Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 142 (2009)

BOUNDS ON MONOTONE SWITCHING NETWORKS FOR DIRECTED CONNECTIVITY

AARON POTECHIN

ABSTRACT. We prove that any monotone switching network solving directed connectivity on N vertices must have size $N^{\Omega(\log N)}$

1. INTRODUCTION

L versus NL, the problem of whether non-determinism helps in logarithmic space bounded computation, is a longstanding open question in computational complexity. At present, only a few results are known. It is known that the problem is equivalent to the question of whether there is a log-space algorithm for the *directed connectivity* problem, namely given an N vertex directed graph G and pair of vertices s, t, find out if there is a directed path from s to t in G. In 1970, Savitch [9] gave an $O(\log^2 N)$-space deterministic algorithm for directed connectivity, thus proving that $NSPACE(g(n)) \subseteq DSPACE((g(n)^2))$ for every space constructable function g. In 1987 and 1988, Immerman [3] and Szelepcsenyi [10] independently gave an $O(\log N)$-space non-deterministic algorithm for directed *non-connectivity*, thus proving that $NL = co\text{-}NL$. For the problem of *undirected connectivity* (i.e. where the input graph G is undirected), a probabalistic algorithm was shown in 1979 using random walks by Aleliunas, Karp, Lipton, Lovász, and Rackoff [1], and in 2005, Reingold [8] gave a deterministic $O(\log N)$-space algorithm for the same problem, showing that undirected connectivity is in L. Trifonov [11] independently gave an $O(\lg N \lg\lg N)$ algorithm for undirected connectiivty.

In terms of monotone computation, in 1988 Karchmer and Wigderson [4] showed that any monotone circuit solving directed connectivity must have superlogarithmic depth, showing that monotone-$NC^1 \subsetneq$ monotone-L. In 1997 Raz and McKenzie [6] proved that monotone-$NC \neq$ monotone-P and for any i, monotone-$NC^i \neq$ monotone-NC^{i+1}.

So far, most of the work trying to show that $L \neq NL$ has been done using branching programs or the JAG model, introduced in [5] and [2] respectively. Instead, we explore trying to prove $L \neq NL$ using the switching network model, described in [7]. This model can be applied to any problem, and a general definition is given in Section 2. In this paper, we focus on switching networks solving directed connectivity.

The best way to describe what such a switching network is through an example, see Figure 1 and the accompanying explanation. A formal specialized definition is given below:

Definition 1.1. *A switching network solving directed connectivity on a set $V(G)$ of vertices with distinguished vertices s, t is a tuple $< G', s', t', \mu' >$ where G' is an undirected multi-graph with distinguished vertices s',t' and μ' is a labeling function such that:*
1. Each edge $e' \in E(G')$ has a label of the form $v_1 \to v_2$ or $\neg(v_1 \to v_2)$ for some vertices

Key words and phrases. L,NL,computational complexity, switching networks.

ISSN 1433-8092

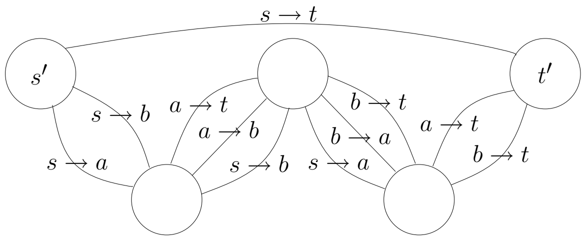

FIGURE 1. A switching network solving directed connectivity is an undirected multi-graph G' that takes a directed graph G and tells us if there is a path from s to t in G as follows: If an edge in G' has a label $a \to b$ for some vertices a and b in G, then we can take it if and only if the edge $a \to b$ is in G. Similarly, if an edge in G' has a label $\neg(a \to b)$, we can take it if and only if the edge $a \to b$ is not in G. Under these conditions, there is a path from s' to t' in G' if and only if there is a path from s to t in G.
In this figure, we have a switching network that solves directed connectivity when G has four vertices, s, t, a, and b. As needed, there is a path from s' to t' in G' if and only if there is a path from s to t in G. For example, if we have the edges $s \to a$, $a \to b$, and $b \to t$ in G, so there is a path from s to t in G, then in G', starting from s', we can take the edge labeled $s \to a$, then the edge labeled $a \to b$, then the edge labeled $s \to a$, and finally the edge labeled $b \to t$, and we will reach t'. If in G we have the edges $s \to a$, $a \to b$, $b \to a$, and $s \to b$ and no other edges, so there is no path from s to t, then in G' there is no edge that we can take to t', so there is no path from s' to t'.

$v_1, v_2 \in V(G)$.
2. Given a directed graph G with vertex set $V(G)$, there is a path in G' from s' to t' such that all of the labels are consistent with G, i.e. of the form e for some edge $e \in E(G)$ or $\neg e$ for some $e \notin E(G)$, if and only if there is a path from s to t in G.
We say that such a switching network solves directed connectivity on N vertices, where $N = |V(G)|$, and we take its size to be $|V(G')|$. A switching network solving directed connectivity is monotone if it has no labels of the form $\neg(v_1 \to v_2)$.

Notation: In this paper, we use lower case letters (i.e. a, e, f) to denote vertices, edges, and functions, and we use upper case letters (i.e. G, V, E) to denote graphs and sets of vertices and edges. We use unprimed symbols to denote vertices, edges, etc. in the directed graph G, and we use primed symbols to denote vertices, edges, etc. in the switching network G'.

1.1. **Our Results.** In Section 2, we give a proof that if there is no polynomial-sized switching network solving directed connectivity, then $L \neq NL$. Thus, our goal is to prove a superpolynomial lower size bound on switching networks solving directed connectivity. In this paper, we focus on showing lower size bounds for monotone switching networks solving directed connectivity.

We can view the vertices of a switching network solving directed connectivity as encoding how much we know about the directed graph G, where at s' we know nothing about G and at t' we know there is a path from s to t in G. When we move from one vertex in the switching network to another, it represents a change in our knowledge, which is allowed because the fact that we can make this move gives us information about G.

The key property of moving in switching networks is that everything is reversible. Thus, it is natural to start by restricting ourselves to simple states of knowledge and some basic reversible operations for getting from one state of knowledge to another.

In Section 3, we implement these ideas by defining a subclass of monotone switching networks solving directed connectivity, which we call certain-knowledge switching networks. We first show that certain-knowledge switching networks can capture a variant of Savitch's algorithm, which implies that there is a certain-knowledge switching network of size $N^{O(\log N)}$ solving directed connectivity. We then show that this is tight with the following theorem:

Theorem 1.2. *Any certain-knowledge switching network solving directed connectivity on N vertices has size at least* $N^{\Omega(\log N)}$.

In Section 4, we analyze general monotone switching networks solving directed connectivity. We give a useful simplification of monotone switching networks that can be accomplished by increasing the size of the switching network by a factor of at most N, and we show a theorem that in a weak sense reduces monotone switching networks to certain-knowledge switching networks.

In Section 5, we introduce a Fourier transformation technique. We then use this technique to prove an $\Omega(N^2)$ lower size bound on monotone switching networks solving directed connectivity, and we give a condition that is sufficient to prove a superpolynomial bound.

In Section 6, we give Fourier analogues of results in Sections 3 and 4 and use these to prove the above condition, thus proving a superpolynomial bound on monotone switching networks solving directed connectivity.

Finally, in Section 7, we modify and expand our techniques slightly to prove the main result:

Theorem 1.3. *Any monotone swtiching network solving directed connectivity on N vertices has size at least* $N^{\Omega(\log N)}$

1.2. **Proof Overview.** We now give a high level informal overview of the proof, ignoring details and subtleties.

The main idea involved in proving lower size bounds for monotone switching networks solving directed connectivity is as follows. Since G' solves directed connectivity, for every path P in G from s to t, there is a path P' in G' from s' to t' that uses only the edges of P. We show that this P' must include a vertex a'_P that gives significant information about P, i.e. there cannot be too many paths $P_1, P_2, \cdots$ in G such that each pair of paths P_i, P_j has very few vertices in common and all of these share the same vertex a' in G'. Then if we can find a large collection of paths such that each pair of paths has very few vertices in common, this will give a good lower bound on the number of vertices in G'.

In Section 3, we apply this approach to prove Theorem 1.2, that any certain-knowledge switching network solving directed connectivity on N vertices has size at least $N^{\Omega(\log N)}$. Lemma 3.8 shows if we have a path P' from s' to t' in G' that uses only the edges of some path P from s to t in G, then we can pick a vertex a'_P in P' which tells us at least $\log k$ of the vertices in P, where k is the length of P. Thus, if two paths P_1 and P_2 of length k in G have less than $\log k$ vertices in common, then a'_{P_1} cannot be the same as a'_{P_2}, so each such path gives a distinct vertex in G'. It is not hard to find a large collection of paths from s to t in G of length k such that each pair of paths has less than $\log k$ vertices in common, and this completes the proof.

However, the way that a'_P gives information about P in certain-knowledge switching networks is somewhat artificial and cannot be extended to general monotone switching networks solving directed connectivity. In Section 5, we introduce a fourier transformation technique and assign each vertex a' in G' a function $J_{a'} : \mathcal{C} \to \mathbb{R}$, where $\mathcal{C}$ is the set of all possible cuts of G.

In Section 6, we prove Theorem 6.1, showing that for each directed path P in G we can find a function $g_P : \mathcal{C} \to \mathbb{R}$ such that if P' is a path from s' to t' using only the edges of P, then $\sum_{a' \in V(P')} |J_{a'} \cdot g_P|$

is relatively large. Moreover, if P_1 and P_2 have very few vertices in common, then g_{P_1} and g_{P_2} are orthogonal. In this way, the vertices of P' give significant information about P. As shown in Theorem 5.23, this is sufficient to show a superpolynomial lower size bound.
Finally, in Section 7, we refine the above arguments to prove Theorem 1.3, that any monotone switching networks solving directed connectivity on N vertices has size at least $N^{\Omega(\log N)}$.

2. Switching-and-rectifier Networks and Switching Networks

In this section, we give a proof that if there is no polynomial-sized switching network solving directed connectivity, then $L \neq NL$. Although the results in this section are not new, we include them for the sake of completeness.

To see how switching networks capture logspace computation, it is useful to first look at how a related model, switching-and-rectifier networks, captures non-deterministic logspace computation. Accordingly, we give the following definition from [7]:

Definition 2.1. *A switching-and-rectifier network is a tuple $< G, s, t, \mu >$ where G is a directed graph with distinguished vertices s, t and μ is a labeling function that associates with some edges $e \in E(G)$ a label $\mu(e)$ of the form $x_i = 1$ or $x_i = 0$ for some i between 1 and n. We say that this network computes the function $f : \{0,1\}^n \to 0,1$, where $f(x) = 1$ if and only if there is a path from s to t such that each edge of this path either has no label or has a label that is consistent with x.*
We take the size of a switching-and-rectifier network to be $|V(G)|$, and for a function $f : \{0,1\}^n \to 0,1$, we define $RS(f)(n)$ to be size of the smallest switching-and-rectifier network computing f.

Proposition 2.2. *If $f \in NSPACE(g(n))$ where $g(n)$ is at least logarithmic in n, then $RS(f)(n)$ is at most $2^{O(g(n))}$*

Proof. Let T be a non-deterministic Turing machine with a read/write tape and an input tape computing f using $g(n)$ space. To create the corresponding switching-and-rectifier network, first create a vertex v_j for each possible configuration c_j of T, where a configuration includes the state of the Turing machine and all of the bits on the read/write tape. Now add edges in the obvious way, adding an edge from v_{j_1} to v_{j_2} if the Turing machine could go from c_{j_1} to c_{j_2}. If there is a dependence on the input, put the appropriate label on this edge. Finally, merge all accepting configuarations into one vertex t. It is easily verified that the resulting switching-and-rectifier network computes f and has size at most $2^{O(g(n))}$, as needed. □

We give the general definition of switching networks below.

Definition 2.3. *A switching network is a tuple $< G', s', t', \mu' >$ where G' is an undirected graph with distinguished vertices s', t' and μ' is a labeling function that associates with each edge $e' \in E(G')$ a label $\mu(e')$ of the form $x_i = 1$ or $x_i = 0$ for some i between 1 and n. We say that this network computes the function $f : \{0,1\}^n \to 0,1$, where $f(x) = 1$ if and only if there is a path from s' to t' such that each edge of this path has a label that is consistent with x.*
We take the size of a switching network to be $|V(G')|$, and for a function $f : \{0,1\}^n \to 0,1$, we define $S(f)(n)$ to be size of the smallest switching network computing f.

Remark 2.4. *Note that switching networks are the same as switching-and-rectifier networks except that all edges are now undirected and we cannot have edges with no label. However, allowing edges with no label does not increase the power of switching networks, as we can immediately contract all such edges to obtain an equivalent switching network where each edge is labeled. Also, note that switching networks solving directed connectivity are just switching networks where the input is taken to be the adjacency matrix of a directed graph G.*

Theorem 2.5. *If $f \in DSPACE(g(n))$ where $g(n)$ is at least logarithmic in n, then $S(f)(n)$ is at most $2^{O(g(n))}$.*

Proof. We can start by treating the Turing machine as non-deterministic and taking the switching-and-rectifier network as in Proposition 2.2. Now note that for a given input x, since the Turing machine is deterministic, each vertex has at most one edge going out from it. This means that G has the structure of a forest where the root of each tree is either t, a vertex corresponding to a rejecting configuration, or a directed cycle. But then whether or not there is a path from s to t is unaffected by making all of the edges undirected. Thus, we can obtain a switching network that computes f simply by making all of the edges of this switching-and-rectifier network undirected. The result follows immediately. □

Corollary 2.6. *If there is no switching network of polynomial size solving directed connectivity, then $L \neq NL$.*

3. certain-knowledge Switching Networks

In this section, we consider a subclass of monotone switching networks solving directed connectivity, which we call certain-knowledge switching networks, where we can assign each vertex $a' \in V(G')$ a simple state of knowledge and there are simple reversible rules for moving from one state of knowledge to another. We show that certain-knowledge switching networks can capture a variant of Savitch's algorithm, so there is a certain-knowledge switching network of size at most $N^{O(\log N)}$ solving directed connectivity on N vertices. We then prove Theorem 1.2, showing that any certain-knowledge switching network solving directed connectivity on N vertices has size at least $N^{\Omega(\log N)}$, and this bound is tight.

We make the following definitions:

Definition 3.1. *A knowledge set K is a directed graph with $V(K) = V(G)$, and we represent K by the set of its edges.*

Given a knowledge set K, we can form a knowledge set $\bar{K}$ as follows:
If there is no path from s to t in K, then $\bar{K} = \{v_1 \to v_2 :$ there is a path from v_1 to v_2 in $K\}$.
If there is a path from s to t in K, then $\bar{K}$ is the complete directed graph on $V(G)$.
Call $\bar{K}$ the transitive closure of K.

Each transitive closure represents an equivalence class of knowledge sets. We say $K_1 = K_2$ if $\bar{K_1} = \bar{K_2}$ and we say $K_1 \subseteq K_2$ if $\bar{K_1} \subseteq \bar{K_2}$.

Remark 3.2. *We will try to label each vertex a' in the switching network with a knowledge set $K_{a'}$ so that if $v_1 \to v_2 \in K_{a'}$ and there is a path from s' to a' in G' using only edges in G, then we know that either there is a path from v_1 to v_2 in G or there is a path from s to t in G, in which case we do not care which other paths are in G. In this way, $K_{a'}$ represents our knowledge about G when we are at the vertex a'.*

If there is no path from s to t in K, then K and $\bar{K}$ represent exactly the same knowledge about G, so they are equivalent. If $K_{a'}$ contains a path from s to t, then if there is a path from s' to a' in G' using only edges in G, we know there is a path from s to t in G. Thus, we may as well merge a' and t'. To do this, we make all knowledge sets with a path from s to t equivalent by giving them the same transitive closure, the complete directed graph on $V(G)$.

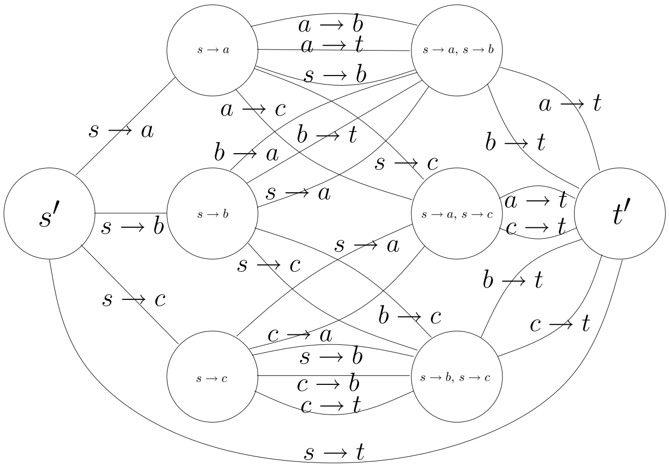

FIGURE 2. A certain-knowledge switching network that solves directed connectivity with five vertices, s, t, a, b, and c. The label inside each vertex represents the K for that vertex.

Remark 3.3. *The statement $K_1 \subseteq K_2$ does not imply that every edge in K_1 is in K_2. For example, if $K_1 = \{s \to a, s \to b\}$ and $K_2 = \{s \to a, a \to b\}$, then $\bar{K_1} = K_1$ and $\bar{K_2} = \{s \to a, a \to b, s \to b\}$, so $\bar{K_1} \subseteq \bar{K_2}$ and thus $K_1 \subseteq K_2$. It is best to think of the statement $K_1 \subseteq K_2$ as saying that the knowledge K_1 respresents is included in the knowledge K_2 respresents.*

We would like to label each vertex in the switching network with a knowledge set. In order for these labels to be meaningful, we must know that for any $a', b' \in V(G')$ and $v_1, v_2 \in V(G)$, if we are at a' and use an edge with label $v_1 \to v_2$ to reach vertex b', then knowing there is a path from $v_1 \to v_2$ in G and having the knowledge represented by $K_{a'}$ are sufficient to give the knowledge represented by $K_{b'}$. This leads naturally to the following definition:

Definition 3.4. *We say a monotone switching network solving directed connectivity is a certain-knowledge switching network if we can assign a $K_{a'}$ to each vertex $a' \in V(G')$ such that the following conditions hold:*
1. $K_{s'} = \{\}$ and $K_{t'} = \{s \to t\}$.
2. If there is an edge with label $v_1 \to v_2$ between vertices a' and b', then
$K_{b'} \subseteq K_{a'} \cup \{v_1 \to v_2\}$ and $K_{a'} \subseteq K_{b'} \cup \{v_1 \to v_2\}$

Proposition 3.5. *The condition that $K_{b'} \subseteq K_{a'} \cup \{v_1 \to v_2\}$ and $K_{a'} \subseteq K_{b'} \cup \{v_1 \to v_2\}$ is equivalent to the condition that we can obtain $K_{b'}$ from $K_{a'}$ using only the following reversible operations:*
Operation 1: Add or remove $v_1 \to v_2$.
Operation 2: If $v_3 \to v_4, v_4 \to v_5$ are both in K, add or remove $v_3 \to v_5$ from K.
Operation 3: If $s \to t$ is in K, add or remove any edge except $s \to t$ from K
If this condition is satisfied, we say we can get from $K_{a'}$ to $K_{b'}$ with the edge $v_1 \to v_2$.

Remark 3.6. *The operations in Proposition 3.5 are not a good starting point for definitions, but they are very effective for analyzing certain knowledge switching networks solving directed connectivity. The reader would do very well to understand these operations thoroughly. In particular, note that each of these operations is reversible. This reflects the undirected nature of the switching network; we can undo any move that we make.*

3.1. **certain-knowledge switching networks and Savitch's Theorem.** While this model is restricted, it is not trivial. In particular, it is capable of capturing the following variant of Savitch's algorithm:

Savitch's algorithm works as follows. To check if there is a path of length at most k between vertices s and t, we go through all of the possible midpoints m and recursively check whether there is a path of length at most $\frac{k}{2}$ from s to m and whether there is a path of length at most $\frac{k+1}{2}$ from m to t. If $k = 1$, then we check the adjacency matrix of the graph directly. There is a path from s to t in G if and only if both subpaths are in G for some m.

This algorithm reaches depth at most $\log N$ and stores one vertex at each level, so it requires $O((\log N)^2)$ space.

Savitch's algorithm implicitly keeps track of a knowledge set K of which paths are in G. Each time the algorithm checks for and finds a path, it adds it to K. If we check for and find a path of length 1 between v_1 and v_2 in the adjacency matrix, we can add $v_1 \to v_2$ to K using operation 1 of 3.5. If we check for and find a longer path, we have found the subpaths from v_1 to m and from m to v_2 for some m, so we can add $v_1 \to v_2$ to K using operation 2 of Proposition 3.5.

The problem is that in checking for a longer path from v_1 to v_2, after finding the paths from v_1 to m and m to v_2, the original algorithm only keeps the path from v_1 to v_2 in K and discards the paths from v_1 to m and from m to v_2 from K. Similarly, if a path from v_1 to m is found but no path from m to v_2 is found, the algorithm immediately discards $v_1 \to m$ from K. This is not allowed under the rules of Proposition 3.5, as discarding information is not reversible.

We fix this by modifying the algorithm so that whenever the algorithm wants to remove a path $v_1 \to v_2$ of length greater than 1 from K, it must first check for and find this path again. This gives us the subpaths $v_1 \to m$ and $m \to v_2$ so that we can remove the path $v_1 \to v_2$ using the operations in Proposition 3.5.

With this modification, whenever we go from one K to another, we are using only the operations in Proposition 3.5. Moreover, at each level, we add at most 3 paths, $v_1 \to m$, $m \to v_2$, and $v_1 \to v_2$, so there are at most $N^{(O(\log N))}$ possible K that we could reach.

We can create a certain-knowledge switching network from this by creating one vertex a'_K for each possible K we could reach and adding all possible labeled edges that satisfy condition 2 of Definition 3.4. When we run the modified Savitch's algorithm, we can follow its progress in the switching network, and if the algorithm finds a path from s to t, we will be at t'. Since the algorithm finds all possible paths, this certain-knowledge switching network successfully solves directed connectivity on N vertices. This immediately gives the following thoerem:

Theorem 3.7. *There is a certain-knowledge switching network of size $N^{O(\log N)}$ that solves directed connectivity on N vertices.*

3.2. **Lower size bound on certain-knowledge switching networks solving directed connectivity.** We now prove Theorem 1.2, showing that this bound is tight.

Theorem 1.2. *Any certain-knowledge switching network that solves directed graph connectivity on N vertices has size at least $N^{\Omega(\log N)}$.*
We will first show that the result follows from the following lemma. We will then prove the lemma.

Lemma 3.8. *If the input consists of a path P in the directed graph $s \to v_1, v_1 \to v_2, \cdots, v_{2^k} \to t$ and no other edges, then any path P' in G' from s' to t' must pass through at least one vertex a' such that $K_{a'} \neq K_{t'}$ and the union of the endpoints of the edges in $K_{a'}$ is a subset of $\{s, v_1, v_2, \cdots, v_{2^k}, t\}$ that contains at least $k+1$ of $v_1, v_2, \cdots, v_{2^k}$.*

Proof of Theorem 1.2 using Lemma 3.8. For any prime p, if $k < p$, if we take all of the polynomials in $Z_p[x]$ of degree at most k, then any two distinct polynomials will have at most k values in common. Thus, if $p > 2^k$, given a polynomial $f(x)$ of degree at most k, if we take v_i to be vertex $p \cdot (i-1) + f(i)$ of G for $i = 1$ to 2^k, then the corresponding paths will share at most k vertices in common.

However, by Lemma 3.8, we can associate a vertex in G' to each such path, and no two such paths can share the same vertex. Hence, there are at least p^{k+1} vertices in G', and we can do this as long as $N \geq p^2 + 2$ and $k < \log p$. The result follows immediately. □

Proof of Lemma 3.8.

Definition 3.9. *Call the vertices $L = \{v_1, \cdots, v_{2^{k-1}}\}$ the left half of P and the vertices $R = \{v_{2^{k-1}+1}, \cdots, v_{2^k}\}$ the right half of P.*

Definition 3.10. *K satisfies the lemma for the left half if the union of the endpoints of the edges in K contains at least k of the vertices in L.*
We define satisfying the lemma for the right half in a similar way.

We begin by giving an informal version of the proof. We prove this lemma by induction. If there is a path $P' = s' \to v'_1 \to v'_2 \to \cdots \to v'_r \to t'$ in G' using only the edges in P, consider the sequence $K_{s'}, K_{v'_1}, \cdots K_{v'_r}, K_{t'}$. We get from each K to the next K using only the operations given by Proposition 3.5. Thus, we are trying to use these operations to obtain an edge from s to t (which represents a path from s to t in G).

To obtain an edge from s to t using only these operations, it is necessary (but not sufficient) to first obtain an edge from s to a vertex $r \in R \cup \{t\}$ and an edge from a vertex $l \in L \cup \{s\}$ to t. By the inductive hypothesis, to obtain an edge from s to a vertex $r \in R \cup \{t\}$, we must first reach a $K_{v'_i}$ that satisfies the lemma for the left half. If the union of the endpoints of the edges in $K_{v'_i}$ contains even one vertex in R, $K_{v'_i}$ will satisfy the lemma. If not, then either $K_{v'_i}$ already has an edge from a vertex $l \in L \cup \{s\}$ to t or there is neither an edge from $l \in L \cup \{s\}$ to a vertex $r \in R \cup \{t\}$ nor an edge from a vertex $r \in R$ to t, so there is no progress towards obtaining an edge from a vertex $l \in L \cup \{s\}$ to t. A similar argument holds if we try to obtain an edge from a vertex $l \in L \cup \{s\}$ to t.

If in trying to obtain an edge from s to a vertex $r \in R \cup \{t\}$ or an edge from a vertex $l \in L \cup \{s\}$ to t, when we reach such a $K_{v'_i}$, we always have no progress towards obtaining the other required edge, then we will never obtain an edge from $s \to t$. Thus, we can only obtain an edge from s to t if we reach such a $K_{v'_i}$ and the other required edge has already been obtained. But this means that we have reached a $K_{v'_j}$ such that either $K_{v'_j}$ contains an edge from s to a vertex $r \in R$ and the union of the endpoints of the edges in $K_{v'_j}$ does not contain any vertex in L or $K_{v'_j}$ contains an edge from a vertex $l \in L$ to t and the union of the endpoints of the edges in $K_{v'_j}$ does not contain any vertex in R.

If we could reach such a $K_{v'_j}$, we would indeed be close to obtaining the path $s \to t$. However, reaching such a $K_{v'_j}$ without going through a $K_{v'_i}$ satisfying the lemma is impossible for the following reason:
When we first obtain an edge from s to a vertex $r \in R$, we must have the edges $s \to l$ and $l \to r$

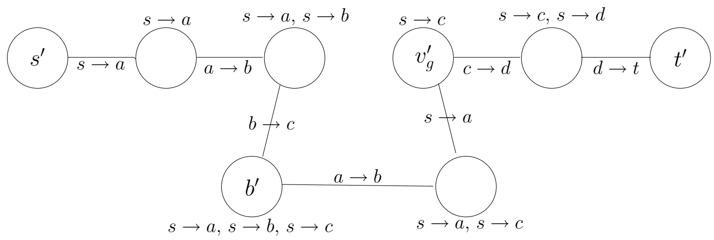

FIGURE 3. An example of a possible path P' in G' from s' to t' corresponding to an input which only has the edges $s \to a$, $a \to b$, $b \to c$, $c \to d$, and $d \to t$. The K for each vertex is given above or below that vertex. In this example, if we project on the left half, $K_{v'_g} = \{\}$ and $K_{b'} = \{s \to a, s \to b\}$, and we can get from $K_{b'}$ to $K = \{s \to t\}$ with the edge $b \to c$ (as $c = t$). Also note that in this example, $a' = b'$ and for every v' in the path between b' and v'_g, $K_{v'}$ includes the left-jumping edge $s \to c$.

for some $l \in L$. Removing these edges is just as difficult as obtaining them, which means that to remove them, we must pass through a $K_{v'_i}$ such that $K_{v'_i}$ satisfies the lemma for the left half. But if we also hold on to the edge $s \to r$, then $K_{v'_i}$ also contains a vertex in R, so $K_{v'_i}$ satisfies the lemma. A similar argument holds if we try to obtain an edge from a vertex $l \in L$ to t.

We now make this argument rigorous:

First add all labeled edges allowed by condition 2 of Definition 3.4 to G'. Now note that the only way to introduce vertices besides s, t, and $v_1, \cdots, v_{2^k}$ is through operation 3 of Proposition 3.5. But if we are at a point where we could use operation 3, then we could instead immediately go to t'. Thus, we may assume P' does not use operation 3, and we do not need to worry about any vertices in G except s, t, and $v_1, \cdots, v_{2^k}$.

We now introduce several useful definitions:

Definition 3.11. *Define reducing to the left half as follows:*
1. Make all vertices in the right half of G equal to t. This applies to the K of all vertices in G'.
We define reducing to the right half in a similar way.

Proposition 3.12. *If we reduce to either half, G' still satisfies both conditions of Definition 3.4.*

Definition 3.13. *Define projecting on the left half as follows:*
1. Remove t' and all vertices with $K = K_{t'}$ from G'.
2. For all vertices $v' \in V(G')$, if we can get from $K_{v'}$ to $K_{t'}$ with the edge $v_1 \to v_2$, then remove all edges with label $v_1 \to v_2$ that are incident with v'.
3. Make all vertices in the right half of P equal to t. This applies to the K of all vertices in G'.
4. For each K, remove the edge $\{s \to t\}$ if it is there.
We define projecting on the right half in a similar way.

Proposition 3.14. *If we project on either half, G' still satisfies condition 2 of Definition 3.4.*

Remark 3.15. *Reducing to the left half or the right half allows us to focus on the process of obtaining an edge across that half.*

If we have an edge from s to a vertex in the right half, this cannot help us add or remove edges *with an endpoint in the left half unless operation 3 of Proposition 3.5 is used. The first two steps of projecting on the left half eliminate this possibility. Thus, projecting on the left half allows us to focus on the process of adding or removing edges with an endpoint in the left half regardless of whether or not we have an edge that crosses the left half. Projecting on the right half has a similar effect.*

We prove Lemma 3.8 by induction. The base case $k = 0$ is trivial. Assume the lemma is true for $k - 1$. We will show that it is impossible to have a path P' in G' from s' to a vertex v'_g such that $K_{v'_g}$ satisfies any of the following three conditions unless P' passes through a vertex a' such that $K_{a'}$ satisfies the lemma:

1. $K_{v'_g} = K_{t'}$.
2. $K_{v'_g}$ has an edge from s to a vertex in the right half, and it has no edges with an endpoint in the left half. $K_{v'_g} \neq K_{t'}$.
3. $K_{v'_g}$ has an edge from a vertex in the left half to t, and it has no edges with an endpoint in the right half. $K_{v'_g} \neq K_{t'}$.

Assume there is a path P' in G' from s' to a vertex v'_g such that $K_{v'_g}$ is of type 1 and P' does not pass through a vertex that satisfies the lemma or has a K of type 1, 2, or 3.

Let b' be the last vertex on P' before v'_g is reached such that for one of the left half or the right half, if we reduce to that half, then $K_{b'}$ satisfies the lemma for that half.

We may assume without loss of generality that it is the right half. If $K_{b'}$ does not satisfy the lemma, then there are no edges in $K_{b'}$ with an endpoint in the left half. Now reduce to the left half. $K_{b'} = \{s \to t\}$ or $K_{b'} = \{\}$. If $K_{b'} = \{s \to t\}$, then $K_{b'}$ was originally of type 1 or 2. Contradiction. $K_{b'} = \{\}$. But $K_{t'} = \{s \to t\}$. By the inductive hypothesis, there must be a vertex a' on the path from b' to v'_g such that $K_{a'}$ satisfies the lemma for the left half. But this contradicts the definition of b'. Contradiction.

The only case remaining is if b' does not exist. However, reducing to either half it is clear that this is impossible.

Thus, it is impossible to reach a vertex whose K is of type 1 without first going through a vertex that satisfies the lemma or has a K of type 1, 2, or 3.

Assume there is a path P' from s' to a vertex v'_g in G' with a K of type 2 that does not pass through a vertex that satisfies the lemma or has a K of type 1, 2, or 3.

If we project on the left half and a vertex in P' is removed, this vertex had a K of type one. Contradiction. If an edge from v'_1 to v'_2 in this path is deleted, then we could have instead gone directly from v'_1 to t', which would give us a path from s' to t' that does not pass through a vertex that satisfies the lemma or has a K of type 1, 2, or 3. From the above, this is impossible. Thus, the entire path is preserved when projecting on the left half. This also implies that we are only using operations 1 and 2 of Proposition 3.5.

Call an edge from s to a vertex in the right half (this vertex cannot be t) a left-jumping edge.

Let $P' = s' \to v'_1 \to v'_2 \to \cdots \to v'_r \to v'_g$. We can start with $K = K_{s'}$ and then go from each $K_{v'_i}$ to $K_{v'_{i+1}}$ using only operations 1 and 2 in Proposition 3.5. Choose one possible sequence of such operations and look at the last time in that we add or remove a left-jumping edge from K without having a shorter left-jumping edge. We know that we do this at least once because $K_{v'_g}$ has a left-jumping edge. Note that the only way to add or remove a left-jumping edge is to use operation 2, so at this point we must have edges $s \to v_i$ and $v_i \to v_j$, where $j > 2^{k-1}$. If $s \to v_i$ is a left-jumping edge, then we have a shorter left-jumping edge. Thus, we may assume $i \leq 2^{k-1}$. At this point, if we project on the left half, we have a K that includes $s \to t$. This occurs in the middle of a transition between some vertices v'_1 and v'_2 using some edge $v_l \to v_{l+1}$, which implies that if we project on the left half, we can go from $K_{v'_1}$ or $K_{v'_2}$ to $K_{t'}$ using the edge $v_l \to v_{l+1}$. Let $b' = v'_2$. $b' \neq v'_g$.

Projecting on the left half, $K_{v'_g} = \{\}$ and we can get from $K_{b'}$ to $K_{t'}$ with the edge $v_l \to v_{l+1}$. By the inductive hypothesis, there must be a vertex a' (which may be equal to b' but cannot equal v'_g) on P' from v'_g to b' such that $K_{a'}$ satisfies the lemma for the left half.

Since once we reach b' we never remove the shortest left-jumping edge that we have and $K_{v'_g}$ has a left-jumping edge, all vertices from b' onwards also have at least one left-jumping edge. Thus, a' satisfies the lemma.

Thus, it is impossible to reach a vertex whose K is of type 2 without first reaching a vertex that satisfies the lemma or has a K of type 1, 2, or 3. Similar logic applies if we want to get to a K of type 3, and this completes the proof. □

4. Preliminary Results on Monotone Switching Networks

In this section, we begin our analysis of general monotone switching networks solving directed connectivity. We give a definition for monotone switching networks solving directed connectivity that generalizes the definition of certain-knowledge switching networks. We then give a useful simplification of monotone switching networks solving directed connectivity that can be accomplished by increasing the size of the switching network by a factor of at most N. Finally, we prove Theorem 4.6, showing that in some sense, monotone switching networks solving directed connectivity can be reduced to certain-knowledge switching networks.

Not every monotone switching network solving directed connectivity is a certain-knowledge switching network. The monotone switching networks shown in Figures 1 and 4 are not certain-knowledge switching networks. The reason why they are not certain-knowledge switching networks is because at the vertices in these switching networks, a depth-1 monotone formula (only ANDs) is insufficient to describe our knowledge about G. Instead, we need a depth-2 monotone formula (ORs of ANDs).

Definition 4.1. *A state of knowledge J is a set $\{K_1, \cdots, K_m\}$ of knowledge sets.*
Let $J_1 = \{K_{11}, K_{12}, \cdots, K_{1m}\}$ and let $J_2 = \{K_{21}, K_{22}, \cdots, K_{2n}\}$. We say $J_1 \subseteq J_2$ if for every j there exists a i such that $K_{1i} \subseteq K_{2j}$.
We say $J_1 = J_2$ if $J_1 \subseteq J_2$ and $J_2 \subseteq J_1$.
Let $J = \{K_1, \cdots, K_m\}$. Define $J \cup \{v_1 \to v_2\}$ to be $\{K_1 \cup \{v_1 \to v_2\}, \cdots, K_m \cup \{v_1 \to v_2\}\}$.
We say that we can get from J_1 to J_2 with the edge $v_1 \to v_2$ if
$J_2 \subseteq J_1 \cup \{v_1 \to v_2\}$ and $J_1 \subseteq J_2 \cup \{v_1 \to v_2\}$

Remark 4.2. *The state of knowledge $J = \{K_1, \cdots, K_m\}$ represents knowing that the paths in K_1 are in G OR the paths in K_2 are in G OR $\cdots$ OR the paths in K_m are in G OR there is a path*

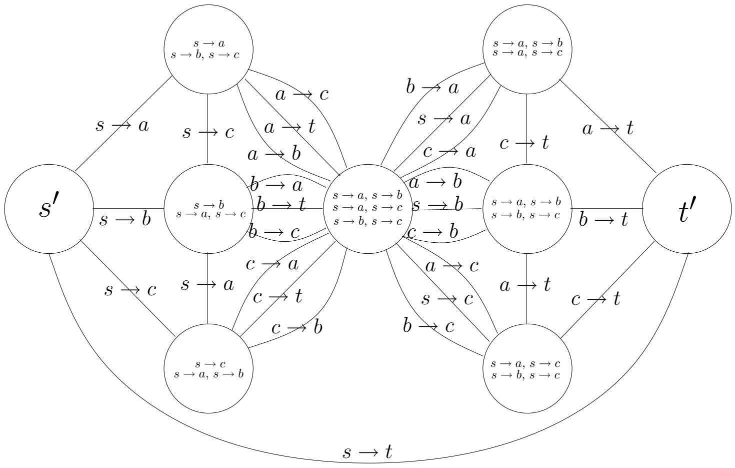

FIGURE 4. A monotone switching network that solves directed connectivity with five vertices, s, t, a, b, and c. The label inside each vertex gives the J for that vertex, with each line corresponding to one of its K.

from s to t in G. Thus, J is characterized by its least informative K. The condition for $J_1 \subseteq J_2$ ensures that J_2 represents at least as much information about G as J_1.

Proposition 4.3. *We can get from J_1 to J_2 with the edge $v_1 \to v_2$ if and only if for every i there exists a j such that $K_{2j} \subseteq K_{1i} \cup \{v_1 \to v_2\}$ and for every j there exists a i such that $K_{1i} \subseteq K_{2j} \cup \{v_1 \to v_2\}$.*
To do this, the following 4 reversible operations are sufficient:
Operation 1: Add or remove $v_1 \to v_2$ from any K_i.
Operation 2: If $v_3 \to v_4, v_4 \to v_5$ are both in K_i, add or remove $v_3 \to v_5$ from K_i.
Operation 3: If $s \to t$ is in K_i, add or remove any path except $s \to t$ from K_i
Operation 4a: If $K_i \subseteq K_{i'}$ for some i, i', remove $K_{i'}$ from J.
Operation 4b: If for some knowledge set K and some i, $K_i \subseteq K$, add K to J.

Proposition 4.4. *For any monotone switching network solving directed connectivity, we can assign a $J_{a'}$ to each $a' \in V(G')$ so that the following properties hold:*
1. $J_{s'} = \{\{\}\}$ and $J_{t'} = \{\{s \to t\}\}$.
2. If there is an edge with label $v_1 \to v_2$ between a' and b', then it is possible to get from $J_{a'}$ to $J_{b'}$ with the edge $v_1 \to v_2$.

Proof. For each vertex $a' \in V(G')$, take $J_{a'}$ to be the set of all K such that using the edges of K, it is possible to reach a' from s' in G'. It is easy to check that both of the above properties are satisfied. □

We will now describe a useful simplification for monotone switching networks that can be accomplished with an increase of at most a factor of N in the size of the network.

Theorem 4.5. *If there is a monotone switching network (G', s', t', μ') solving directed connectivity on N vertices, then there is a monotone switching network (G'', s'', t'', μ'') with $|V(G'')| \leq N|V(G')|$ such that for any vertex a'' of G'', for any K in $J_{a''}$, K consists only of edges of the form $s \to v$ for some $v \in V(G)$.*

Proof. We construct G'' by taking N copies of G' and making the s' of each copy equal to the t' of the previous copy. We take s'' to be the s' of the first copy and t'' to be the t' of the last copy.

Now for a vertex a'' we construct $J_{a''}$ as follows. For a given path from s'' to a'' in G'', create a K for that path as follows:

1. Let e_i be the ith edge in G that this path uses. Edges can be repeated.
2. Start with a set $X_0 = \{S\}$ of vertices in G.
3. If e_i is the edge from v to w, let $X_i = X_{i-1}$ if $v \notin X_{i-1}$ and let $X_i = X_{i-1} \cup w$ if $v \in X_{i-1}$. Let X be the set obtained after taking the final edge in the path.
4. Set $K = \cup_{v \in X}\{s \to v\}$.

Now take $J_{a''}$ to be the set of all such K.

It is easy to check that G'' satisfies property 2 of Proposition 4.4. To see that G'' satisfies property 1, note that for each time a path goes through a copy of G', at least one new vertex must be added to X. Thus, for any path from s'' go t'', we must have that X contains every vertex including t. Thus, $J_{t''} = \{\{s \to t\}\}$, as needed. ☐

Finally, we prove a theorem that shows that in some sense, monotone switching networks can be reduced to certain-knowledge switching networks. Although this theorem is not strong enough to prove any lower size bounds, the reduction used in this theorem is very deep and will play a crucial role in Section 6.

Theorem 4.6. *For any monotone switching network, if there is a path in G' from s' to t' using only edges that have a label in a subset E of $E(G)$, then there is a sequence of K_i, $0 \leq i \leq m$ with the following properties:*
1. $K_0 = \{\}$. $K_m = \{s \to t\}$.
2. For all i, there exists an edge $e_i \in E$ such that it is possible to go from K_i to K_{i+1} using the edge e_i and the three given operations.
3. For all i, there exists a vertex a'_i on this path such that K_i is the union of some subset of $J_{a'_i}$.

Proof. For each edge e' in this path, do the following:

Let e be the label of e', and let a' and b' be the endpoints of e'. For each $K_{a'i} \in J_{a'}$, there is a $K_{b'j} \in J_{b'}$ such that $K_{b'j} \subseteq K_{a'i} \cup e$. Draw an orange arrow from each $K_{a'i}$ to one such $K_{b'j}$. Similarly, for each $K_{b'j} \in J_{b'}$, there is a $K_{a'i} \in J_{a'}$ such that $K_{a'i} \subseteq K_{b'j} \cup e$. Draw an orange arrow from each $K_{b'j}$ to one such $K_{a'i}$. We now have a set of directed cycles with tails. Take one representative $K_{a'i}$ and one representative $K_{b'j}$ from each directed cycle.

Now draw a black arrow from each $K_{a'i}$ to the unique $K_{b'j}$ such that there is a path of orange arrows from $K_{a'i}$ to $K_{b'j}$ and $K_{b'j}$ is a representative of a cycle. Similarly, draw a black arrow from each $K_{b'j}$ to the unique $K_{a'i}$ such that there is a path of orange arrows from $K_{b'j}$ to $K_{a'i}$ and $K_{a'i}$ is a representative of a cycle.

Looking only at the black arrows, the following properties hold:
1. If there is an arrow going from $K_{a'i}$ to $K_{b'j}$, then $K_{b'j} \subseteq K_{a'i} \cup e$.
2. If there is an arrow going from $K_{b'i}$ to $K_{a'j}$, then $K_{a'j} \subseteq K_{b'i} \cup e$.
3. If we there are arrows going both ways between $K_{a'i}$ and $K_{b'j}$, we can get from $K_{a'i}$ to $K_{b'j}$ with e.

Finally, for each vertex a', order the $K_{a'}$.

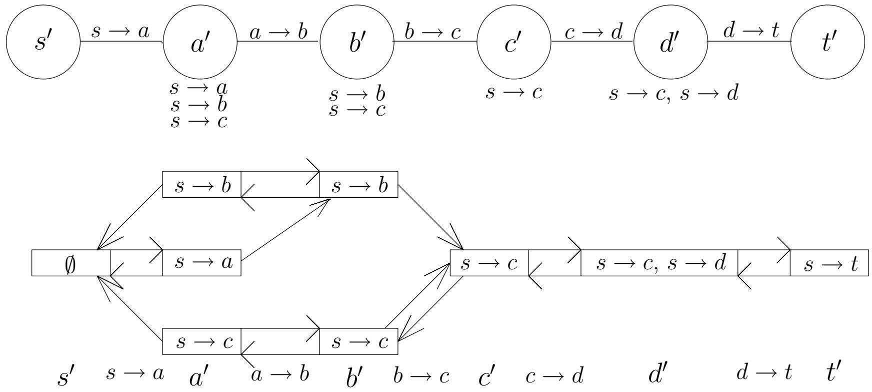

FIGURE 5. This is an illustration of the ideas used in the proof of Theorem 4.6. Above, we have the original path from s' to t', where the J for each vertex is given below that vertex. Below, we have the relations between all of the K, where each box has one K. To get from s' to t' we have the following sequence of K_i: $K_0 = \{\}$ at s', $K_1 = \{s \to a\}$ at a', $K_2 = \{s \to a, s \to b\}$ at a', $K_3 = \{s \to b\}$ at a', $K_4 = \{s \to b\}$ at b', $K_5 = \{s \to b, s \to c\}$ at b', $K_6 = \{s \to b, s \to c\}$ at a', $K_7 = \{s \to a, s \to b, s \to c\}$ at a', $K_8 = \{s \to a, s \to c\}$ at a', $K_9 = \{s \to c\}$ at a', $K_{10} = \{s \to c\}$ at b', $K_{11} = \{s \to c\}$ at c', $K_{12} = \{s \to c, s \to d\}$ at d', $K_{13} = \{s \to t\}$ at t'

Now we will try to travel from s' to t' on this path while always keeping a subset of the K of the vertex we are on. When attempting to go from a vertex a' to a vertex b', we will allow only the following operation:

If every $K_{a'i}$ we have is the representative of a cycle as described above, then travel to b' and replace each $K_{a'i}$ with the corresponding $K_{b'j}$. If not, then do the following:
1. For each $K_{a'i}$ we have that is the representative of a cycle, replace it by the corresponding $K_{b'j}$.
2. Take the earliest $K_{a'i}$ we have that is not the representative of a cycle. Take the $K_{b'j}$ that the arrow going from this $K_{a'i}$ is pointing to. Remove this $K_{b'j}$ if it is in our set and add it if it is not.
3. For each $K_{b'j}$ we have, replace it by the corresponding $K_{a'i}$.

Note that for each of these steps, we can get from the union of the K before that step to the union of the K afterwards with some edge $e \in E$. Thus, if we use only this operation, the resulting sequence of K_i will obey the given rules.

Also note that each such operation is reversible and if we are at a vertex in the middle of the path, we have exactly two choices for where to go next regardless of which subset we have. However, if we are at s' or t', our subset is fixed and we only have one choice for where to go next. Thus, we must be able to get from s' to t' using only the given operation, and this completes the proof. □

5. Fourier Analysis on Monotone Switiching Networks

Unfortunately, the above results are insufficient to prove a superpolynomial lower size bound on monotone switching networks solving directed connectivity. To prove a good lower size bound, more sophisticated techniques are needed. In this section, we introduce a fourier transformation technique for monotone switching networks solving directed connectivity. We then use this technique to prove an $\Omega(N^2)$ lower size bound. Finally, we give a condition which is sufficient to prove a superpolynomial lower size bound.

An alternate way of solving directed connectivity is to look at cuts of G. There is a path from s to t if and only if there is no cut $C = (V_1, V_2)$ such that $s \in V_1$, $t \in V_2$, and there is no edge from a vertex in V_1 to a vertex in V_2. Thus, instead of describing each state of knowledge J in terms of paths in G, we can describe each J as a function of the cuts of G. We do this below.

5.1. **Definitions and Basic Properties.**

Definition 5.1. *We define an s-t cut (below we use cut for short) of G to be a subset C of $V(G)$ such that $s \in C$ and $t \notin C$. We denote the complement of C by C^c, and we say an edge $v_1 \to v_2$ crosses C if $v_1 \in C$ and $v_2 \in C^c$.*
Let $\mathcal{C}$ denote the set of all cuts C. $|\mathcal{C}| = 2^{N-2}$.

Given a state of knowledge J, we want $J(C)$ to be 1 if given the information J represents we know that G contains an edge crossing C and -1 otherwise. This leads to the following definitions:

Definition 5.2. *Given a cut C and a set of edges K, define $K(C)$ to be 1 if there is an edge in K that crosses C and -1 otherwise.*

Definition 5.3. *Given a cut C and a state of knowledge $J = \{K_1, \cdots, K_m\}$, define $J(C)$ to be 1 if for all i, $K_i(C) = 1$ and -1 otherwise.*

It is easy to verify that for every knowledge set K and state of knowledge J, $K(C)$ and $J(C)$ are well-defined, i.e. if $K = K'$, $K(C) = K'(C)$ and if $J = J'$, $J(C) = J'(C)$.
Note that for all C, $J_{s'}(C) = -1$ and $J_{t'}(C) = 1$.
We define basis functions as follows:

Definition 5.4. *Given a set of vertices $V \subseteq V(G)$ that does not include s or t, define $e_V(C) = (-1)^{|V \cap C|}$.*

We define the dot product as follows:

Definition 5.5. *Given two functions $f, g : \mathcal{C} \to \mathbb{R}$, $f \cdot g = 2^{2-N} \sum_{C \in \mathcal{C}} f(C)g(C)$*

Note that $e_V(C)e_{V'}(C) = (-1)^{(V \Delta V') \cap C}$ for every cut C, where Δ denotes the symmetric difference of two sets, and hence the functions $\{e_V\}$ form an orthonormal basis for the vector space $\mathbb{R}^{\mathcal{C}}$ with the standard dot product $f \cdot g = 2^{2-N} \sum_{C \in \mathcal{C}} f(C)g(C)$.
We define Fourier coefficients as follows:

Definition 5.6. $\hat{f_V} = f \cdot e_V$

Proposition 5.7. *For any function f, $f = \sum_V \hat{f_V} e_V$ and $f \cdot f = \sum_V \hat{f_V}^2$, where we are summing over all subsets V of $V(G)$ such that $s, t \notin V$.*

Proposition 5.8. *Given a monotone switching network G', if there an edge with label $v_1 \to v_2$ between vertices a' and b', then for any cut C, if $v_1 \notin C$ or $v_2 \notin C^c$, then $J_{a'}(C) = J_{b'}(C)$.*

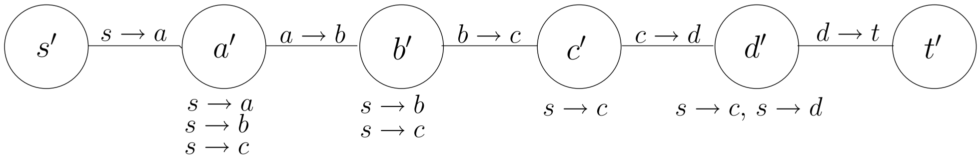

FIGURE 6. This is a possible path in G' from s' to t'. Below, we express the six functions $J_{s'}$, $J_{a'}$, $J_{b'}$, $J_{c'}$, $J_{d'}$, and $J_{t'}$ in terms of the basis functions:
$J_{s'} = -e_{\{\}}$.
$J_{a'} = -\frac{3}{4}e_{\{\}} + \frac{1}{4}e_{\{a\}} + \frac{1}{4}e_{\{b\}} + \frac{1}{4}e_{\{a,b\}} + \frac{1}{4}e_{\{c\}} + \frac{1}{4}e_{\{a,c\}} + \frac{1}{4}e_{\{b,c\}} + \frac{1}{4}e_{\{a,b,c\}}$.
$J_{b'} = -\frac{1}{2}e_{\{\}} + \frac{1}{2}e_{\{a\}} + \frac{1}{2}e_{\{b\}} + \frac{1}{2}e_{\{a,b\}}$.
$J_{c'} = e_{\{c\}}$.
$J_{d'} = \frac{1}{2}e_{\{\}} + \frac{1}{2}e_{\{a\}} + \frac{1}{2}e_{\{b\}} - \frac{1}{2}e_{\{a,b\}}$.
$J_{t'} = e_{\{\}}$.

5.2. **Warm-up: linear and quadratic lower size bounds.** We will now use the Fourier transformation technique to show an $\Omega(N^2)$ lower size bound on monotone switching networks solving directed connectivity.

To do this, we will consider linear combinations of the $J_{v'}$ functions.

Proposition 5.9. *If span$\{J_{v'}\}$ has rank at least m, then G' has at least $m+1$ vertices.*

Proof. The vector space of linear combinations of $J_{s'}$ and $J_{t'}$ has rank 1. Each new vertex can add at most 1 to the rank of the vector space, and this completes the proof. □

Definition 5.10. *Given a directed walk P' from v_1' to v_2' in G' and a label e, define $d(P', e) \in \mathbb{R}^{\mathcal{C}}$ to be $\frac{1}{2}(\sum_{v' \in V_{sink}} J_{v'} - \sum_{v' \in V_{source}} J_{v'})$, where V_{sink} is the set of vertices in G' with an edge in P' with label e going into it and V_{source} is the set of vertices in G' with an edge in P' with label e going out from it, counted with multiplicity.*

Remark 5.11. *Together, all of the edges in P' allow us to go from v_1' to v_2', and the change in our knowledge is $J_{v_2'} - J_{v_1'}$. $d(P', e)$ measures the contribution to this change made by edges with label e.*

Clearly, for any P' and e, $d(P', e)$ is in *span*$\{J_{v'}\}$.

Theorem 5.12. *If G' is a monotone switching network solving directed connectivity on N vertices, then G' has at least N vertices.*

Proof. We obtain this lower size bound by combining several simple statements.

Proposition 5.13. *If P' is a directed path in G' from s' to t' using only edges with labels $s \to a$ and $a \to t$, then for a cut C, $d(P', s \to a)(C)$ is 1 if $a \in C^c$ and 0 otherwise.*

Proof of Proposition 5.13. For a cut C, if $a \in C$ then using Proposition 5.8, $d(P', s \to a)(C) = 0$. If $a \in C^c$, then using Proposition 5.8, $d(P', a \to t)(C) = 0$. Since
$d(P', s \to a)(C) + d(P', a \to t)(C) = \frac{1}{2}(J_{t'}(C) - J_{s'}(C)) = 1$, $d(P', s \to a)(C) = 1$.
□

Proposition 5.14. *If P' is a directed path in G' from s' to t' using only edges with labels $s \to a$ and $a \to t$, then for a cut C, $d(P', a \to t)(C)$ is 1 if $a \in C$ and 0 otherwise.*

Proof of Proposition 5.14. For a cut C, if $a \in C^c$ then using Proposition 5.8, $d(P', a \to t)(C) = 0$. If $a \in C$, then using Proposition 5.8, $d(P', s \to a)(c) = 0$. Since
$d(P', s \to a)(C) + d(P', a \to t)(C) = \frac{1}{2}(J_{t'}(C) - J_{s'}(C)) = 1$, $d(P', a \to t)(C) = 1$.

□

Corollary 5.15. *Let $f = d(P', s \to a) - d(P', a \to t)$. Then $\hat{f}_{\{a\}} = 1$, and all other Fourier coefficients are zero.*

Proof of Theorem 5.12 using Corollary 5.15. For each of the $N - 2$ vertices v that are not equal to s or t, we can create a linear combination of $J_{v'}$ such that the resulting function f has all Fourier coefficients 0 except for $\hat{f}_{\{v\}}$, which is nonzero. Also, if $f = \frac{1}{2}(J_{t'} - J_{s'})$, $\hat{f}_{\{\}} = 1$ and all other fourier coefficients are zero. Thus, these $N - 1$ functions are linearly independent, and the result follows from Proposition 5.9. □

□

Theorem 5.16. *If G' is a monotone switching network solving directed connectivity on N vertices, then G' has at least $\frac{(N-2)(N-3)}{2} + N$ vertices.*

Proof. Again, we obtain this lower size bound by combining several simple statements.

Proposition 5.17. *If P' is a directed path in G' from s' to t' using only edges with labels $s \to a$, $a \to b$, and $b \to t$, then for a cut C, $d(P', a \to b)(C)$ is 1 if $a \in C$ and $b \in C^c$ and 0 otherwise.*

Proof of Proposition 5.17. For a cut C, if $a \notin C$ or $b \notin C^c$ then using Proposition 5.8, $d(P', a \to b)(c) = 0$. If $a \in C$ and $b \in C^c$, then using Proposition 5.8, $d(P', s \to a)(C) + d(P', b \to t)(C) = 0$. Since
$d(P', s \to a)(C) + d(P', a \to b)(C) + d(P', b \to t)(C) = \frac{1}{2}(J_{t'}(C) - J_{s'}(C)) = 1$, $d(P', a \to b)(C) = 1$.
□

Proposition 5.18. *If P' is a directed path in G' from s' to t' using only edges with labels labels $s \to a$, $a \to b$, and $b \to t$, then for a cut C, $d(P', s \to a) + d(P', b \to t)(C)$ is 0 if $a \in C$ and $b \in C^c$ and 1 otherwise.*

Proof of Proposition 5.18. For a cut C, if $a \in C$ and $b \in C^c$ then using Proposition 5.8, $d(P', a \to t)(C) + d(P', b \to t)(C) = 0$. If not, then using Proposition 5.8, $d(P', a \to b)(C) = 0$. Since $d(P', s \to a)(C) + d(P', a \to b)(C) + d(P', b \to t)(C) = \frac{1}{2}(J_{t'}(C) - J_{s'}(C)) = 1$, $d(P', a \to t)(C) + d(P', b \to t)(C) = 1$.

□

Corollary 5.19. *Let $f = d(P', s \to a) - d(P', a \to b) + d(P', b \to t)$. Then $\hat{f}_{\{\}} = \frac{1}{2}$, $\hat{f}_{\{a\}} = \frac{1}{2}$, $\hat{f}_{\{b\}} = -\frac{1}{2}$, $\hat{f}_{\{a,b\}} = \frac{1}{2}$, and all other Fourier coefficients are zero.*

Proof of Theorem 5.16 using Corollary 5.19. For each pair of vertices $\{v_1, v_2\}$ not equal to s or t, as shown above, we can create a function where $\hat{f}_{\{v_1,v_2\}} \neq 0$. As long as each pair of vertices is used only once, this will be the only function for which this is true. In this way, we can obtain $\frac{(N-2)(N-3)}{2}$ linearly independent functions. After this, we can still use the same $N - 1$ functions from before, so this gives us a total of $\frac{(N-2)(N-3)}{2} + N - 1$ linearly independent functions. Again, the result follows from Proposition 5.9.

□

□

5.3. **General techniques for obtaining lower size bounds.** In this subsection, we show how more general lower size bounds can be obtained.

Definition 5.20. *Given a directed walk P' from v_1' to v_2' in G' using only the edges of some directed path P in G from s to t and a partition of the edges of P into two sets, E_1 and E_2, let* $f_{P',P,E_1,E_2} = \sum_{e \in E_1} d(P',e) - \sum_{e \in E_2} d(P',e)$.

Definition 5.21. *We say a cut C is (P,E_1,E_2)-invariant if all edges e in P that cross C are in E_1 or all edges e in P that cross C are in E_2. We say a function $g : \mathcal{C} \to \mathbb{R}$ is (P,E_1,E_2)-invariant if $f_{P',P,E_1,E_2} \cdot g$ is the same for all switching networks G' solving directed connectivity on $V(G)$ and paths P' in G' from s' to t' using only the edges of P. If g is (P,E_1,E_2)-invariant, define $z(g,P,E_1,E_2)$ to be this constant.*

Proposition 5.22. *A function $g : \mathcal{C} \to \mathbb{R}$ is (P,E_1,E_2)-invariant if and only if $g(C) = 0$ for every C that is not (P,E_1,E_2)-invariant.*

Proof. For any cut C that is not (P,E_1,E_2)-invariant, we can change the value of $f_{P',P,E_1,E_2}(C)$ without changing $f_{P',P,E_1,E_2}(C')$ for any other cut C'. To see this, given a G', create a new G' by creating a new s'. Let a' be the old s' and for each edge e such that e crosses C, create an edge with label e between s' and a'. This is still a valid monotone switching network solving directed connectivity on $V(G)$ and for all vertices v' except s', $J_{v'}(C) = 1$. Also, $J_{a'}(C') = 1$ if $C' = C$ and -1 otherwise. Thus, we can change $f_{P',P,E_1,E_2}(C)$ without changing $f_{P',P,E_1,E_2}(C')$ for any other C' by choosing whether to use an edge with label in E_1 or E_2 to go from s' to a'. Thus, if $g(C) \neq 0$, then g cannot be (P,E_1,E_2)-invariant.

Let P' be a path in G' from s' to t' using only the edges of P. If C cannot be crossed by any edge in E_1, then $\sum_{e \in E_1} d(P',e)(C) = 0$. Again, $\sum_{e \in E_1} d(P',e)(C) + \sum_{e \in E_2} d(P',e)(C) = \frac{1}{2}(J_{t'}(C) - J_{s'}(C)) = 1$, so $\sum_{e \in E_2} d(P',e)(C) = 1$, and $f_{P',P,E_1,E_2}(C) = -1$. Similarly, if C cannot be crossed by any edge in E_2, then $f_{P',P,E_1,E_2}(C) = 1$. Thus, if $g(C) = 0$ for every C that is not (P,E_1,E_2)-invariant, then $f_{P',P,E_1,E_2} \cdot g$ is the same for all G' and all paths P' in G' from s' to t' using only the edges of P, as needed. □

For paths P of length 2 and 3, we were able to choose E_1 and E_2 so that for all cuts C, $f_{P',P,E_1,E_2}(C)$ is the same for all G' and all paths P' in G' from s' to t' using only the edges of P. Unfortunately, for longer paths, this is no longer possible. This makes proving linear independence much harder.

To end this section, we show that a lower size bound can be obtained from these techniques even without using linear independence.

Theorem 5.23. *If when $V(G) = \{s, v_1, \cdots, v_{k_1}, t\}$ and P is the path $s \to v_1 \to \cdots \to v_{k_1} \to t$, we have a partition of the edges of P into two groups E_1 and E_2 and a function g_P such that g_P is (P,E_1,E_2)-invariant, $z(g_P,P,E_1,E_2)$ is nonzero, and $\hat{g_P}_V = 0$ for any set of vertices V such that $|V| < k_2$, then any monotone switching network solving directed connectivity on N vertices has size at least $\Omega(N^{\frac{k_2}{2}})$.*

Proof. Without loss of generality, we may assume $g_P \cdot g_P = 1$. Given such a g_P, note that if we add vertices to $V(G)$ so that $|V(G)| = N$, we can keep the same g_P (expressed in fourier coefficients) and it will still be (P,E_1,E_2)-invariant and have the same $z(g_P,P,E_1,E_2)$. Let $M = z(g_P,P,E_1,E_2)$.

If we have another path P_2 from s to t of length $k_1 + 1$, by symmetry, we have a function g_{P_2} and a partion (E_3,E_4) of the edges of P_2 so that $g_{P_2} \cdot g_{P_2} = 1$, g_{P_2} is (P_2,E_3,E_4)-invariant, and $z(g_{P_2},P_2,E_3,E_4) = M$. Moreover, if P and P_2 have less than k_2 vertices in common (excluding s and t), then $g_P \cdot g_{P_2} = 0$. If we have K paths $P_1, \cdots, P_K$ of length $k_1 + 1$ from s to t in G such that any pair of them have less than k_2 vertices in common (excluding s and t), then we have K orthonormal functions $\{g_{P_i}\}$.

Now assume G' solves directed connectivity on N vertices, and let N' be the number of vertices in G'. We wish to bound N' from below. Note that given any set of orthonormal functions $\{g_i\}$,

$$N' \geq \sum_i (\sum_{a' \in V(G')} |J_{a'} \cdot g_i|^2) \tag{1}$$

Using Cauchy-Schwarz (specifically $\sum_{j=1}^{N'} |c_j|^2 \geq \frac{1}{N'}(\sum_{j=1}^{N'} |c_j|)^2$ for any $c_1, \cdots, c_{N'}$),

$$N' \geq \sum_i (\sum_{a' \in V(G')} |J_{a'} \cdot g_i|^2) \tag{2}$$

$$N' \geq \frac{1}{N'} \sum_i (\sum_{a' \in V(G')} |J_{a'} \cdot g_i|)^2 \tag{3}$$

$$N' \geq \sqrt{\sum_i (\sum_{a' \in V(G')} |J_{a'} \cdot g_i|)^2} \tag{4}$$

If P' is a path in G' from s' to t' using only the edges of P, by the definition of f_{P',P,E_1,E_2},

$$M = |f_{P',P,E_1,E_2} \cdot g_P| \leq \sum_{a' \in V(P')} |J_{a'} \cdot g_P| \tag{5}$$

Also, we clearly have that

$$\sum_{a' \in V(P')} |J_{a'} \cdot g_P| \leq \sum_{a' \in V(G')} |J_{a'} \cdot g_P| \tag{6}$$

Combining 5 and 6, we get that

$$\sum_{a' \in V(G')} |J_{a'} \cdot g_P| \geq M \tag{7}$$

$$(\sum_{a' \in V(G')} |J_{a'} \cdot g_P|)^2 \geq M^2 \tag{8}$$

By symmetry, 8 holds for each g_{P_i}. Plugging 8 into 4,

$$N' \geq \sqrt{KM^2} = M\sqrt{K} \tag{9}$$

Following similar logic as in the proof of Theorem 1.2, we can easily obtain make K at least $\Omega(N^{k_2})$, so N' is at least $\Omega(N^{\frac{k_2}{2}})$, as needed. □

Corollary 5.24. *If for all k, when $V(G) = \{s, v_1, \cdots, v_{2^k}, t\}$ and P is the path $s \to v_1 \cdots \to v_{2^k} \to t$, there exists a partition of the edges of P into two groups E_1 and E_2 and a function g_P such that g_P is (P, E_1, E_2)-invariant, $z(g_P, P, E_1, E_2)$ is nonzero, and $\hat{g_{P}}_V = 0$ for any set of vertices V such that $|V| \leq k$, then any monotone switching network solving directed connectivity must have superpolynomial size.*

Proof. Applying Theorem 5.23 for a fixed k gives that any monotone switching network solving directed connectivity must have size at least $c(k)(N^{\frac{k+1}{2}})$, where $c(k)$ is a constant depending on k. Thus, the size of a monotone switching network solving directed connectivity grows faster than any polynomial, as needed. □

Remark 5.25. *Without a bound on $c(k)$, we cannot give an explicit lower bound on the size of a switching network solving directed connectivity, we can only say that it is superpolynomial. These bounds will be given in Section 7.*

6. A Superpolynomial Bound

In this section, we use Fourier analogues of results in Sections 3 and 4 to prove the following theorem:

Theorem 6.1. *For all k, if $V(G) = \{s, v_1, \cdots, v_{2^k}, t\}$ and P is the path $s \to v_1 \cdots \to v_{2^k} \to t$, there exists a partition of the edges of P into two groups E_1 and E_2 and a function g_P such that g_P is (P, E_1, E_2)-invariant, $z(g_P, P, E_1, E_2)$ is nonzero, and $\hat{g_P}_V = 0$ for any set of vertices V such that $|V| \leq k$.*

By Corollary 5.24, this is sufficient to prove a superpolynomial lower size bound on monotone switching networks solving directed connectivity.

6.1. **Proof Overview.** We now give an informal overview of the proof of Theorem 6.1.

It is instructive to first note how this function g_P relates to certain-knowledge switching networks and Lemma 3.8. If we let W be the set of all a' such that the union of the endpoints of $K_{a'}$ contains at least $k + 1$ vertices, then Lemma 3.8 says that any path P' in G' from s to t using only the edges of P must pass through at least one vertex $w' \in W$. We can think of W as a barrier preventing us from easily going from s' to t'. The function g_P describes this barrier more precisely, as if we let W' be the set of all vertices a' such that $K_{a'} \cdot g_{P,E_1,E_2} \neq 0$ and the union of the endpoints of $K_{a'}$ contains no vertices not in P, then P' must pass through at least one vertex $w' \in W'$. Also, $W' \subseteq W$.
Thus, the existence of such a g_P implies Lemma 3.8. Roughly speaking, we want to show the converse, that the existence of such a barrier for certain-knowledge switching networks implies the existence of such a g_P.
To show that a function $g : \mathcal{C} \to \mathbb{R}$ is (P, E_1, E_2)-invariant, we either need to show that $g(C) = 0$ for all cuts C that are not (P, E_1, E_2)-invariant, or we need to show that $f_{P',P,E_1,E_2} \cdot g$ is the same for all G' and all paths P' from s' to t'. If we had an explicit formula for $g(C)$, it would be easiest to use the first approach. However, since we do not have such a general formula, we use the second approach.
Lemma 6.4, the Fourier analogue of Theorem 4.5, shows that it is sufficient to consider only G' where each $J_{a'}$ contains only knowledge sets such that all edges in the knowledge set are of the form $s \to v$. Theorem 6.5, the Fourier analogue of Theorem 4.6, shows that if we add the condition that $f_{L',P,E_1,E_2} \cdot g_P = 0$ for all directed cycles L' of G' using only the edges of P, then it is sufficient to consider only certain-knowledge switching networks. Combining these results, we have Theorem 6.3, which says that with the added condition, we only need to consider certain-knowledge switching networks such that all knowledge sets contain only edges of the form $s \to v$.
Since there are now at most $2^{2^k} + 1$ knowledge sets and we must have $J_{t'} = -J_{s'}$, as noted in Lemma 6.12, we can arbitrarily choose the values $g \cdot J_{a'}$ for all $a' \in V(G')$ except t'. Lemma 6.13 shows that if we can split the vertices of G' into 4 groups with a mapping $b : V(G') \to \{0,1\} \times \{0,1\}$ that has ceratin properties, then using this freedom, we can create a g that satisfies the conditions given by Theorem 6.3 and is thus (P, E_1, E_2)-invariant. Moreover, $z(g_P, P, E_1, E_2)$ is nonzero and $\hat{g_P}_V = 0$ for any set of vertices V such that $|V| \leq k$. Lemma 6.15 shows that if we have a barrier W similar to the one provided by Lemma 3.8 with one additional property, then we can create a mapping $b : V(G') \to \{0,1\} \times \{0,1\}$ as required by Lemma 6.13. Finally, Lemma 6.17 modifies Lemma 3.8 so that it provides the barrier W with the needed additional property. Putting everything together, we can create a function g_P satisfying Theorem 6.1.

6.2. **Reduction to certain-knowledge Switching Networks.** In this subsection, we prove Theorem 6.3, showing that to prove a function $g : \mathcal{C} \to \mathbb{R}$ is (P, E_1, E_2)-invariant, it is sufficient to look

at the behavior of g on certain-knowledge switching networks G' where all knowledge sets contain only edges of the form $s \to v$.

Remark 6.2. *Throughout this subsection, we will always assume that we have a directed path P in G from s to t and a partition (E_1, E_2) of the edges of P, and we will not consider any directed paths or cycles in G' that use an edge not in P.*

Theorem 6.3. *If for a function $g : \mathcal{C} \to \mathbb{R}$, for any certain-knowledge G' such that all knowledge sets contain only edges of the form $s \to v$, $f_{P',P,E_1,E_2} \cdot g$ is the same for all paths P' in G' from s' to t' and $f_{L',P,E_1,E_2} \cdot g = 0$ for all directed cycles L' in G', then g is (P, E_1, E_2)-invariant.*

Proof. We begin with the following analogue of Theorem 4.5:

Lemma 6.4. *If for a function $g : \mathcal{C} \to \mathbb{R}$, for all G' such that each $J_{a'}$ contains only knowledge sets such that each edge in the knowledge set is of the form $s \to v$, $f_{P',P,E_1,E_2} \cdot g$ is the same for all paths P' in G' from s' to t', then g is (P, E_1, E_2)-invariant.*

Proof of Lemma 6.4. Assume g is not (P, E_1, E_2)-invariant. Then there is a cut C such that C is not (P, E_1, E_2)-invariant and $g(C) \neq 0$.
Let $v_1, \cdots, v_m$ be the vertices in C and let $w_1, \cdots, w_k$ be the vertices in C^c. Let $K = \{s \to v_1, s \to v_2, \cdots, s \to v_m\}$. For each possible state of knowledge J that contains only knowledge sets such that each edge in the knowledge set is of the form $s \to v$, create a vertex v' with state of knowledge $J_{v'} = J$. Add all labeled edges allowed by property 2 of Proposition 4.4.
Let a' be the vertex with state of knowledge $J_{a'} = \{K\}$. Let b' be the vertex with state of knowledge $J_{b'} = \{K \cup s \to w_1, K \cup s \to w_2, \cdots, K \cup s \to w_k\}$.
Now note that $J_{b'}(C) = 1$, $J_{a'}(C) = 0$ and if $C' \neq C$, then $J_{b'}(C') = J_{a'}(C')$. We can easily find a path P' in G' from s' to t' using only edges in P that has an edge e' from a' to b'. We can choose whether e' has a label in E_1 or E_2, and this will change the value of $f_{P',P,E_1,E_2} \cdot g$. This completes the proof. □

We now give the following analogue of Theorem 4.6:

Theorem 6.5. *If for a function $g : \mathcal{C} \to \mathbb{R}$, for any certain-knowledge G', $f_{P',P,E_1,E_2} \cdot g$ is the same for all paths P' from s' to t' and $f_{L',P,E_1,E_2} \cdot g = 0$ for all directed cycles L' in G', then g is (P, E_1, E_2)-invariant.*

Proof of Theorem 6.5.

Proposition 6.6. *If $J = \{K_1, K_2, \cdots, K_m\}$ where $m \neq 0$, then $J(C) - J_{s'}(C) = \sum_I (-1)^{|I|+1}((\cup_{i \in I} K_i)(C) - J_{s'}(C))$ where I ranges over all of the possible subsets of $\{1, 2, \cdots m\}$.*

Proof of Proposition 6.6. $J(C) - J_{s'}(C) = 2$ if $K_i(C) = 1$ for every i and 0 otherwise.

If $K_i(C) = -1$ for some i, then we can add or remove i from I without affecting $(\cup_{i \in I} K_i)(C) - J_{s'}(C)$. But then the sum on the right is automatically 0.
If $K_i(C) = 1$ for all i, then unless I is empty, $(\cup_{i \in I} K_i)(C) - J_{s'}(C) = 2$. From this, it is easy to see that the right hand side is 2, as needed. This completes the proof. □

Lemma 6.7. *$J_{b'} - J_{a'} = \sum_{moves} K_{end} - K_{start}$, where both K_{start} and K_{end} are unions of subsets of $\{K_{a'i}\}$ or unions of subsets of $\{K_{b'j}\}$ and the moves are as described in Theorem 4.6. We give each move a direction by requiring that K_{start} is either the union of an odd number of $K_{a'i}$ or the union of an even number of $K_{b'j}$ and K_{end} is either the union of an even number of $K_{a'i}$ or the union of an odd number of $K_{b'j}$.*

Proof of Lemma 6.7. Recall that the moves in Theorem 4.6 are as follows: If we are at a vertex a' with a subset of the $\{K_{a'i}\}$ and we want to move to the vertex b', do the following:

If every $K_{a'i}$ we have is the representative of a cycle, then travel to b' and replace each $K_{a'i}$ with the corresponding $K_{b'j}$. If not, then do the following:
1. For each $K_{a'i}$ we have that is the representative of a cycle, replace it by the corresponding $K_{b'j}$.
2. Take the earliest $K_{a'i}$ that is not the representative of a cylce. Take the $K_{b'j}$ that the arrow going from this $K_{a'i}$ is pointing to. Remove this $K_{b'j}$ if it is in our set and add it if it is not.
3. For each $K_{b'j}$ we have, replace it by the corresponding $K_{a'i}$.

Note that every move either changes where we are or changes the number of knowledge sets by 1. Thus, if we look at the pairs of K that are connected by a move, then one of them will be in K_{start} and the other will be in K_{end}. Thus, we can give each move a direction as described. Also, note that for each possible K_{start}, there is exactly one move from it and for each possible K_{end}, there is exactly one possible move to it. Thus, each possible K_{start} or K_{end} is counted exactly once. The result now follows immediately from Proposition 6.6. □

Proof of Theorem 6.5 from Lemma 6.7. Now for each vertex a' in P' not equal to s' or t', for each possible nonempty subset of the $\{K_{a'i}\}$, create a vertex. This corresponds to being at a' and having that subset. Create a vertex s'' corresponding to being at s' and having $K = \{\}$ and create one vertex t'' corresponding to being at t' and having $K = \{s \to t\}$. For each move, create an edge between the corresponding vertices. Call the resulting graph H'.
After we are done, every vertex excluding s'' and t'' has degree 2. Thus, this graph consists of a path between s'' and t'' and cycles. Note that for each move, we are starting at one vertex in P' and attempting to move to an adjacent vertex in P'. Thus, we can give each move a direction according to Lemma 6.7. For a given vertex a' in P' not equal to s' or t' and subset of the knowledge sets in $J_{a'}$, one move from it attempts to go to the next vertex in P' and the other move attempts to go to the previous vertex in P'. Thus, after we make the edges directed, each vertex in H' except s'' and t'' has indegree 1 and outdegree 1. H' consists of a directed path $P'_{H'}$ from s'' to t'' and directed cycles.

Definition 6.8. *Given an edge e' in G' and a direction for this edge, define $d_{G'}(e')$ to be $J_{v'_{sink}} - J_{v'_{source}}$, where v'_{sink} is the vertex in G' that e' goes to and v'_{source} is the vertex in G' that e' comes from.*

Corollary 6.9. *For any edge e' in P',*
$d_{G'}(e') = \sum_{e' \in E_{e'}} d_{H'}(e')$, where $E_{e'}$ is the set of all edges in H' that correspond to e'.

Proof. This follows immediately from Lemma 6.7 and the definition of H'. □

Corollary 6.10. $d(P', e) = d(P'_{H'}, e) + \sum_{L' \in H'} d(L', e)$

Proof. This follows immediately from Corollary 6.9 and the definitions. □

Theorem 6.5 follows directly from Corollary 6.10, and this completes the proof. □

□

Proof of Theorem 6.3 from Lemma 6.4 and Theorem 6.5. To prove Theorem 6.3, first use Lemma 6.4 and then use the exact same argument as in Theorem 6.5. Since we now start with a G' such that all edges in the knowledge sets have the form $s \to v$, when we create H', all of the knowledge sets in H' will only contain paths of the form $s \to v$, and this completes the proof. □

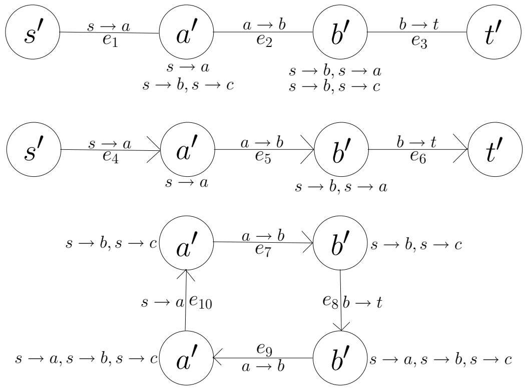

FIGURE 7. This figure illustrates the ideas used in the proof of Theorem 6.5. P' is shown above, and H' is shown below. The labels inside the vertices of H' show which vertex in P' we are on at that point, and the labels next to the vertices of H' show which K we have at that point. Take
$K_1 = \{s \to a\}$, $K_2 = \{s \to a, s \to b\}$, $K_3 = \{s \to b, s \to c\}$, and
$K_4 = K_1 \cup K_3 = K_2 \cup K_3 = \{s \to a, s \to b, s \to c\}$.
e_4 and e_{10} correspond to e_1, and
$d_{G'}(e_1) = J_{a'} - J_{s'} = (K_1 - K_{s'}) + (K_3 - K_4) = d_{H'}(e_4) + d_{H'}(e_{10})$.
e_5, e_7, and e_9 correspond to e_2, and
$d_{G'}(e_2) = J_{b'} - J_{a'} = (K_2 + K_3 - K_4) - (K_1 + K_3 - K_4) = (K_2 - K_1) + (K_3 - K_3) + (K_4 - K_4) = d_{H'}(e_5) + d_{H'}(e_7) + d_{H'}(e_9)$.
e_6 and e_8 correspond to e_3, and
$d_{G'}(e_3) = J_{t'} - J_{b'} = (K_{t'} - K_2) + (K_4 - K_3) = d_{H'}(e_6) + d_{H'}(e_8)$.

□

6.3. **Construction of g_P.** In this subsection, we complete the proof of Theorem 6.1 by constructing a function $g_P : \mathcal{C} \to \mathbb{R}$ with the given properties.

Looking at certain-knowledge G' where each knowledge set only has paths of the form $s \to v$, there are only $2^{2^k} + 1$ possible knowledge sets: $s \to t$ and anything of the form $\cup_{v \in V}\{s \to v\}$ for some set of vertices V. Denote each such K by K_V.

Proposition 6.11. *For all subsets V, V' of $V(G)$ not containing s or t, $e_V \cdot K_V \neq 0$ and if $V' \not\subseteq V$, then $e_{V'} \cdot K_V = 0$.*

Lemma 6.12. *For any set of values $\{a_V\}$, there is a function $g : \mathcal{C} \to \mathbb{R}$ such that for all subsets V of $V(G)$ not containing s or t, $g \cdot K_V = a_V$. Furthermore, if there is a k such that if $|V| \leq k$, then $g \cdot K_V = 0$, then writing $g = \sum_{V'} c_{V'} e_{V'}$, if $|V'| \leq k$ then $c_{V'} = 0$.*

Proof of Lemma 6.12. To see the first part of the lemma, pick an ordering of the V such that no V is a subset of an earlier V. Now pick each c_V in that order. Since if $V' \not\subseteq V$, then $e_{V'} \cdot K_V = 0$ and $e_V \cdot K_V \neq 0$, this means that when we pick each c_V, we can change the value of a_V without affecting any earlier a_V. Thus, we can freely choose each a_V.
To see the second part of the lemma, let V be a set such that $c_V \neq 0$ and for all proper subsets V' of V, $c_{V'} = 0$. Then by the above proposition, $a_V \neq 0$, as needed. This completes the proof. □

Lemma 6.13. *If we have a directed path P in G, a partition of the edges of P into two sets E_1 and E_2, and a mapping $b : V(G') \to \{0,1\} \times \{0,1\}$ such that if we write $b(v') = (b_1(v'), b_2(v'))$, then:*
1. $b_1(s') = b_2(s') = 0$.
2. $b_1(t') = b_2(t') = 1$.
3. If $b_i(v'_1) \neq b_i(v'_2)$, $i \in \{1,2\}$, then there is no edge with label in E_i between v'_1 and v'_2.
Then if we also have a $g : \mathcal{C} \to \mathbb{R}$ such that $g \cdot J_{v'} = b_2(v') - b_1(v')$ for all $v' \in V(G')$, then for any directed cycle L' in G' using only the edges of P, $f_{L',P,E_1,E_2} \cdot g = 0$ and for any path P' in G' from s' to t' using only the edges of P, $f_{P',P,E_1,E_2} \cdot g = 1$.

Proof of Lemma 6.13. This follows immediately from the following proposition:

Proposition 6.14. *With the above conditions, if P'' is a path in G' from s' to a', then*
$\frac{1}{2}(\sum_{e \in E_1} d(P'',e) - \sum_{e \in E_2} d(P'',e)) \cdot g = \frac{1}{2}(b_2(a') + b_1(a'))$

Proof of Proposition 6.14. We prove this by induction. It is clearly true for paths of length 0. Assume we have a path P'' from s' to some vertex $v'_1 \in V(G')$ for which the proposition is true and an additional edge e' from v'_1 to some vertex $v'_2 \in V(G')$. Let P''' be P'' with the edge e' added.
If e' has a label in E_1, then $b_1(v'_2) = b_1(v'_1)$, so
$\frac{1}{2}(\sum_{e \in E_1} d(P''',e) - \sum_{e \in E_2} d(P''',e)) \cdot g = \frac{1}{2}(\sum_{e \in E_1} d(P'',e) - \sum_{e \in E_2} d(P'',e)) \cdot g + \frac{1}{2}(g \cdot J_{v'_2}) - \frac{1}{2}(g \cdot J_{v'_1})$
$= \frac{1}{2}(b_2(v'_1) + b_1(v'_1)) + \frac{1}{2}(b_2(v'_2) - b_1(v'_2)) - \frac{1}{2}(b_2(v'_1) - b_1(v'_1))$
$= \frac{1}{2}(b_2(v'_2) + b_1(v'_2))$, as needed.
Similarly, if e' has a label in E_2, then $b_2(v'_2) = b_2(v'_1)$, so
$\frac{1}{2}(\sum_{e \in E_1} d(P''',e) - \sum_{e \in E_2} d(P''',e)) \cdot g = \frac{1}{2}(\sum_{e \in E_1} d(P'',e) - \sum_{e \in E_2} d(P'',e)) \cdot g - \frac{1}{2}(g \cdot J_{v'_2}) + \frac{1}{2}(g \cdot J_{v'_1})$
$= \frac{1}{2}(b_2(v'_1) + b_1(v'_1)) + \frac{1}{2}(-b_2(v'_2) + b_1(v'_2)) - \frac{1}{2}(-b_2(v'_1) + b_1(v'_1))$
$= \frac{1}{2}(b_2(v'_2) + b_1(v'_2))$, as needed.

This completes the proof. □

□

Lemma 6.15. *If there is a set of vertices W in G' such that any path P' from s' to t' using only edges with labels in P contains a vertex $w' \in W$ incident with both an edge in P' with label in E_1 and an edge in P' with label in E_2, then it is possible to find a mapping $b : V(G') \to \{0,1\} \times \{0,1\}$ as described in Lemma 6.13 so that all vertices v' such that $b_1(v') \neq b_2(v')$ are in W.*

Proof of Lemma 6.15. Delete all edges in G' whose labels are not in P. Treat all edges in E_1 as equivalent and treat all edges in E_2 as equivalent.
Let W' be a subset of W for which the same condition holds and if we remove any vertex from W', this condition no longer holds.
If for some a', b', c', and $i \in \{1,2\}$ there is an edge between vertex a' and b' with label in E_i and an edge between vertex b' and c' with label in E_i, then add an edge with label in E_i between a' and c'. Keep on doing this until doing so does not add any new edges.

Remark 6.16. *Such a step cannot affect the given condition. To see this, assume this creates a new path P' violating the condition. P' must contain this new edge. But then we can replace this new edge by the two old edges to obtain a path we already had that still violates the condition*

If a' and b' are two adjacent vertices in $V(G')$ that are not in W', then we require that $b(a') = b(b')$. This partitions the vertices of G' that are not in W' into connected components. Since any path from s' to t' contains a vertex in W', s' and t' are in different componenets. Set $b(s') = (0,0)$ and $b(t') = (1,1)$. Call the component with s' the starting component and call the component with t' the ending component. For all vertices v' in the starting component, $b(v') = (0,0)$. For all vertices

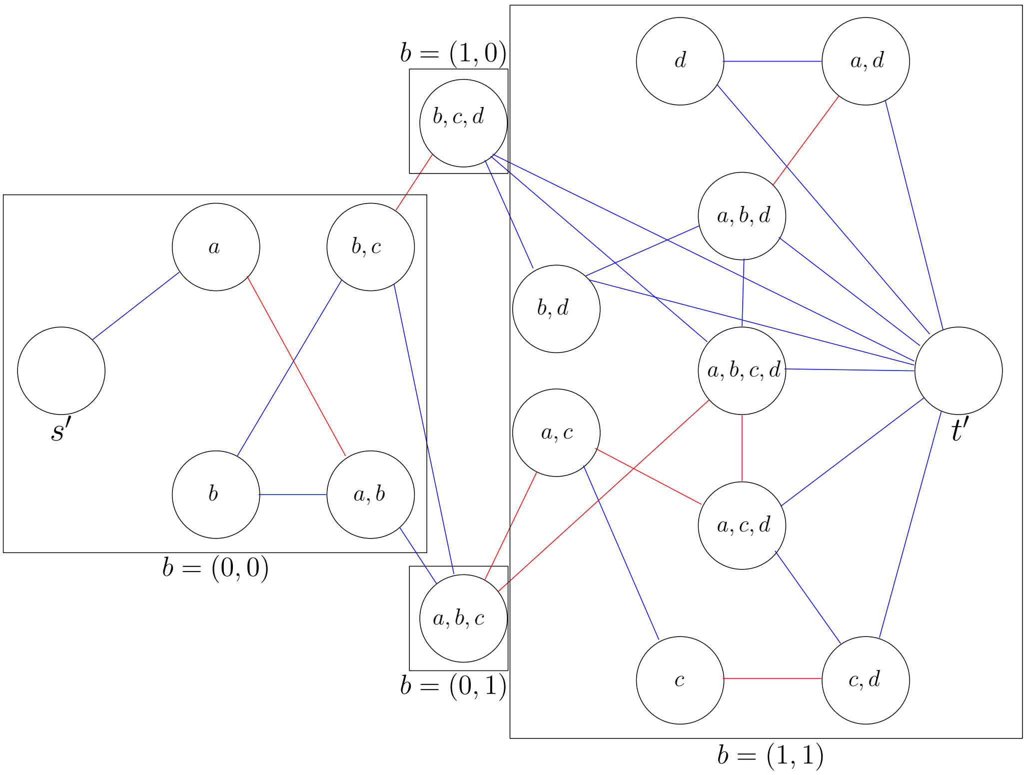

FIGURE 8. This is a partition of the vertices in G' into 4 groups with a mapping $b : V(G') \to \{0,1\} \times \{0,1\}$ as described in Lemma 6.13, where G' is a certain-knowledge switching network such that all knowledge sets contain only edges of the form $s \to v$, $P = s \to a \to b \to c \to d \to t$, $E_1 = \{s \to a, b \to c, d \to t\}$, and $E_2 = \{a \to b, c \to d\}$. In this diagram, each vertex has knowledge set K_V, where V is the set of vertices inside of the vertex. Edges with label in E_1 are blue and edges with label in E_2 are red. Taking $g = 4e_{\{a,b,c\}} - 4e_{\{b,c,d\}}$, $g \cdot J_{a'} = b_2(a') - b_1(a')$.

v' in the ending component, $b(v') = (1,1)$.

For each vertex $w' \in W'$, there is a path $P'_{W'}$ in G' from s' to t' containing w' where w' is incident with both an edge in $P'_{w'}$ with label in E_1 and an edge in $P'_{w'}$ with label in E_2 and this is true for no other vertex in W'. Otherwise, we could have removed w' from W' and the condition would still hold. Now note that if $P'_{w'}$ contains any other vertices in W', they can be bypassed using the added edges. Thus, we can obtain a $P'_{w'}$ containing w' and no other vertices in W'. Thus, each $w' \in W'$ is adjacent to at least one vertex in the starting component and one vertex in the ending component.

Given a vertex v'_1 in the starting component that is adjacent to w' and a vertex v'_2 in the ending component that is adjacent to w', we can create a path $P'_{w'}$ by taking the path from s' to v'_1, taking the edge e'_1 from v'_1 to w', taking the edge e'_2 from w' to v'_2, and taking the path from v'_2 to t'. e'_1 and e'_2 must have different labels, or else we could bypass w' entirely.

Note that the label of e'_1 cannot depend on the choice of v'_1, or else we could choose it to have the same label as e_2'. Similarly, the label of e_2' cannot depend on the choice of v_2'. If e_1' has label in E_1 and e_2' has label in E_2, then set $b(w') = (0,1)$. If e_1' has label in E_2 and e_2' has label in E_1, then set $b(w') = (1,0)$. We have now chosen $b(w')$ for all $w' \in W'$.

If two vertices w_1' and w_2' in W' are adjacent, then with the added edges, there must be a vertex v' of G' that is in the starting or ending component and is adjacent to both w_1' and w_2'. From the above, we must have that $b(w_1') = b(w_2')$.

It is now easy to verify that at this point, all conditions of Lemma 6.13 are satisfied:
1. If v' and w' are adjacent, $b(v') = (0,0)$, and $b(w') = (0,1)$, then because of the way $b(w')$ was chosen, the edge between them must have label in E_1.
2. If v' and w' are adjacent, $b(v') = (0,0)$, and $b(w') = (1,0)$, then because of the way $b(w')$ was chosen, the edge between them must have label in E_2.
3. If v' and w' are adjacent, $b(v') = (1,1)$, and $b(w') = (0,1)$, then because of the way $b(w')$ was chosen, the edge between them must have label in E_2.
4. If v' and w' are adjacent, $b(v') = (1,1)$, and $b(w') = (1,0)$, then because of the way $b(w')$ was chosen, the edge between them must have label in E_1.
5. No vertex in the starting component is adjacent to a vertex in the ending component.
6. If $w_1', w_2' \in W'$ are adjacent, then $b(w_1') = b(w_2')$.

If there is a vertex v' such that $b(v')$ has not yet been determined, then v' cannot be adjacent to any vertices in the starting component or the ending component. Also, v' cannot be adjacent to any vertices in W', as otherwise with the added edges v' would be adjacent to a vertex in the starting component or a vertex in the ending component. We can set $b(v') = (0,0)$ for all such v', and all of the conditions of Lemma 6.13 will still be satisifed. This completes the proof. □

The final Lemma we need is a slight modification of Lemma 3.8:

Lemma 6.17. *If P is the path $s \to v_1, v_1 \to v_2, \cdots, v_{2^k} \to t$, then setting $s = v_0$, $t = v_{2^k+1}$, taking E_1 to be all edges of the form $v_i \to v_{i+1}$ where i is even and taking E_2 to be the remaining edges, then if G' is a certain-knowledge switching network, any path P' in G' from s' to t' using only the edges in P must pass through at least one vertex a' such that $K_{a'} \neq K_{t'}$ and the union of the endpoints of the edges in $K_{a'}$ contains at least $k+1$ of $v_1, v_2, \cdots, v_{2^k}$ and contains no other vertices except s and t. Furthermore, a' is incident with both an edge in P' with label in E_1 and an edge in P' with label in E_2.*

Proof of Lemma 6.17. The proof is identical to the proof of Lemma 3.8, except that in the inductive hypothesis we also require that a' is incident with both an edge in P' with label in E_1 and an edge in P' with label in E_2. □

Proof of Theorem 6.1. We put everything together as follows. Let $V(G) = \{s, v_1, \cdots, v_{2^k}, t\}$ and let P be the path $s \to v_1, v_1 \to v_2, \cdots, v_{2^k} \to t$. Using Lemma 6.17, we obtain a W which we can use in Lemma 6.15. This gives us a mapping $b : V(G') \to \{0,1\} \times \{0,1\}$ which we can use in Lemma 6.13. By Lemma 6.12, we can obtain a function $g_P : \mathcal{C} \to \mathbb{R}$ that satisfies all of the conditions of Lemma 6.13, so for any directed cycle L' in G' using only the edges of P, $f_{L',P,E_1,E_2} \cdot g_P = 0$ and for any path P' in G' from s' to t' using only the edges of P, $f_{P',P,E_1,E_2} \cdot g_P = 1$. Also, by Lemma 6.12, if $|V| \leq k$, $\hat{g_P}_V = 0$. Using Theorem 6.3, g_P is (P, E_1, E_2)-invariant and $z(g_P, P, E_1, E_2) = 1$, as needed.

□

7. Proof of the Main Result

We will now modify the above ideas slightly to prove Theorem 1.3.

Throughout this section, we will take partitions (W_1, W_2) of the vertices of G, where $s \in W_1$ and $t \in W_2$. Also, in this section, unless we state that G' solves directed connectivity on $V(G)$, we do not require that there is a path from s' to t' in G' if and only if there is a path from s to t in G. Instead, we only require that if there is a path from s' to t' in G', then there must be a path from s to t in G. It is easily verified that this is true if and only if we can assign states of knowledge to the vertices of G' so that G' satisfies the properties of Proposition 4.4

Theorem 7.1. *Given a switching network G' solving directed connectivity on $V(G)$, we can create a switching network G'' such that:*
1. $|V(G'')| \leq N|V(G')|$.
2. All of the edges except $s \to t$ in the knowledge sets of G'' have the form $s \to v$ for some $v \in W_1$ or $v \to t$ for some $v \in W_2$.
3. If P is a path in G from s to t that does not have any edges of the form $a \to b$ where $a \in W_2$ and $b \in W_1$, then there is a path from s'' to t'' in G'' using only the edges of P.

Proof. The proof is similar to the proof of Theorem 4.5. First, for each edge e' with label of the form $a \to b$ where $a \in W_2$ and $b \in W_1$ in G', replace it with two edges, one with label $s \to b$ and the other with label $a \to t$. After these replacements, we still have that if there is a path from s' to t' in G', then there must be a path from s to t in G.

Again, construct G'' by taking N copies of G' and making the s' for each copy equal to the t' of the previous copy. Take s'' to be the s' of the first copy and take t' to be the t' of the last copy. Now for each path in G'', we keep track of a state of knowledge J as follows:

1. If we use an edge of the form $a \to b$ where $a, b \in W_1$, then for each knowledge set K in J that includes the edge $s \to a$, add the edge $s \to b$.
2. If we use an edge of the form $a \to b$ where $a \in W_1$ and $b \in W_2$, then for each knowledge set K in J that includes the edges $s \to a$ and $b \to t$, add the edge $s \to t$.
3. If we use an edge of the form $a \to b$ where $a, b \in W_2$, then for each knowledge set K in J that includes the edge $b \to t$, add the edge $a \to t$.

Take the J for each vertex to be the union of all J that could be obtained in this way. It is easily verified that property 2 of Proposition 4.4 is satisfied.

For each time a path goes through a copy of G', each K in J must obtain the edge $s \to t$ or at least one new edge of the form $s \to v$ for some $v \in W_1$ or $v \to t$ for some $v \in W_2$. Thus, $J_{t''} = \{\{s \to t\}\}$, as needed. This completes the proof. □

Lemma 7.2. *If for a function $g : \mathcal{C} \to \mathbb{R}$, a directed path P in G from s to t that does not use any edges of the form $v \to w$ where $v \in W_2$ and $w \in W_1$, and a partition (E_1, E_2) of the edges of P, $f_{P',P,E_1,E_2} \cdot g$ is the same for all G' such that for all of its states of knowledge, each of the knowledge sets contains only edges of the form $s \to v$ for some $v \in W_1$ or $v \to t$ for some $v \in W_2$ and all paths P' from s' to t' in G', then g is (P, E_1, E_2)-invariant.*

Proof. This can be proved in the same way as Lemma 6.4. □

Theorem 7.3. *If for a function $g : \mathcal{C} \to \mathbb{R}$, a directed path P in G that does not use any edges of the form $v \to w$ where $v \in W_2$ and $w \in W_1$, and a partition (E_1, E_2) of the edges of P, for any certain-knowledge G' such that all of the edges in the knowledge sets have the form $s \to v$ for some $v \in W_1$ or $v \to T$ for some $v \in W_2$, $f_{P',P,E_1,E_2} \cdot g$ is the same for all P' and $f_{L',P,E_1,E_2} \cdot g = 0$for all directed cycles L' in G' using only the edges of P, then for any G', $f \cdot g$ is the same for all P'.*

Proof. First, use Lemma 7.2. Then apply the reasoning used in the proof of Theorem 6.5. This completes the proof. ☐

Definition 7.4. *For a set of vertices I that does not contain s or t, define K_I to be the knowledge set $\{s \to v_1, \cdots, s \to v_k, w_1 \to t, \cdots, w_l \to t\}$, where $v_1, \cdots v_k$ are the vertices in $I \cap W_1$ and $w_1, \cdots w_l$ are the vertices in $I \cap W_2$.*

Definition 7.5. *If I is nonempty, define $g_I(C)$ to be:*
0 *if there exists a vertex v such that $v \notin I$ and $v \in W_1 \cap C$ or $v \in W_2 \cap C^c$*
$2^{N-3}(-1)^{1+|I \cap W_1 \cap C^c|+|I \cap W_2 \cap C|}$ *otherwise*
Define $g_{\{\}}(C)$ to be:
2^{N-3} *if C is W_1 or W_2*
0 *otherwise*

Lemma 7.6. *g_I is the unique function such that $g_I \cdot K_{I'} = 1$ if $I = I'$ and 0 otherwise.*

Proof. If I is nonempty,
$g_I \cdot K_{I'} = (g_I \cdot e_{\{\}}) - 2^{3-N} \sum_{C \in C_{I'}} g_I(C)$, where $C_{I'}$ is the set of all cuts C such that $K_{I'}(C) = -1$.
$g_I \cdot e_{\{\}} = 0$, so
$g_I \cdot K_{I'} = -2^{3-N} \sum_{C \in C_{I'}} g_I(C)$.

$K_{I'}(C) = -1$ if and only if for all vertices $v \in I'$, $v \in W_1 \cap C$ or $v \in W_2 \cap C^c$. Thus, $C_{I'}$ is the set of all cuts such that for all vertices $v \in I'$, $v \in W_1 \cap C$ or $v \in W_2 \cap C^c$.

Let D_I be the set of all cuts such that there no vertex v such that $v \notin I$ and $v \in W_1 \cap C$ or $v \in W_2 \cap C^c$. If $C \notin D_I$, then $g_I(C) = 0$.

$g_I \cdot K_{I'} = \sum_{C \in (C_{I'} \cap D_I)} (-1)^{|I \cap W_1 \cap C^c|+|I \cap W_2 \cap C|}$.

If I' contains a vertex not in I, then $C_{I'} \cap D_I$ is empty. If I' is a subset of I and I contains a vertex v not in I', then v can either be in C or C^c, and these cuts cancel out, so $g_I \cdot K_{I'} = 0$. Finally, if $I = I'$, then $g_I \cdot K_{I'} = 1$, as needed.

For any nonempty I', $K_{I'}(C) = -1$ if $C = W_1$ and $K_{I'}(C) = -1$ if $C = W_2$. Thus, we clearly have that $g_{\{\}} \cdot K_{I'} = 0$, and it is easily checked that $g_{\{\}} \cdot K_{\{\}} = 1$.

Assume these functions are not unique. Then there is a g such that $g \neq 0$ and $g \cdot K_I = 0$ for all I. But the given g_I must be linearly independent, so they form a basis for $\mathbb{R}^{\mathcal{C}}$, so if $g \cdot K_I = 0$ for all I, then $g = 0$. Contradiction. This completes the proof. ☐

Proof of Theorem 1.3. Take $N = 2^k + 2$. Let $m = 2^k$. Regardless of what W_1 and W_2 are, we can find a path P in G from s to t of length $2^k + 1$ that does not have any edge of the form $a \to b$ where $a \in W_2$ and $b \in W_1$. By Lemma 6.17, taking the usual E_1 and E_2, if W is the set all K_I such that $|I| > k$, then any path P' from s' to t' in G' must go through a vertex in W incident with both an edge with label in E_1 and an edge with label in E_2.

Now note that we can remove all K_I such that $|I| > 2k+1$ from W and it will still be valid. To see this, note that it is impossible to go from a K_I where $|I| \leq k$ to a K_I with $|I| > 2k+1$ without either going through t' or using an edge from both E_1 and E_2.

Combining Lemma 6.15 and Lemma 6.13, using Lemma 7.6 to find the corresponding g, we have that $g \cdot g \leq 2^m (m^6) m^{4k} \leq 2^m m^{5k}$ for large enough m. If C differs by more than $2k+1$ from the C where $C = W_2$, then $g(C) = 0$. Also, $\sum_{a' \in V(G')} |J_{a'} \cdot g| \geq 1$.

Let $M^2 = \frac{1}{g \cdot g}$, and let $g' = Mg$. Now $\sum_{a' \in V(G')} |J_{a'} \cdot g| \geq M$.

Since we can freely choose W_1 and W_2, from basic coding theory, we can create at least $\frac{2^m}{m^{5k}}$ mutually orthonormal g', where each $M^2 \geq \frac{1}{2^m m^{5k}}$.

Now if we add more vertices to G, we can still use these same paths using these m vertices and the corresponding g'. If $m \leq N^{\frac{1}{3}}$, then we can pick at least $N^{\frac{1}{2}k}$ distinct subsets of size m of $V(G)\backslash s \backslash t$ such that any two subsets have at most k vertices in common.

Thus, in total, we have $K = N^{\frac{1}{2}k} \frac{2^m}{m^{5k}}$ orthonormal g'. If G' solves directed connectivity on N vertices, following the same reasoning as in the proof of Theorem 5.23, $N' \geq \sqrt{KM^2} = \frac{N^{\frac{1}{4}k}}{m^{5k}}$.

Taking m to be about $N^{\frac{1}{40}}$, we have that
$N' \geq N^{\frac{1}{320} \log N}$ for large enough N. This completes the proof. □

Acknowledgement. The author wishes to thank Boaz Barak and Eli-Ben Sasson for their advice on this research and for their help in editing the article.

References

[1] R. Aleliunas, R. M. Karp, R. J.Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. Proceedings of the 20th Annual Symposium on Foundations of Computer Science, p.218-223, 1979
[2] S. A. Cook and C. W. Rackoff. Space lower bounds for maze threadability on restricted machines. SIAM Journal on Computing, 9(3)636-652, Aug 1980
[3] N. Immerman. Nondeterministic Space is Closed Under Complementation, SIAM J. Comput. 17, pp. 935-938, 1988
[4] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require superlogarithmic depth, Proceedings of ACM STOC'88, pp. 539-550, 1988.
[5] W. Masek. A fast algorithm for the string editing problem and decision graph complexity. Master's Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1976.
[6] R. Raz, P. McKenzie. Separation of the monotone NC hierarchy, Proceedings of the 38th Annual Symposium on Foundations of Computer Science, pp 234-243, 1997
[7] A. Razborov. Lower Bounds for Deterministic and Nondeterministic Branching Programs, Proceedings of the 8th FCT, Lecture Notes in Computer Science, vol. 529, 1991, 47-60.
[8] O. Reingold. Undirected ST-connectivity in Log-Space, STOC 2005.
[9] W. J. Savitch. Relationship between nondeterministic and deterministic tape classes, J.CSS, 4, pp 177-192, 1970
[10] R. Szelepcsenyi. The method of forcing for nondeterministic automata, Bull. EATCS 33, pp. 96-100, 1987
[11] V. Trifonov. An O(log n log log n) space algorithm for undirected st-connectivity, Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, May 2005.

E-mail address: aaron@potechin.org

MIT

ECCC ISSN 1433-8092

http://eccc.hpi-web.de