CS 880: Complexity of Counting Problems 1/26/2012

Lecture 2: Introduction - Part 2

Instructor: Jin-Yi Cai Scribe: Tyson Williams

We continue introducing the concepts that will arise frequently in this class, namely, the Holant polynomial, the tractability of (nontrivial) signature sets, the equivalence among problems under holographic transformations, and proving hardness via polynomial interpolation.

1 Review

Last time we introduced the Holant. This function is so important, we introduce it again.

A counting problem is defined by a set $\mathcal{F}$ of functions (these functions are also called signatures). An instance of a counting problem includes a graph $G = (V, E)$ and a function $\pi : V \to \mathcal{F}; v \mapsto f_v$ that labels the vertices of G with a function in $\mathcal{F}$. We combine these three objects into $\Omega = (G, \mathcal{F}, \pi)$ and call it a signature grid. Then the counting problem on the instance Ω is

$$\text{Holant}(\Omega) = \sum_{\sigma: E \to \{0,1\}} \prod_{v \in V} f_v(\sigma \mid_{E(v)})$$

where $f_v(\sigma \mid_{E(v)})$ is the evaluation of f_v on the bits assigned to the incident edges of v.

As an example, if we just have a single function $f = \text{Exact-One}$, then the Holant counts the number of perfect matchings in G. If instead $f = \text{At-Most-One}$, then the Holant counts the number of (not necessarily prefect) matchings. Both of these problems are #P-hard over general graphs, but counting perfect matchings becomes tractable over planar graphs by the FKT algorithm. The Holant is also tractable if f is a Fibonacci signature, which we prove next.

2 Tractability of Fibonacci Signatures

Recall that a symmetric signature $f = [f_0, f_1, \ldots, f_n]$ is Fibonacci if $f_{k+2} = f_{k+1} + f_k$ for $0 \le k \le n-2$. The proof that a set of Fibonacci signatures is tractable basically follows from the fact that the "connection" of two Fibonacci signatures is again a Fibonacci signature.

Theorem 1 (Theorem 3.1 in [2])**.** *Let $\mathcal{F}$ be a subset of the Fibonacci signatures. Then for any signature grid Ω,* Holant(Ω) *is computable in polynomial time.*

Proof. Consider the case depicted in Figure 4 of [2].[1] Now, after connecting F and G together by uniting the edge labeled z, we get another Fibonacci signature. Wait, really?

[1] There are some printing issues in that figure. The middle box should be labeled G, the top-left y should be y_1, and the two y's on the right should be y_4 and y_5 respectively.

This signature does not even appear to be symmetric, so how can it be Fibonacci? Well, we first show that it is symmetric and then use this fact to show that it is indeed Fibonacci.

We have

$$H(y_1, \ldots, y_s, y_{s+1}, \ldots, y_{s+t}) = \sum_{z \in \{0,1\}} F(y_1, \ldots, y_s, z) G(y_{s+1}, \ldots, y_{s+t}, z)$$

and need to show that it is symmetric. Fix an assignment for the y_i's. It suffices to show that any 0 and 1 gives the same evaluation. The easy case is if both bit are inputs to F or both inputs to G. Since F and G are symmetric, this changes nothing. The interesting case is when one bit is an input to F and the other is an input to G.

Say that the bits being swapped are for the inputs y_1 and y_{s+1}. We suppress the other inputs (since their assignment has been fixed) and want to show

$$H(y_1 = 0, y_{s+1} = 1) = H(y_1 = 1, y_{s+1} = 0).$$

Expanding the left side, we get

$$\begin{aligned}
H(0,1) &= F(0,0)G(1,0) + F(0,1)G(1,1) \\
&= F(0,0)G(1,0) + F(0,1)\left(G(0,0) + G(0,1)\right) && (G \text{ is Fibonacci}) \\
&= F(0,0)G(1,0) + F(0,1)G(0,0) + F(0,1)G(1,0) && (G \text{ is symmetric}) \\
&= F(1,0)G(0,0) + \left(F(0,0) + F(0,1)\right)G(1,0) && (F \text{ is symmetric}) \\
&= F(1,0)G(0,0) + \left(F(0,0) + F(0,1)\right)G(0,1) && (F \text{ is Fibonacci}) \\
&= F(1,0)G(0,0) + F(1,1)G(0,1) \\
&= H(1,0),
\end{aligned}$$

where z is the second input to F and G.

Now to prove that H is Fibonacci, we proceed similarly by fixing all by two inputs, one in F and one in G. Then

$$\begin{aligned}
H(0,0) + H(0,1) &= F(0,0)G(0,0) + F(0,1)G(0,1) + F(0,0)G(1,0) + F(0,1)G(1,1) \\
&= F(0,0)G(1,1) + F(0,1)G(0,1) + F(0,1)G(1,1) && (G \text{ Fibonacci}) \\
&= F(0,1)G(0,1) + F(1,1)G(1,1) && (F \text{ Fibonacci}) \\
&= F(1,0)G(1,0) + F(1,1)G(1,1) && (F \text{ and } G \text{ symmetric}) \\
&= H(1,1).
\end{aligned}$$

Notice that we use that H is symmetric by when only considering $H(0,0) + H(0,1)$ and not $H(0,0) + H(1,0)$.

By preforming the above operation, we appear to be going backwards since we are increasing the number of variables with each application. However, there is a second operation depicted in Figure 5 of [2] that decreases the number of variables by two. For this operation, it is easy to see that the resulting signature H is both symmetric and Fibonacci. □

Later we will see that Fibonacci signatures are not just "accidentally" good.

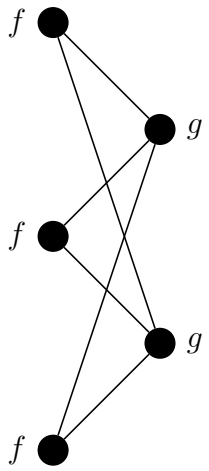

Figure 1: A bipartite graph.

3 Holographic Transformations

We can extend the tractability of the Fibonacci signatures for more problems by a holographic transformation. A holographic transformation between to problems shows that they have the exact same Holant evaluation for all signature grids. What follows is an "executive summary" of holographic transformations. For all the gory details, see [1].

The proper way to view holographic transformations is in the language of tensors. A tensor of vectors $u = (u_1, \ldots, u_k)$ and $v = (v_1, \ldots, v_\ell)$ is

$$u \otimes v = (u_1v_1, u_1v_2, \ldots, u_1v_\ell, u_2v_1, u_2v_2, \ldots, u_2v_\ell, \ldots, u_kv_1, u_kv_2, \ldots, u_kv_\ell).$$

The tensor produce is also defined for matrices. If $M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ and $N = \begin{bmatrix} e & f \\ g & h \end{bmatrix}$, then

$$M \otimes N = \begin{bmatrix} ae & af & be & bf \\ ag & ah & bg & bh \\ ce & cf & de & df \\ cg & ch & dg & dh \end{bmatrix}.$$

Holographic transformations are properly viewed as happening in a bipartite graph. Consider the graph is Figure 1. The vertices on the left are assigned the signature f, a (row) vector of length $2^2 = 4$. The vertices on the right are assigned the signature g, a (column) vector of legnth $2^3 = 8$. The general idea is to combine these signatures on each side of the bipartite graph into one signature. The combining operation is the tensor product. On the left side, we have $f \otimes f \otimes f$, a (row) vector of length $4^3 = 64$. On the right side, we have $g \otimes g$, a (column) vector of length $8^2 = 64$. Then the inner product $\langle f \otimes f \otimes f, g \otimes g \rangle$ is precisely the Holant of this signature grid. In quantum computing, some know this as a tensor contraction.

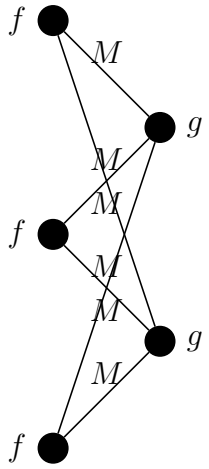

Figure 2: A bipartite graph with the transformation on f visable.

And just like quantum computing, we can change the basis of computation in which this inner product takes place. Let M be a 2-by-2 invertible matrix. Then we can change f to $fM^{\otimes 2}$. This gives some "holographic mix" of the values of f and since M is invertible, we have not lost any information. Visually, we can picture M as sitting on the edges adjacent to f (see Figure 2). The matrix product distributes over the tensor product, so

$$(f \otimes f \otimes f)\left(M^{\otimes 2} \otimes M^{\otimes 2} \otimes M^{\otimes 2}\right) = fM^{\otimes 2} \otimes fM^{\otimes 2} \otimes fM^{\otimes 2} = f' \otimes f' \otimes f',$$

where f' is our transformed function!

So as to preserve the evaluation of the Holant, we do the inverse transformation to the g's. Both transformations can be pictured in Figure 3. The right side becomes

$$(g \otimes g)\left(\left(M^{-1}\right)^{\otimes 3} \otimes \left(M^{-1}\right)^{\otimes 3}\right) = \left(g\left(M^{-1}\right)^{\otimes 3} \otimes g\left(M^{-1}\right)^{\otimes 3}\right) = g' \otimes g'.$$

Thus, we have

$$\langle f \otimes f \otimes f, g \otimes g\rangle = \langle (f \otimes f \otimes f)M^{\otimes 6}, \left(M^{-1}\right)^{\otimes 6}(g \otimes g)\rangle = \langle f' \otimes f' \otimes f', g' \otimes g'\rangle.$$

So, what do we have? We have a *constant time* reduction that two counting problems are exactly the same!

4 Polynomial Interpolation

The final ingredient that will be used extensively in this class is polynomial interpolation. This technique is used to prove a counting problem #P-hard and we explain it with an example.

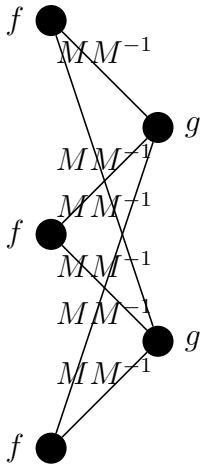

Figure 3: A bipartite graph with the full holographic transformation visible. (Each MM^{-1} is supposed to be on an different edge.)

Theorem 2 (Theorem 6.1 of [2])**.** *Holant on 3-regular graphs with signature* $[0, 1, 1, 0]$ *is* #P*-hard.*

Proof. We also write the signature $[0, 1, 1, 0]$ as $\text{NOT-ALL-EQUAL}_3 = \text{NAE}_3$. Every vertex is a Not-All-Equal clause of size 3. Every edge is a variable that is read twice, once by the constraint on each indicent vertex.

It is already known that #NAE-SAT is #P-hard (since NAE-SAT is NP-hard by Schaefer's dichotomy). This problem has no read restrictions on the variables. We are only allowed to construct gadgets with vertices having the signature $[0, 1, 1, 0]$. However, suppose that we also had the signature $[1, 0, 0, 1]$, the equality function on three bits. Then we could solve instances of #NAE-SAT as follows. We already have the NAE constraint. What we are missing is unbounded reads on variables and the ability to negate a variable. Well, Figures 6 and 8 of [2] show how to construct equality gates of arity 1 and arity 3 or more respectively, which removes the bound on variable reads and Figure 7 implements a NOT gate. Thus, Holant on 3-regular graphs with signatures $\{[0, 1, 1, 0], [1, 0, 0, 1]\}$ is #P-hard. Now, there is no way to directly construct a gadget with signature $[1, 0, 0, 1]$ using only the signature $[0, 1, 1, 0]$. Instead, we reduce from this counting problem via polynomial interpolation.

For the time being though, suppose we still had $[1, 0, 0, 1]$ on some vertices $V' \subset V$ with $|V'| = n$. Let x_j be the number of edge assignments where exactly j vertices in V' receive the assignment $(0, 0, 0)$ or $(1, 1, 1)$ and all the vertices in $V \backslash V'$ receive a NAE-assignment. Ok, so we have grouped the $2^{|}E|$ terms in the Holant sum into $n + 1$ groups. But, have we made any progress? Look, every $x_j = 0$ unless $j = n$. That is, the Holant is exactly x_n. We do not know how to just compute x_n. However, we *do* know how to compute x_n and all the other x_j's after replacing all the signatures $[1, 0, 0, 1]$ with $[0, 1, 1, 0]$ by creating a Vandermonde system with the x_j's as the unknowns and an oracle for signature grids with

(only) $[0, 1, 1, 0]$.[2]

The interpolation construction is given in Figure 9 of [2]. Every N_i is a symmetric. This is clear from the picture (by rotational and horizontal symmetry). Then by induction, one can show that the signature is of the form $[a_i, b_i, b_i, a_i]$. So, for x_j, there are there are x_j values with a_j and $n-x_j$ values with b_j. This allows us to express the Holant on the signature grid Ω_j, which uses N_j, as

$$\text{Holant}(\Omega_j) = \sum_{j=0}^{n} x_j a_j^j b_j^{n-j} = \sum_{j=0}^{n} x_j b_j^n (a_j/b_j)^j.$$

Using our oracle, we can determine the left-hand side, so we have a system of linear equations. All that remains is to show that the matrix coefficient matrix formed by the (a_j/b_j)'s is a non-singular matrix.

The coefficient matrix is a Vandermonde matrix

$$\begin{bmatrix} (a_1/b_1)^0 & (a_1/b_1)^1 & \cdots & (a_1/b_1)^n \\ (a_2/b_2)^0 & (a_2/b_2)^1 & \cdots & (a_2/b_2)^n \\ \vdots & \vdots & \ddots & \vdots \\ (a_n/b_n)^0 & (a_n/b_n)^1 & \cdots & (a_n/b_n)^n \end{bmatrix},$$

which is nonsingular exactly when the (a_i/b_i)'s are distinct. Going back to the construction of the N_i's, we can express the signature for N_{i+1} as a linear combination of the entries in N_i. Thus, there exists a matrix A such that

$$\begin{bmatrix} a_{i+1} \\ b_{i+1} \end{bmatrix} = A^{i+i} \begin{bmatrix} a_0 \\ b_0 \end{bmatrix}.$$

We are almost finished with the proof. What remains is to verify that three conditions hold for A, and we do this in the next lecture. □

References

[1] Jin-Yi Cai and Vinay Choudhary. Valiants holant theorem and matchgate tensors. *Theoretical Computer Science*, 384(1):22 – 32, 2007.

[2] Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Holographic algorithms by fibonacci gates. *Linear Algebra and its Applications*, 2011.

[2]Note that unlike NP-hardness, #P-hardness allows for oracle reductions.