THE CIS PROBLEM AND RELATED RESULTS IN GRAPH THEORY

RYAN ALWEISS, YANG LIU

Abstract. In this survey, we will show that there are instances of the CIS problem on n vertices which cannot be solved deterministically in $\tilde{\Omega}(\log n)$ time. Until recently, the best known lower bound for this problem was only linear in $\log(n)$. The approach will be to instead lower bound the conondeterministic complexity $\mathbf{coNP^{cc}}(CIS_G)$ through a reduction to a problem in decision tree complexity. Interestingly, this result has an application to graph theory, the negative resolution of the Alon-Saks-Seymour conjecture.

1. Introduction

Given a communication matrix M of a function F, define $\chi_0(F)$ to be the minimal number of rectangles necessary to partition the 0s of M, $\chi_1(F)$ similarly, and $\chi(F) = \chi_0(F) + \chi(F)$.

Clearly the deterministic communication complexity of F is always at least $\log \chi(F)$, and it is always at most $\log^2(\chi(F))$. It was unsolved until recently whether some problems required superlogarithmic amounts of communication. Here, we present some results about the complexity of the clique versus independent set problem (CIS) that imply there are problems with superlogarithmic communication complexity.

In this CIS problem, there is a graph G on n vertices which Alice and Bob both see. Alice has vertices that form a complete subgraph, and Bob has vertices that form an independent set. Alice and Bob wish to see if their sets of vertices intersect or not. A solution of CIS in some time t extends to a solution of any communication problem in the same amount of time. In the communication matrix, blocks of 1 can be realized as vertices, connected if they share a common row. Then the 1-rectangles that form some column are an independent set, and those that intersect some row form a complete graph. If there is some algorithm to figure our whether the complete graph and independent set intersect in time t, then the problem can be solved in time t. As such, proving lower bounds for $\log(\chi_1)$ is equivalent to proving lower bounds for CIS in much the same way that solving NP is equivalent to solving SAT.

In this survey, we will show there are instances of the CIS problem with conondeterministic complexity $\tilde{\Omega}(\log^{1.128}(n))$, where the graph has n vertices. As such, there is a CIS problem with deterministic complexity $\tilde{\Omega}(\log^{1.128}(n))$, or $\tilde{\Omega}(\log^{1.128}(\chi_1(F)))$. In fact, a later result shows that some problems require complexity $\tilde{\Omega}(\log^2(\chi_1(F))$ and $\tilde{\Omega}(\log^{1.5}(\chi(F))$ [GPW 2015].

Furthermore, the CIS problem is related to a conjecture in graph theory, the Alon-Saks-Seymour conjecture. Given a graph G, let $\chi(G)$ denote its chromatic number and $\mathbf{bp}(G)$ denote the minimal number of bipartite graphs into which its edges can be partitioned. The initial conjecture was that $\chi(G) \leq \mathbf{bp}(G) + 1$, but it turns out that showing lower bounds for the CIS problem shows that it is possible to have $\chi(G)$ be more than polynomial in $\mathbf{bp}(G)$. In other words, we have $\chi(G) \geq \mathbf{bp}(G)^m$ for any fixed $m > 0$ and for infinitely many graphs G.

2. General Structure of this Survey

The main objective of the first part of this survey will be to prove the result due to Goos, that $\mathbf{coNP^{cc}}(CIS_G)$ and therefore $\mathbf{P^{cc}}(CIS_G)$ are at least $\tilde{\Omega}(\log^{1.128}(n))$. First, we will develop the theory of decision trees. Then, we will proceed to prove this results. To conclude the first part, we will briefly discuss more recent improvements and examine the results from the literature that we cite.

We will then turn our attention to graph theory, discussing the relations between the CIS_G problem and the Alon-Saks-Seymour conjecture. We will examine how these conjectures from communication complexity statements translate to graph theory. As such we will highlight the rich connection between these two areas of mathematics.

3. Decision Trees

Throughout the remainder of this paper, we will be using results about decision trees. A decision tree is a directed binary tree with a single root whose out-edges are labeled 0 or 1. Leaves are also labeled with 0 or 1. All other vertices are labeled x_j for $1 \leq j \leq n$, and have two out-edges, labeled with 0 and 1 respectively. When on some vertex labeled x_j, one queries x_j and traverses the out-edge from x_j corresponding to its value. As such, a decision tree induces a Boolean function f on n variables. We say that such a decision tree is a decision tree for f. The queries are analogous to bits exchanged in a communication protocol, and thus are important for understand communication complexity. This is made explicit in the appendix.

We construct complexity classes for decision trees. Denote by $\mathbf{P^{dt}}(f)$ the minimal possible height of a decision tree for f. This is called the deterministic *query complexity* of f, because it is the minimal number of queries necessary to always compute f.

We define $\mathbf{NP^{dt}}(f)$, the nondeterministic query complexity, to be the minimal k such that for for any string x of length n with $f(x) = 1$, there exists some k bits of the string that provide enough information to show that $f(x) = 1$. In other words, any y that agrees with x just on those bits has $f(y) = f(x) = 1$. We say these k bits are a *1-certificate* for f. Indeed, $\mathbf{NP^{dt}}(f)$ is also the minimal k such that we can write f as a k-DNF, since the terms of the DNF correspond to 1-ceritificates.

We define $\mathbf{coNP^{dt}}(f)$ similarly, as the minimal k such that whenever $f(x) = 0$ there is an analogous 0-certificate for f. We have $\mathbf{coNP^{dt}}(f) = \mathbf{NP^{dt}}(\neg f)$.

We define $\mathbf{UP^{dt}}(f)$, the unambiguous query complexity, to be the minimal k such that for any string x with $f(x) = 1$, there is a unique 1-certificate of size k. This is also the minimal k such that there exists a k-DNF for f where exactly one clause is satisfied when $f(x) = 1$, and no clauses are satisfied otherwise. Again, this is because the DNFs correspond exactly to 1-certificates.

We can generalize the notion of certificates. An $f(x)$-certificate of size k for a string of x of length n, are a set of k bits of the string that provide enough information to determine the value of $f(x)$. The *certificate complexity* of f is the minimal k such that there is an $f(x)$-certificate of size k for every string.

In this paper, we will be concerned with relationships between the above quantities. It is an easy result that $\mathbf{NP^{dt}}(f) \leq \mathbf{P^{dt}}(f)$. If there exist a tree of height k computing f and if $f(x) = 1$ the values of the at most k variables on the path taken on input x are sufficient to conclude $f(x) = 1$. Replacing f with $\neg f$, $\mathbf{coNP^{dt}}(f) \leq \mathbf{P^{dt}}(f)$.

Proposition 3.1. $\mathbf{P^{dt}}(f)$ is at most quadratic in $\mathbf{UP^{dt}}(f)$. More explicitly, we have the inequality $\mathbf{P^{dt}}(f) \leq \mathbf{UP^{dt}}(f)^2$.

Proof. Consider a specific decision tree with a unique certificate of size $k = \mathbf{UP^{dt}}(f)$ for any x with $f(x) = 1$. For any C_1 and C_2 there exists x_j such that in one certificate $x_j = 0$ and in the other $x_j = 1$. Else, some assignment would satisfy both C_1 and C_2, contradicting uniqueness. Now, take some certificate C_0 (say, the first lexicographic one) and ask for all k of its associated variables. After these variables are determined, any assignment to the remaining variables satisfies exactly one certificate C of size k of the original problem. But because C intersects C_0, to determine whether some assignment satisfies C it now suffices to query at most $k-1$ variables. Indeed, this resulting problem f' has $\mathbf{UP^{dt}}(f') \leq \mathbf{UP^{dt}}(f)-1 = k-1$ and we reduced f to f' by querying k variables. We rename $k-1$ as k and repeat. Through this reduction argument, in fact we find that in fact

$$\mathbf{P^{dt}}(f) \leq \sum_{k=1}^{\mathbf{UP^{dt}}(f)^2} k = \frac{\mathbf{UP^{dt}}(f)^2 + \mathbf{UP^{dt}}(f)}{2}.$$

We do not need the full strength of this result. We have that

$$\mathbf{coNP^{dt}}(f) \leq \mathbf{P^{dt}}(f) \leq \frac{\mathbf{UP^{dt}}(f)^2 + \mathbf{UP^{dt}}(f)}{2} \leq \mathbf{UP^{dt}}(f)^2,$$

so $\mathbf{coNP^{dt}}(f) \leq \mathbf{UP^{dt}}(f)^2$. We will show that 2 cannot be replaced by 1.128. □

4. Reductions to Decision Tree Complexity

The goal of this section will be reduce our problem of communication complexity to a problem relating to query and decision tree complexity. First,

note that $\mathbf{UP^{cc}}(CIS_G) = \log n$, as the set of vertices constitutes a set of 1-certificates. Because it is known that the CIS_G faily of problems is *complete* for unambiguous communication, we can focus on constructing a family of functions F that exhibit an exponent of $\alpha > 1.128$ separation between $\mathbf{UP^{cc}}$ and $\mathbf{coNP^{cc}}$. This would imply that $\mathbf{coNP^{cc}}(CIS_G) \geq O((\log n)^{1.128})$. We rephrase this in the following result.

Theorem 4.1. There is an infinite family of functions F such that $\mathbf{coNP^{dt}}(F) \geq \mathbf{UP^{dt}}(F)^\alpha$ for some constant $\alpha > 1.128$.

We claim that it suffices to show the corresponding theorem for query complexity by using a result of [GLM$^+$14].

Theorem 4.2. There is an f such that $\mathbf{UP^{dt}}(f) \geq \mathbf{coNP^{dt}}(f)^\alpha$ for some $\alpha > 1.128$.

We introduce some notation before stating the next theorem. Let $g : \{0,1\}^b \times \{0,1\}^b \to \{0,1\}$ be a function taking a pair of inputs of length $b = \Theta(\log n)$. Define $F = f \circ g^n$ as a function from $\{0,1\}^{bn} \times \{0,1\}^{bn} \to \{0,1\}$ as follows. Split the input into strings $x_1, x_2, \ldots, x_n$, and $y_1, y_2, \ldots, y_n$ of length b. Then let

$$F := f(g(x_1, y_1), g(x_2, y_2), \ldots, g(x_n, y_n)).$$

Theorem 4.3 (GLM$^+$14)**.** There is a gadget g on $b = \Theta(\log n)$ bits that such that for all $f : \{0,1\}^n \to \{0,1\}$, we have that

$$\mathbf{coNP^{cc}}(f \circ g^n) \geq \Omega(\mathbf{coNP^{dt}}(f) \cdot b).$$

We also a lower bound on $\mathbf{UP^{dt}}$, so we prove the following proposition.

Proposition 4.4. For all gadgets g on $b = \Theta(\log n)$ bits and all functions $f : \{0,1\}^n \to \{0,1\}$, we have that

$$\mathbf{UP^{cc}}(f \circ g^n) \leq \mathbf{UP^{dt}}(f).$$

Proof. Simply simulate the decision tree and use b queries each for Alice and Bob each time we need to access a bit of the input. By using b queries, they can find the value of $g(x_i, y_i)$ for any i. It isn't hard to check that if the decision tree is unambiguous, then this protocol is also unambiguous. □

Using the above results along with Theorems 4.1, 4.2, and 4.3, we can deduce our desired result up to a few extra $\log n$ factors. These will end up being insignificant though, as $\mathbf{UP^{dt}}(f)$ will be polynomial in n in our constructions. As a final note before continuing, we will use larger, not necessarily binary, alphabets in our construction. These can be easily converted back to binary, and we will make a note of it after the proof of Theorem 4.3.

5. Separation Factor of 2 Using the Projective Plane

Here we're going to describe an example that has a constant factor of 2 separation between $\mathbf{UP^{dt}}$ and $\mathbf{coNP^{dt}}$. The example is based on projective planes, and later on, we will do *recursive composition* on it.

Definition 5.1. A *finite projective plane* H is hypergraph with $n = k^2 - k + 1$ nodes and hyperedges. Each node is part of k edges, and each edge contains k nodes. Also, each pair of edges intersects exactly at one node. Finite projective planes exist for all k such that $k - 1$ is a prime power.

Definition 5.2. Now we can define a *symmetric incidence ordering* on H. For each node in H, we create an ordering on the k edges that contain it, and for each edge, we create an ordering on the k nodes that it contains. We want this ordering to be symmetric, that is, if an edge e is the i-th edge in the ordering of node v, then v must be the i-th node in the ordering of edge e.

Proposition 5.3. Symmetric incidence ordering exist for all finite projective planes.

Proof. Consider constructing the bipartite incidence graph of H with nodes on the left and edges on the right. We prove the more general statement that a d-regular bipartite graph $G = (A, B)$ (where A denotes the vertices on the left, and B denotes the vertices on the right) can be partitioned into d perfect matchings. The statement is obvious for $d = 1$. For $d > 1$, we claim that for any set S of vertices in A, they are incident to at least $|S|$ vertices in B. Say that S is incident to a set S' in B. Then it is clear that

$$d|S| = \sum_{s \in S} \deg(s) \leq \sum_{s' \in S'} \deg(s') = d|S'| \implies |S| \leq |S'|,$$

so Hall's condition is satisfied. Therefore, we can remove one perfect matching from this graph to reduce it to a $(d-1)$-regular graph and induct. It's easy to see how we can create the symmetric incidence ordering from these perfect matchings.

□

Our goal now is to extract a query problem f from a projective plane H. To do this, we make the nodes of H correspond to input variables that have values in the set $\{0\} \cup [k]$. These values should be understood as *pointers*, where a node with input i points to the i-th edge in its symmetric incidence ordering. 0 is interpreted as a *null pointer*. Say that an assignment $x : V(H) \to \{0\} \cup [k]$ *satisfies* an edge e if all incident nodes $v \in e$ point to e according to the input $x(v)$. Note that each x can satisfy at most one edge, as every pair of edges intersect at some node, and this node can only point to one of the two edges.

Define a function $f : (\{0\} \cup [k])^n \to \{0, 1\}$ such that $f(x) = 1$ iff x satisfies some edge. By the above discussion, there is a clear set of k-cost $\mathbf{UP^{dt}}$ certificates that correspond exactly with the edges of H. This function f doesn't have large $\mathbf{coNP^{dt}}$ complexity though; the certificate complexity of $f(\vec{0}) = 0$ is small, as k positions suffice. We fix this by "weighting" the cost of querying the inputs.

Input Weights. We will now describe how to change f so that querying to inputs to x is harder. More specifically, we will alter f so that if we want to decide whether $x_j = i$ or not, we will need $w(i)$ queries (instead of 1),

where $w : [k] \to \mathbb{N}$ is some weighting function that we'll choose later. To do this, we set $f_w = f \circ g_w^n$, where g_w will implement the desired weights. More specifically, for the integer $m = \max w([k])$, let $g_w : (\{0\} \cup [k])^m \to \{0\} \cup [k]$ be defined the following way. If $x = \vec{0}$, then $g(x) = 0$. Otherwise, let x_j be the first nonzero coordinate of x, and say that $x_j = i$. If $x_j \leq w(i)$, let $g_w(x) = i$; otherwise, let $g_w(x) = 0$. It is clear that we need at least $w(i)$ queries to determine whether $g_w(x) = i$, and that this is sufficient. The following 2 claims are easy to see from the construction.
$(P1)$ The certificate complexity of "$g_w(\vec{0}) \neq i$" is $w(i)$.
$(P2)$ If $\hat{w}(i) = \ell \cdot w(i)$ for all i, then $\mathbf{coNP^{dt}}(f_{\hat{w}}) = \ell \cdot \mathbf{coNP^{dt}}(f_w)$.

Our next goal is to use this weighting idea described above to create a (small) separation between $\mathbf{UP^{dt}}$ and $\mathbf{coNP^{dt}}$ using the function f associated with a projective plane H above. To do this, we set $w(i) = i$ and $f_w = f \circ g_w^n$.

Proposition 5.4. The following 2 inequalities hold, and thus show an asymptotic factor of 2 separation between $\mathbf{UP^{dt}}$ and $\mathbf{coNP^{dt}}$ for f_w.

$$\mathbf{UP^{dt}}(f_w) \leq \frac{k(k+1)}{2},$$

$$\mathbf{coNP^{dt}}(f_w) \geq k^2 - k + 1.$$

Proof. To prove the upper bound, we can describe a certificate for f_w in the following way. We have to determine for an edge e and input x, whether it is satisfied by x. We do this in the direct way: check for the i-th node incident to the edge e, whether that node points to i according to x. By construction, this takes $w(i) = i$ queries. Therefore, $\mathbf{UP^{dt}}(f_w) \leq \sum_{i=1}^{k} i = \frac{k(k+1)}{2}$.

We will now lower bound $\mathbf{coNP^{dt}}(f_w)$. Let $g_{w,v}$ denote the gadget in g_w^n that corresponds to the node v. Note that in order to show that $f_w(\vec{0}) = 0$ for the input x, for all edges e we must exhibit an integer i such that the i-th incident node to edge e does not point to e according to x. More explicitly, if we let v denote the i-th incident node to e, we must prove that $g_{w,v}(x_v) \neq i$, where x_v denotes the restriction of the input x to just the part corresponding to the node v.
Now we prove a lemma giving a lower bound on the number of queries needed.

Lemma 5.5. The certificate complexity of showing that $g_{w,v}(x_v) \notin S$ for some set $S \subseteq [k]$ is at least $|S|$.

Proof. Let $m = \max_{s \in S} s$. Then by definition, we must show that $g_{w,v}(x_v) \neq m$, which requires $w(m) = m$ size certificates. □

To finish, say that we must show that $g_{w,v}(x_v) \in S_v$ for some sets S_v. Since different gadgets are independent, we'll need certificates of size at least $\sum |S_v| = n = k^2 - k + 1$, as every of the n edges contributes to one of the sets S_v.

□

6. The General Result

In this section, we will prove the main result, restated below. The key idea is we can "recursively compose" the inequalities that we proved in above section.

Main Result. There is a function f such that $\mathbf{coNP^{dt}}(f) \geq \mathbf{UP^{dt}}(f)^{\alpha}$ where $\alpha > 1.128$ is some constant.

In this section, we will consider pairs of functions and weights (f, w), where w is once again the weighting function on the inputs of f. We define $f_w = f \circ g_w^n$ as in the previous section, where f_w corresponds with the pair (f, w). Define a *0-avoiding* $\mathbf{UP^{dt}}$ decision tree for f to be one whose certificates only contain letters from Σ, the input alphabet, and not 0. On the other hand, certificates for f_w can contain 0's, but only ones that come from the gadgets g_w^n. Intuitively, we care about the certificates for f not having 0's, but because the g_w^n is simply a way to create our desired weighting function, we don't care about whether the certificates there contain 0's or not. We define a bit of new notation now, given the above discussion. Define $\mathbf{UP}_{\star}^{\mathbf{dt}}(f_w)$ to be the minimum complexity 0-avoiding $\mathbf{UP^{dt}}$ decision tree for a function f_w. Also, define $\mathbf{coNP}_{\star}^{\mathbf{dt}}(f_w)$ to be the minimum cost that we need to certify that $f_w(\vec{0}) \neq 0$.

The main idea of this section will be the recursively improve the bound given in the above section. We do this in the following way.

Proposition 6.1. Given a pair (f, w) of functions, we can construct a new pair (f', w') of functions such that

$$\mathbf{UP}_{\star}^{\mathbf{dt}}(f'_{w'}) \leq \frac{k(k+1)}{2} \cdot \mathbf{UP}_{\star}^{\mathbf{dt}}(f_w) \text{ and}$$
$$\mathbf{coNP}_{\star}^{\mathbf{dt}}(f'_{w'}) \geq (k^2 - k + 1) \cdot \mathbf{coNP}_{\star}^{\mathbf{dt}}(f_w)$$

We do this in two steps. First, we construct a new function $\hat{f}_{\hat{w}}$ that has a larger size alphabet for both the input and output than f_w. Then we reduce the output size back to $\{0, 1\}$ by composing $\hat{f}_{\hat{w}}$ with another function based on the projective plane H. This will all be done more explicitly in the following 2 sections.

First step: Expanding the alphabet size, construction of $\hat{f}_{\hat{w}}$

Let Σ denote an alphabet, and σ throughout denote characters in the alphabet. Starting with the function $f : (\{0\} \cup \Sigma)^N \to \{0, 1\}$, we will construct a new function $\hat{f} : (\{0\} \cup (\Sigma \times [k]))^N \to \{0\} \cup [k]$ that exists the same factor of separation between $\mathbf{UP^{dt}}$ and $\mathbf{coNP^{dt}}$ as f. Also define $\pi_{\Sigma} : \{0\} \cup (\Sigma \times [k]) \to \Sigma$, $\pi_k : \{0\} \cup (\Sigma \times [k]) \to [k]$ to be the natural restriction maps. Define $\pi_{\Sigma}(0) = \pi_k(0) = 0$ to extend the function to its full domain. By abuse of notation, define π_{Σ}, π_k on $(\{0\} \cup (\Sigma \times [k]))^N$ by applying the functions pointwise.

We now define $(\hat{f}, \hat{w})$ as follows. Take an $x \in (\{0\} \cup (\Sigma \times [k]))^N$. If $f(\pi_\Sigma(x)) = 0$, define $\hat{f}(x) = 0$. Otherwise, let $\mathcal{T}$ denote a $\mathbf{UP^{dt}}$ certificate for $\pi_\Sigma(x)$, and let $S \subseteq [N]$ be the inputs that the certificate $\mathcal{T}$ looks at. Note that $x_s \neq 0$ for all $s \in S$, as the certificate is 0-avoiding. If there exists an i such that $\pi_k(x_s) = i$ for all $s \in S$, define $\hat{f}(x) = i$, otherwise set $\hat{f}(x) = 0$. Finally, define $\hat{w} : \Sigma \times [k] \to \mathbb{N}$ as $\hat{w}(\sigma, i) = i \cdot w(\sigma)$, where σ is a character in Σ.

Proposition 6.2. There is a 0-avoiding $\mathbf{UP^{dt}}$ decision tree certifying that $\hat{f}_{\hat{w}}(\cdot) = i$ that has complexity at most $i \cdot \mathbf{UP}^{\mathbf{dt}}_\star(f_w)$. Also, certifying that $\hat{f}_{\hat{w}}(\vec{0}) \neq i$, requires complexity at least $i \cdot \mathbf{coNP}^{\mathbf{dt}}_\star(f_w)$.

Proof. To prove the upper bound, we do the most natural thing. More specifically, let $\mathcal{T}$ be the original certificate for f_w. We check the exact same thing as $\mathcal{T}$ tells us to check, except we also recursively check whether $\pi_k(x_s) = i$ for all $s \in S$. This costs $i \cdot \mathbf{UP}^{\mathbf{dt}}_\star(f_w)$ as the weights are now exactly i times heavier by construction.
To prove the lower bound, note that certifying that $\hat{f}_{\hat{w}}(\vec{0}) \neq i$ is the exact same as certifying whether $f_w(\vec{0}) \neq 0$. More specifically, consider the restriction of $\hat{f}_{\hat{w}}$ onto the domain $0 \cup (\Sigma \cup \{i\})$. It isn't hard to see that $\hat{f}_{\hat{w}}$ on this domain behaves exactly as f_w on this domain, and the output of $\hat{f}_{\hat{w}}$ is i if and only if the corresponding output of f_w is 1. Now, since the weights were all multiplied by i, the necessary cost here is at least $i \cdot \mathbf{coNP}^{\mathbf{dt}}_\star(f_w)$. □

Second step: Making the output binary, Composition with h

Define the function h to be the same as the function f defined in Section 5. As a reminder, $h : (0 \cup [k])^n \to \{0, 1\}$ is defined so that $h(x) = 1$ if and only if x satisfies some edge. I claim that setting $f' = h \circ \hat{f}^n_{\hat{w}}$, and $w' := \hat{w}$ satisfies the properties in Proposition 6.1.

Proof. For both parts we will simulate the proof of Proposition 5.4.
We first show the upper bound. For the edge e and i from 1 to k, we have to check whether i-th node incident to e points to e. To do this, we have to recursively check whether $\hat{f}_{\hat{w}}(\cdot) = i$, which requires cost $i \cdot \mathbf{UP}^{\mathbf{dt}}_\star(f_w)$ by Proposition 6.2. Therefore, the total cost is at most

$$\sum_{i=1}^{k} i \cdot \mathbf{UP}^{\mathbf{dt}}_\star(f_w) = \frac{k(k+1)}{2} \cdot \mathbf{UP}^{\mathbf{dt}}_\star(f_w).$$

To prove the lower bound, we once again lower bound the cost needed to certify that $f'_{w'}(\vec{0}) = 0$. In order to stay as close as possible to our proof of Proposition 5.4, it will be useful to think of the $\hat{f}_{\hat{w}}$ as "gadgets". We prove a variation of the lemma from the proof of Proposition 5.4.

Lemma 6.3. The certificate complexity of showing that $\hat{f}_{\hat{w}}(\vec{0}) \notin S$ is at least $|S| \cdot \mathbf{coNP}^{\mathbf{dt}}_\star(\hat{f}_{\hat{w}})$.

Proof. Let $m = \max_{s \in S} s$. Then we must certify that $\hat{f}_{\hat{w}}(\vec{0}) \neq m$, which requires cost at least $m \cdot \mathbf{coNP}^{\mathbf{dt}}_{\star}(\hat{f}_{\hat{w}}) \geq |S| \cdot \mathbf{coNP}^{\mathbf{dt}}_{\star}(\hat{f}_{\hat{w}})$, by Proposition 6.2. □

Now we can follow through with the proof in the same way as in Proposition 5.4. Indeed, say that q_v queries are made to the v-th gadget in $\hat{f}^n_{\hat{w}}$. Since every edge contributes a pointer, $\sum q_v = n = k^2 - k + 1$. Since each gadget is independent, we need at least $\sum q_v \cdot \mathbf{coNP}^{\mathbf{dt}}_{\star}(\hat{f}_{\hat{w}}) = (k^2 - k + 1) \cdot \mathbf{coNP}^{\mathbf{dt}}_{\star}(\hat{f}_{\hat{w}})$ queries by Lemma 6.3. □

Overall Analysis and Wrap-Up

Define (f^i, w^i) to be the functions gotten from the i-th stage of recursive composition as described. The following claims are easy to see from the construction.

- The largest weight in w^i is k^i.
- The alphabet has size $1 + k^i$, where the 1 comes from the 0 character.
- f^i takes $n^i = (k^2 - k + 1)^i$ inputs.
- $\mathbf{UP}^{\mathbf{dt}}_{\star}(f^i_{w^i}) \leq \left(\frac{k(k+1)}{2}\right)^i$.
- $\mathbf{coNP}^{\mathbf{dt}}_{\star}(f^i_{w^i}) \geq (k^2 - k + 1)^i$.

Therefore, as we take i to be large, we get an infinite class of graphs such that

$$\mathbf{coNP}^{\mathbf{dt}}_{\star}(f^i_{w^i}) \geq \mathbf{UP}^{\mathbf{dt}}_{\star}(f^i_{w^i})^{\beta}$$

for $\beta = \log_{k(k+1)/2}(k^2 - k + 1)$. Choosing $k = 8$, we can $\beta = \log_{36} 57 > 1.128$. Before we are completely done, we need to note a slightly technicality: our alphabets are not binary sized. Luckily, this is easily resolvable: the query complexity increases by at a factor of $\log \Sigma = O(i \log k)$ when we convert to binary alphabets. Therefore, the other terms in our final inequality dominate. Specifically, we can still achieve a separation exponent of α for any $\alpha < \beta$. This is enough to finish.

7. CI and CIS

First we'll need some preliminaries. We define a CI-separator for a graph G to be a set of cuts of G such that for every disjoint clique and independent set in the graph, there exist a cut that separates them. If there is a CI-separator of size m for some graph G, then $\mathbf{coNP}^{\mathbf{cc}}(CIS_G) = O(\log(m))$, because an external verifier can convince Alice and Bob their clique and independent set are separated with $O(\log(m))$ bits. As such if there is a CI-separator of polynomial size for some graph G, $\mathbf{coNP}^{\mathbf{cc}}(CIS_G) = O(\log(n))$. Until recently, it was only a conjecture (the CI conjecture) whether or not every graph had a polynomial size separator. Of course, because $\mathbf{coNP}^{\mathbf{cc}}(CIS_G)$ can be superlogarithmic this conjecture is known to be false by the above work. It turns out it is an unsolved conjecture that every graph containing a fixed subgraph H has a polynomial size separator, and random graphs are known to have polynomial size separators in general.

8. Alon-Saks-Seymour and motivations from Graham-Pollak

Here a proper coloring of a graph is one where no two adjacent vertices have the same color, $\chi(G)$ is the minimal number of colors needed to properly color G, and a bipartite graph has chromatic number 2. Also $\mathbf{bp}(G)$ is the minimal number of complete bipartite graphs the edges of G can be paritioned into.

The Alon-Saks-Seymour conjecture in its weakest (polynomial) form states that if $\mathbf{bp}(G) = k$, $\chi(G)$ is at most polynomial in k, i.e. there is some polynomial P so that $\chi(G) \leq P(\mathbf{bp}(G))$. The initial stronger version states that $\chi(G) \leq \mathbf{bp}(G) + 1$ and was motivated by the complete graph case, the result that you need $n-1$ complete bipartite graphs to partition the edges of the completele graph K_n. This is an extremely nice result known as Graham-Pollak.

Proposition 8.1. A complete graph K_n cannot be partitioned into fewer than $n-1$ complete bipartite graphs.

Proof. Assume that at most $n-2$ complete bipartite graphs paritioned the edges of K_n. We assign weights v_i to the vertices of K_n so that $\sum v_i = 0$, and for any of the complete bipartite graphs H between $S \subset \{1, 2, ...n\}$ and some other set of vertices, $\sum_{i \in S} v_i = 0$. There are at most $n-1$ equations, so we can find some values of v_i not all 0 which satisfies all this. But then the sum of the weights on every edge is

$$\sum v_i v_j = \frac{(\sum v_i)^2 - \sum v_i^2}{2} = \frac{-\sum v_i^2}{2} < 0.$$

Yet the sum of the weights of the edges on any complete bipartite graph is 0 so the sum of the weights on all edges must also be 0, contradiction. □

Note a K_n can be partitioned into $n-1$ bipartite graphs. If v_i is connected to v_j for $i < j$, put the edge between them in the ith bipartite graph.

9. Alon-Saks-Seymour from CI

We will now prove that the polynomial Alon-Saks-Seymour conjecture implies the CI conjecture. It actually turns out that the two results are equivalent, but it is useless to show that something is implied by a statement we have already established to be false. Because we have that the CI conjecture is false, we will get the polynomial Alon-Saks-Seymour conjecture is also false.

First, an oriented complete bipartite graph is one on a set of vertices A_i and B_i so that there is a directed edge from every A_j to every B_k. We say a set of oriented complete bipartite graphs are *properly packed* into G if every edge XY in G is such that e appears in some oriented bipartite graph, but there do not exist two different ordered bipartite graphs which both have a directed edge from X to Y or from Y to X. It is permissible that one has a directed edge from X to Y and another from Y to X.

Let $\mathbf{bp}_{OR}(G)$ be the minimal number of oriented complete bipartite graphs that are necessary to properly pack G.

Also define $\mathbf{bp}_2(G)$ to be the minimal of complete bipartite graphs needed to cover the edges of G so that every edge is covered at least once and at most twice. Clearly $\mathbf{bp}_2(G) \leq \mathbf{bp}_{OR}(G) \leq \mathbf{bp}_G$.

We call the assertion that $\mathbf{bp}_{OR}(G)$ is at most polynomial in $\chi(G)$ the *oriented Alon-Saks-Seymour conjecture.*

Proposition 9.1. We have $\mathbf{bp}(G) \leq P(\mathbf{bp}_2(G))$ for some polynomial P, assuming the Alon-Saks-Seymour conjecture.

Proof. Say $\mathbf{bp}_2(G) = k$, so we have k complete bipartite graphs B_1,... B_k that cover the edges of G, with every edge covered at least once and at most twice. Let H be the graph formed by taking G and removing edges that are covered by the B_i only once.

We first claim $\mathbf{bp}(H) \leq k^2 - k$. We also direct the edges of each complete bipartite graph B_i arbitrarily from B_i^- to B_i^+ to make it an oriented complete bipartite graph. If some edge e of H is covered by the two bipartite graphs B_i and B_j, label the edge with (B_i, B_j, u) where $i < j$. Assign $u = 1$ if e is directed the same way in B_i and B_j, and $u = -1$ else otherwise. Of course here u depend on the choice of i, j, and the edge and are not absolute constants. Note there are $k^2 - k$ possible labels.

Now, for every labeling (B_i, B_j, u) with $i < j$, consider the subgraph of H with this labeling. I claim it is a complete bipartite graph. We consider the subgraph with $(B_i, B_j, 1)$ - the $u = -1$ case is analogous (after all, B_i and B_j were directed arbitrarily). Now, this subgraph is the complete bipartite graph going from $B_i^- \cap B_j^-$ to $B_i^+ \cap B_j^+$.

Every edge of H is uniquely labeled, so the disjoint union of these subgraphs is all of H. Hence, $\mathbf{bp}(H) \leq k^2 - k$, the number of labels.

By the Alon-Saks-Seymour conjecture the chromatic number of H is polynomial in $k^2 - k$, and thus polynomial in k. We can therefore partition the common vertex set V of G and H into sets $(S_1, ...S_t)$ which are independent in H, where t is polynomial in k. So every edge with both vertices inside S_i is covered exactly once by B_1,... B_k. For each $1 \leq i \leq t$, the sets $B_j \cap S_i$ for $1 \leq j \leq k$ thus properly pack into the induced graph $G[S_i]$ on the vertices of S_i. Now, we use Alon-Saks-Seymour again, so $\chi(G[S_i])$ is at most polynomial in k. Say we color all S_j with a polynomial number of colors in k so that every $\chi(G[S_i])$ is properly colored.

To finish, we define a coloring on G, the product of this coloring and the coloring from H. We give the vertex $v \in S_i$ the color (a, b) where a is the color of $S_i \in H$ and b is the color of $v \in G[S_i]$. Clearly this gives us a number of colors that is polynomial in k. Now, if v and w are incident but have the same color, then they lie in the same S_i. Since v, w are incident and are both in S_i, we have that v and w are incident in $G[S_i]$. But then v and w do not have the same color in $G[S_i]$, a contradiction. Hence, this coloring of G is proper and uses a number of colors that is polynomial in k. We obtain the desired result.

□

Assuming the polynomial Alon-Saks-Seymour conjecture, we then have that since $\chi(G)$ is polynomial in $\mathbf{bp}(G)$, it is also polynomial in $\mathbf{bp}_2(G)$ and hence in $\mathbf{bp}_{OR}(G)$.

As such, the polynomial Alon-Saks-Seymour conjecture implies the polynomial oriented Alon-Saks-Seymour conjecture.

Proposition 9.2. The polynomial oriented Alon-Saks-Semyour conjecture implies the CI conjecture.

Proof. Given a graph G on n vertices, we desire to verify the CI conjecture on it - to find a polynomial size family of cuts.

Again we will construct a graph H, but this time with different properties. The vertices of H are pairs of cliques and independent sets from G that do not share a common vertex. Two vertices h_1 corresponding to clique K_1 and independent set I_1 and h_2 of H corresponding to K_2 and I_2 are connected if and only if K_1 intersects I_2 or K_2 intersects I_1.

I claim $\mathbf{bp}_{OR}(H) \leq n$. For any $v \in G$, let A_v be the set of vertices of H of the form (K, I) with $x \in K$ and let B_v be the vertices of H of the form (K, I) with $x \in I$. Let H_v be the oriented complete bipartite graph going from A_v to B_v. I claim that the graphs H_v for the n vertices v form a proper packing of H. So some edge of H will be covered by a directed edge of H_v from (K_1, I_1) to (K_2, I_2) exactly if $v \in K_1$ and $v \in I_2$, which happens for at most one vertex v. But of course one of (K_1, I_1) and (K_2, I_2) will be covered by a directed edge, because either K_1 meets I_2 or K_2 meets I_1. This yields a proper packing of H of size n. Asssuming oriented Alon-Saks-Seymour, $\chi(H) \leq P(n)$ for some polynomial n. Examine the vertices of H which have a specific color, say blue, in G. Let K_b be the union of all K for which (K, I) is blue for some I, and I_b is the union of all I for which (K, I) is blue for some K. Since these (K, I) are the same color in H, they an independent set of H. It follows that K_b and I_b are disjoint, since if (K_1, I_1) and (K_2, I_2) are in an independent set K_1 cannot intersect I_2. Thus, we have can find a single cut that separates all of K_b from I_b. For every blue vertex (K, I) of H, the cut separates K from I. Continuing in this fashion, we have a cut for every color, and every clique K and independent set I in G are separated by the cut corresponding to the color of (K, I) in H. Thus, we have a $P(n)$ size separator for G for any graph G on n vertices and the CI conjecture is established.

□

The results above prove that the polynomial Alon-Saks-Seymour conjecture implies the CI conjecture. Since we know the CI conjecture is false, the polynomial Alon-Saks-Seymour conjecture is false. A result in communication complexity implies a result in graph theory.

10. Further Directions

We know that $\mathbf{coNP}^{\mathbf{dt}}(f) \leq \mathbf{UP}^{\mathbf{dt}}(f)^{\alpha}$ always holds for $\alpha \geq 2$, and it fails to hold for $\alpha \leq \log_{36} 57$, so naturally one can inquire the infimum of α for which this statement is true. It would be very surprising if the constant

$\log_{36} 57$ was optimal. Any lower bound better than $\log_{36} 57 \approx 1.128$ or any upper bound less than 2 would be a breakthrough.

We can ask similar questions about **NP** vs **UP**. Certainly we know that $\mathbf{NP^{dt}}(f) \leq \mathbf{UP^{dt}}(f)$ with equality for CIS. But can **NP** be much smaller than **UP**? Do there exist f such that $\mathbf{NP^{dt}}(f) \leq \mathbf{UP^{dt}}(f)^c$ for any $\alpha > 0$, and if not what is the infimum of α for which this holds?

11. Thanks and Bibliography

Acknowledgment. Thanks to the entire staff of 18.405, for making this possible and making the course wonderful. The authors have enjoyed the course immensely.

References

[1] Sanjeev Arora, Boaz Barak, Computational Complexity: A Modern Approach (2007)
[2] N. Bousquet, A. Lagoutte, S. Thomassé, Clique Verus Independent Set, European Journal of Combinatorics. 40 (2014) 73-92
[3] Mika Goos, Lower Bounds for Clique vs. Independent Set. Report No. 12, Electronic Colloquium on Computational Complexity (ECCC), 2015. URL: http://eccc.hpi-web.de/report/2015/012/ (2015)
[4] Mika Goos, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles are nonnegative juntas. Technical Report TR14-147, Electronic Colloquium on Computational Complexity (ECCC), 2014. URL: http://eccc.hpi-web.de/report/2014/147/.

18.405J / 6.841J Advanced Complexity Theory

Spring 2016