\ell(h_i-1)}$. Thus we have:

$$\begin{aligned}|\mathcal{S}'| = C_{h_i} &< \frac{1}{10}H|R|L^H X_H \\ &= \frac{1}{10}H|R|L(L^{H-1}15^{\ell(H-h_i)})15^{\ell(h_i-1)} \\ &= \frac{1}{10}H|R|LX_{h_i}Z\end{aligned}$$

We say that a basic instance B is *good* if $|\mathcal{S}_B| \le HL|R|X_{h_i}$. Clearly, the fraction of good basic instances is at least 0.9. We say that a basic instance B is *interesting*, if the fraction of the source-sink pairs of B that are disconnected by $\mathcal{S}' \cup S_{h_i}$ is at least $\frac{15}{16}$. From a simple counting argument, at least a 0.2-fraction of the basic instances are interesting. Thus, in total, we have at least 0.1 fraction of basic instances which are both good and interesting.
Recall that for each basic instance, each source-sink pair (s,t) is replaced by several copies. If we look at some interesting basic instance B with the original (s,t)-pairs (i.e., we have one copy of each pair), then the fraction of these original pairs, which are disconnected by $\mathcal{S}_B$ is at least $\frac{15}{16}$. Therefore, we can invoke Theorem 5 for each good and interesting basic instance, which thus contributes at least $\frac{L|R|X_{h_i}}{2^9}$ vertices to S_{h_i}. In total,

$$\begin{aligned}|S_i| &\ge 0.1 Z_{h_i} \cdot \frac{L|R|X_{h_i}}{2^9} \\ &\ge \frac{1}{2^{13}}\left(15^{\ell(H-h_i)} \cdot L^{H-1}\right) L|R|15^{\ell(h_i-1)} \\ &= \frac{1}{2^{13}}|R|L^H 15^{\ell(H-1)} \\ &= \frac{1}{2^{13}}|R|L^H X_H\end{aligned}$$

□

Corollary 1. $C_{h_0} = \Omega(H|R|L^H X_H)$.

Proof. If for some $i \in [1..F]$, $C_{h_i} \ge \frac{H}{10}|R|L^H X_H$, the corollary is trivially true. Otherwise, by Theorem 6, we know that

$$\begin{aligned}C_{h_0} &= |S_{h_1}| + |S_{h_2}| + \ldots + |S_{h_F}| \\ &\ge |F| \cdot \frac{1}{2^{13}}|R|L^H X_H \\ &\ge \frac{4H}{5} \cdot \frac{1}{2^{13}}|R|L^H X_H \\ &= \Omega(H|R|L^H X_H).\end{aligned}$$

□

Thus we have a gap of $\Omega(H)$ between Yes-Instance and No-Instance even if the No-Instance solution is required to disconnect only a 0.99-fraction of the source-sink pairs.

6. REFERENCES

[1] P. Alimonti and V. Kann. Hardness of approximating problems on cubic graphs. *Theoretical Computer Science*, 237: 123-134, 2000.

[2] S. Arora, J. R. Lee, A. Naor. Euclidean distortion and the sparsest cut. In *Proc. of STOC*, 2005, pp. 553–562.

[3] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[4] S. Arora, S. Rao, U. V. Vazirani. Expander flows, geometric embeddings and graph partitioning. In *Proc. of STOC*, 2004, pp. 222–231.

[5] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *JACM*, 45(1):70–122, 1998.

[6] M. Charikar, K. Makarychev, Y. Makarychev. Directed Metrics and Directed Graph Partitioning Problems. To appear in *Proc. of SODA*, 2006.

[7] S. Chawla, R. Krauthgamer, R. Kumar, Y. Rabani, D. Sivakumar. On the Hardness of Approximating Multicut and Sparsest-Cut. in *Proc. IEEE Conference on Computational Complexity*, 2005, pp. 144–153.

[8] J. Cheriyan, H. J. Karloff, Y. Rabani. Approximating Directed Multicuts. In *Proc. of FOCS*, 2001, pp. 320–328.

[9] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, M. Yannakakis. The Complexity of Multiterminal Cuts. In *SIAM J. Comput.*, 23(4): 864-894, 1994.

[10] U. Feige. Relations between average case complexity and approximation complexity. In *Proc. of STOC*, 2002, pp. 534–543.

[11] U. Feige and S. Kogan. Hardness of Approximation of the Balanced Complete Bipartite Subgraph Problem. Technical Report MCS04-04, Department of Computer Science and Applied Math., The Weizmann Institute of Science, 2004.

[12] L. R. Ford, D. R. Fulkerson, 1962. Flows in Networks. Princeton University Press, Princeton, NJ.

[13] N. Garg, V. Vazirani, M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. in *Proc. of STOC*, 1993, pp. 698–707.

[14] N. Garg, V. Vazirani, M. Yannakakis. Primal-Dual Approximation Algorithms for Integral Flow and Multicut in Trees. *Algorithmica*, 18(1):3–20, 1997. Preliminary version in *Proc. of ICALP*, 1993.

[15] A. Gupta. Improved results for directed multicut. In *Proc. of SODA*, 2003, pp. 454-455.

[16] M.T. Hajiaghayi, H. Räcke. An $O(\sqrt{n})$-Approximation Algorithm For Directed Sparsest Cut, To appear in *Information Processing Letters.*

[17] S. Khot. On the power of unique 2-prover 1-round games. In *Proc. of STOC*, 2002, pp. 767–775.

[18] S. Khot, N. K. Vishnoi. The Unique Games Conjecture, Integrality Gap for Cut Problems and the Embeddability of Negative Type Metrics into ℓ_1. In

Proc. of FOCS, 2005.

[19] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *JACM*, 46(6):787–832, 1999. Preliminary version in *Proc. of FOCS*, 1988.

[20] R. Raz. A parallel repetition theorem. *SIAM J. of Computing*, 27(3):763–803, 1998.

[21] M. E. Saks, A. Samorodnitsky, L. Zosin. A Lower Bound On The Integrality Gap For Minimum Multicut In Directed Networks. Combinatorica 24(3): 525–530 (2004).

APPENDIX

A. APX-HARDNESS OF UNDIRECTED SPARSEST CUT

We perform a reduction from the vertex cover problem on cubic graphs. In the vertex cover problem on cubic graphs we are given a graph $G(V, E)$, where all the vertices have degree 3. The goal is to choose a minimum-cardinality subset S of vertices, such that for each edge in E, at least one endpoint of the edge is in S. Alimonti and Kann [1] show that the problem is APX-hard. Thus, there is some constant $\epsilon : 0 < \epsilon < 1$, such that there is no $(1+\epsilon)$-approximation for this problem, unless $\mathsf{P} = \mathsf{NP}$. For any pair α, β of parameters, $0 < \alpha \leq 1$, $\beta \geq 1$, an approximation algorithm for the vertex cover problem is an (α, β)-bicriteria approximation if it always outputs a solution that covers at least an α-fraction of the edges, and whose cost is at most $\beta\mathsf{OPT}$, where OPT is the cost of the optimal solution to the vertex cover problem, which covers all the edges.

Claim 4. *There is no $(\epsilon/6, (1+\epsilon/2))$-bicriteria approximation algorithm for the vertex cover problem on cubic graphs, unless $\mathsf{P} = \mathsf{NP}$.*

Proof. Assume such an algorithm exists. We show that this algorithm provides an $(1+\epsilon)$-approximation for the vertex cover problem on cubic graphs. Given an input cubic graph $G = (V, E)$, let OPT denote the cost of the optimal solution to the vertex cover problem, that covers all the edges. We apply the algorithm on G, obtaining a set S of vertices, $|S| \leq (1+\epsilon/2)\mathsf{OPT}$ that covers at least $|E|\epsilon/6$ edges. For each one of the edges that are not covered in the solution, we choose one of its endpoints to be added to the solution. Let S' denote the set of these endpoints. Then $|S'| = |E|\epsilon/6$. However, since in a cubic graph each vertex can cover at most three edges, $\mathsf{OPT} \geq |E|/3$. Therefore, $|S'| \leq \mathsf{OPT}\epsilon/2$. Altogether, $S \cup S'$ is a feasible solution to the vertex cover problem, that covers all the edges, and whose cost is at most $\mathsf{OPT}(1+\epsilon)$. $\square$

We now use the reduction of Garg, Vazirani and Yannakakis [14] from vertex cover to undirected multicut on trees of height one. For the sake of completeness we describe the reduction. Given an instance $G = (V, E)$ of the vertex cover problem, the multicut instance is constructed as follows. The vertex set is $V \cup \{r\}$, and for each vertex $v \in V$, there is an edge connecting v to r. For each edge $e = (u, v)$ of the original graph, we create a source-sink pair (u, v). It is easy to see that any solution S to the vertex cover problem can be translated into a solution of the multicut instance, by removing the edges incident on vertices of S. The reverse direction is also immediate. Thus, following Claim 4, there is no $(\epsilon/6, (1+\epsilon/2))$ bicriteria approximation for the undirected multicut problem, unless $\mathsf{P} = \mathsf{NP}$.

Finally, we can use the technique of [7] to prove that undirected sparsest cut is APX-hard.