Geometric Complexity and Applications CIS 6930 August 26 - September 4, 2008

Lecture 1 through 4

Lecturer: Dr. Meera Sitharam Scribe: Venkatakrishnan Ramaswamy

1 Introduction

Geometric Complexity is a term that is used in a wide variety of contexts to mean different things. We will use this term to broadly refer to the study of computational complexity using different geometric techniques.

Geometry, historically was a study of measurement. *Analytic Geometry* is the study of geometry using the principles of algebra. We have studied Analytic Geometry in middle/high school and it enables us to answer questions such as, where does the tangent to a curve at some point intersect a certain other line.

Algebraic Geometry is the more modern avatar, in which we study zeros of polynomials defined over rings.

Example 1 *Consider the polynomials $x^2 + y^2 + z^2 - r^2$ and $ax + by + cz + d$. The ideal generated by these polynomials is the set of points on which both these polynomials go to zero. In this case, geometrically, it is the points of intersection of the sphere $x^2 + y^2 + z^2 - r^2 = 0$ with the plane $ax + by + cz + d = 0$*

While algebraic geometry deals with polynomial equations, *Semi-Algebraic Geometry* deals with polynomial inequalities. This involves tools from Real Analysis, Functional analysis and measure theory.

2 Klein's Erlangen program

Felix Klein with his Erlangen program in 1872 showed the link between Geometry and Abstract Algebra. This program looked at geometry as the study of properties of a space invariant under a given group of transformations. Thus geometric properties could be described by the group of transformations, under the action of which the properties would remain invariant.

Example 2 *Euclidean distance between two points is a property that is invariant under the action of the Euclidean group (which is the group of rigid body motions in Euclidean space). So in $\mathbb{R}^2$, for example, the distance between two points (x_1, y_1) and (x_2, y_2) is given by $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. Now suppose we perform the translations given by:*
$x' = x + a$
$y' = y + b$,

the new distance is $\sqrt{(x'_2 - x'_1)^2 + (y'_2 - y'_1)^2}$
$= \sqrt{(x_2 + a - x_1 - a)^2 + (y_2 + b - y_1 - b)^2}$
$=\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$,

which is the same as the distance between the pair of points before performing the co-ordinate transformations.

Angles, for example, are preserved, under rigid body motions and scaling.

2.1 Geometric Theorem Proving

In the Erlangen program, the hypothesis and the conclusions of a geometric theorem can be written down as polynomials, and a proof of the theorem would be equivalent to showing that if the polynomials corresponding to the hypothesis evaluate to zero, then the polynomial(s) corresponding to the conclusion evaluate to zero. This is the problem of ideal membership for polynomials in a ring of invariants for a geometric group. In general, determining ideal membership for polynomials turns out to be a hard problem (triple-exponential?). However, in this case, we also have the property that the polynomials are invariant under the action of certain groups. The Wu-Ritt decomposition algorithm gives an efficient way to determine ideal membership in this case.

3 Hilbert's Nullstellensatz

Hilbert's Nullstellensatz (German for "theorem of zeros") is a theorem in algebraic geometry that is useful in coming up with algorithms for determining ideal membership for polynomials.

Theorem 1 $P_t \in \mathcal{I}(P_1, \ldots, P_k)$ *if and only if there exist an integer* m *and multiplying polynomials* $Q_1, Q_2, \ldots, Q_k$ *such that* $P_t^m = \sum_{i=1}^{k} Q_i P_i$.

Now given a polynomial P_t, it is straightforward to verify if it is in the ideal generated by $P_1, \ldots, P_k$: Suppose we knew the degrees of the multiplying polynomials $Q_1, Q_2, \ldots, Q_k$, then constructing the multiplying polynomials would be sufficient to show membership of P_t in the ideal. To construct the multiplying polynomials, we need to determine coefficients of each of the terms for each multiplying polynomial. Taking these as variables, simplifying $\sum_{i=1}^{k} Q_i P_i$ to group like terms and equating coefficients of corresponding terms on the right and left hand sides, we observe that we need to solve a system of linear equations, in order to find the coefficients of the multiplying polynomials, which is an easy problem. However, in practice, the coefficients of the multiplying polynomials are not known. Brownawell [1] derived upper bounds on the degrees of $Q_1, \ldots, Q_k$ in terms of the degrees of $P_1, \ldots, P_k$. This immediately gives us an algorithm for checking ideal membership. For each possible degree (up to the Brownawell bound), we get a system of linear equations. If atleast one of these systems is consistent, then P_t lies in the ideal generated by $P_1, \ldots, P_k$, otherwise it does not. We illustrate this with an example:

Example 3 *Let*

$P_1 : x^2 + x + 1$
$P_2 : 2x + 3$
and

$P_t : 5x^2 + 11x + 7.$

We wish to determine if P_t is in the ideal generated by P_1 and P_2. Further, let us assume that the multiplying polynomials have degree atmost 1. Thus let,

$Q_1 : a_1x + b_1$
$Q_2 : a_2x + b_2.$

Now from Hilbert's Nullstellensatz, P_t is in the required ideal iff,
$P_t = Q_1P_1 + Q_2P_2$
$\Leftrightarrow 5x^2 + 11x + 7 = (a_1x + b_1)(x^2 + x + 1) + (a_2x + b_2)(2x + 3)$
$\Leftrightarrow 5x^2 + 11x + 7 = a_1x^3 + (a_1 + b_1 + 2a_2)x^2 + (a_1 + b_1 + 3a_2 + 2b_2)x + (b_1 + 3b_2)$

Equating coefficients on the left and right-hand-side, we get the following system of linear equations:

$a_1 = 0$
$a_1 + b_1 + 2a_2 = 5$
$a_1 + b_1 + 3a_2 + 2b_2 = 11$
$b_1 + 3b_2 = 7.$

This system turns out to have the unique solution: $a_1 = 0, b_1 = 1, a_2 = 2, b_2 = 2$. Thus P_t lies in the ideal generated by P_1 and P_2.

3.1 Checking if a set of polynomials does not have a common zero

It turns out that this problem can also be easily posed as an ideal membership problem.

Proposition 2 *A set of polynomials do not have a common zero if and only if the constant polynomial* 1 *is in the ideal generated by the set.*

4 Reducing Unsatisfiability to Ideal Membership

It turns out that several other problems can be reduced to ideal membership for polynomials. Here is one:

Problem 1 (Unsatisfiability) *Given a propositional formula, determine if no truth assignment to its variables makes the formula satisfiable.*

Unsatisfiability is known to be in co-$\mathcal{NP}$.

To reduce unsatisfiability to ideal membership,for each propositional variable, we construct a polynomial
$P_i : x_i^2 - x_i.$

From the propositional formula, we construct the polynomial P_t by replacing each conjunction with multiplication, each disjunction with addition, and replace each negation $\neg x_i$ with $(1 - x_i)$. The formula is unsatisfiable iff P_t, is in the ideal generated by $P_1, P_2, \ldots, P_n$. We can verify if this is so by using Hilbert's Nullstellensatz along with the Brownawell bounds to generate a sequence of linear systems of equations and check if any of them is consistent. We illustrate this reduction with an example:

Example 4 *Consider the 3-SAT formula* $(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \vee x_4)$.

As described before we have four polynomials $P_1, \ldots, P_4$, *with*
$P_i : x_i^2 - x_i$.
This polynomial has two roots 0 *and* 1, 0 *corresponding to* F *and* 1 *corresponding to* T, *and the polynomial* P_t *is constructed such that* P_t *evaluates to zero for a particular assignment of 0s and 1s to the* x_i*s iff the propositional formula evaluates to false for the corresponding truth assignment to the propositional variables.*

Thus, the polynomial P_t *corresponding to the above propositional formula is:*
$P_t : (x_1 + x_2 + (1 - x_3)).((1 - x_2) + x_3 + x_4)$

The idea is that a literal evaluates to False iff the corresponding expression evaluates to zero. Thus a disjunction of literals evaluates to false, iff in the polynomial, each term in the sum is zero, which makes the sum equal to zero. Similarly, a conjunction is false iff atleast one of the operands is false, and equivalently, the product of expressions evaluates to zero, iff atleast one of them is zero.

From the perspective of complexity theory, this reduction enables us to transfer upper bounds for Unsatisfiability from the problem of ideal membership and transfer lower bounds to ideal membership from Unsatisfiability.

5 Complexity Themes

The study of complexity usually has two different kinds of fields. One is the study of complexity measures related to computation and information. The other is the study of what are called complex systems.

5.1 Complexity measures related to computation and information

Computational Complexity encompasses the following broad themes:

1. **P vs. NP**: Some problems don't have fast algorithms (unless P=NP).
2. **NP vs. co-NP**: Some problems don't have short proofs.
3. **P vs. BPP**: Some problems cannot be derandomized.

This can be contrasted with the usual mathematical notions of complexity:

1. Dimension of representation.
2. Sparseness of representation.

5.1.1 Mulmuley's program

Ketan Mulmuley's program [2] to prove P$\neq$NP involves proving lower bounds on some algebraic complexity measure, which would imply P$\neq$NP.

5.1.2 An example of a problem for P vs. NP lower bounds

Problem 2 *Consider n points in $\mathbb{R}^d$. For $d = 2$,*
maximize $|S|$
such that
$\exists$ a set S of points
$\exists$ a set L of lines
such that any pair of points in S is separated by exactly half the lines in L.

5.1.3 Automated Geometry Theorem Proving

The problem of automated geometry theorem proving may be posed in Euclidean Geometry, Projective Geometry or Similarity Geometry. This is an example of a problem in the theme of NP vs. co-NP upper bounds. [3] is a good starting point for analytic approaches to this problem and [4] is a good reference for combinatorial and algebraic approaches.

5.2 Complex systems

Complex systems is a field that consists of problems from disparate fields. These problems have some common characteristics such as being *indecomposable* and having the property of *emergence*. They include problems from

1. Markets
2. Biological processes such as evolution
3. Weather
4. Dynamical systems
5. Iterated function systems
6. Games
7. Distributed systems

6 Sparse Representations

Here is an example of a sparseness question related to analysis:

Problem 3 (Interpolation) *Given certain points in a function f, pick a small number of basis function from a family of bases, and show that for a small number of bases (defined appropriately), the function cannot be exactly represented by interpolation when*

- *Each basis function is independent.*
- *When the basis functions are not independent.*

The field of *Wavelet Analysis* in particular, and in general *Nonlinear Approximation Theory* are interested in such questions.

The field of *Geometric Function Theory* (starting with Dvoretzky's Theorem) is interested in the following type of problem:

Problem 4 (Low-dimension and low-distortion embeddings) *Given a high dimensional set of points, does there exist a projection onto lower dimensions, which distorts it to a small extent?*

This type of problem is also tackled by Principal Components Analysis in Statistics and Machine Learning.

The Duality Theorem in Geometric Functional Analysis allows us to make non-existence results imply existence results.

7 Algebraic and Analytic approaches to complexity

We will consider analytic and algebraic approaches to the study of complexity and look at the similarities between the two approaches.

7.1 Analytic approach

We study boolean functions, which can be written using as few basis functions as possible,
$f = \sum_{b \in M \subseteq B} a_b b$

That is, we use as few basis functions as possible in M. B is a set of simple basis functions

If we cannot exactly represent the function with the basis, we try to approximate it:
$||f - \sum_{b \in M \subseteq B} a_b b|| < \epsilon$

7.2 Algebraic approaches

In the algebraic geometry approach, we treat sets as the zero set of a system of polynomial equations of low complexity. We might quantify low complexity by specifying bounds on number of equations, degree, number of variables or number of terms. We can also use semi-algebraic sets or a union of semi-algebraic sets.

The strength of the algebraic approach for complexity theory is that we can ask the the *input* $\in$ *set* and the *set* $\in$ *complexity class* questions in the same way.

Here is one way to algebrize:
Elements: linear programs (lps), geometric constraint systems (gcs) (systems of polynomial equalities and inequalities $g(x)$ which are somehow special; for example belong to an invariant ring of a geometric group or something else)
Sets: Set of lps/gcss that have a common zero/do not have a common zero.

Find a system of polynomials S (whose variables are coefficients of gcs g)
s.t. $S(g) = 0$ iff $\exists x \in F$ s.t.$g(x) = 0$
(F is some field, say the reals)

Note that this is a family of systems S_m, since their number of variables (coefficients of g) is unbounded since input set of g's is infinite.

Now to show g is not in the set, we need to show g do not have a common zero, which can be converted to an existential statement via Hilberts nullstellensatz, so then we can again get a system of polynomials
$\bar{S}$ s.t.
$\bar{S}(g) = 0$ iff there is no x s.t. $g(x) = 0$.

Complexity class: set of polynomial system-families S_m for which there is a "few, sparse, lowdegree" set of polynomials P_m
s.t. $S_m(g) = 0$ iff $P_m(g) = 0$ over a field F
i.e. $S =_F P$

Find a family of polynomial systems C_{S_m}
s.t.
$\exists P$ s.t. $C_S(P) = 0$ iff $\exists P$ (few/sparse/low-degree)
s.t. $S =_F P$

To show a lower bound, show C_S do not have a common zero. Which we can try to convert into an existential statement and then use the trick from earlier to find a polynomial system
B s.t.
$B(S) = 0$ iff C_S does not have a common zero iff S is not in the complexity class.

Algebrization is a hurdle to proving complexity lower bounds. See [5] for more.

References

[1] W.D. Brownawell, *Bounds for the Degrees in the Nullstellensatz*, Annals of Mathematics, **126** (1987), pp. 571-591.

[2] K. Mulmuley, M. Sohoni, *Geometric complexity theory I: An approach to the P vs. NP and related problems*, SIAM J. Comput., 31(2), pp. 496-526, (2001).

[3] I. J. Schoenberg, *Metric Spaces and Positive Definite Functions*, Transactions of the American Mathematical Society, Vol. 44, No. 3 (Nov., 1938), pp. 522-536.

[4] L.M. Blumenthal, *Theory and Applications of Distance Geometry*, Oxford, 1953.

[5] S. Aaronson, A. Wigderson, *Algebrization: a new barrier in complexity theory*, Proc. STOC 2008.

[6] C.H. Papadimitriou, *Computational Complexity*, Addison Wesley, 1993.

[7] J. Bourgain, *Harmonic Analysis and Combinatorics: How Much May They Contribute to Each Other?*, in Mathematics: Frontiers and Perspectives, V. Arnold, M. Atiyah, P. Lax, B. Mazur, eds., AMS 2000.

[8] I. Laba, *The Kakeya problem, and connections to harmonic analysis*, At `http://www.math.ubc.ca/ ilaba/kakeya.html`.

[9] N.H. Katz, T. Tao, *BOUNDS ON ARITHMETIC PROJECTIONS, AND APPLICATIONS TO THE KAKEYA CONJECTURE*, Mathematical Research Letters 6, 625-630 (1999).

[10] A. Samorodnitsky, L. Trevisan. *Gowers Uniformity, Influence of Variables, and PCPs*, In Proc. of 38th STOC, ACM, 2006.

[11] S. Arora, B. Barak, *Complexity Theory: A Modern Approach*, Cambridge University Press, expected in 2009. (A draft is available at `http://www.cs.princeton.edu/theory/complexity/` .

[12] T. Tao, *The dichotomy between structure and randomness*, International Congress of Mathematicians presentation, At `http://www.math.ucla.edu/ tao/preprints/Slides/icmslides2.pdf`

[13] S. Khot, A. Naor, *Nonembeddability theorems via Fourier analysis*, Mathematische Annalen 334, number 4, 821-852 (2006).

[14] `http://in-theory.blogspot.com/2006/06/gowers-uniformity.html`

[15] `http://in-theory.blogspot.com/2006/06/analytical-approaches-to-szemeredis_08.html`

[16] `http://lucatrevisan.wordpress.com/tag/additive-combinatorics/`

[17] `http://boolean-analysis.blogspot.com/2007/02/whats-new-in-fourier-analysis.html`

[18] A. A. Razborov, A. A. Sherstov, *The sign-rank of AC^0*, FOCS 2008.

[19] A. A. Sherstov, *The unbounded-error communication complexity of symmetric functions*, FOCS 2008.

[20] A. A. Sherstov, *Communication lower bounds using dual polynomials.*, Bulletin of the EATCS, 95:59-93, 2008. (Invited survey).

[21] P. Indyk, J. Matousek, *Low-distortion embeddings of finite metric spaces*, Handbook of Discrete and Computational Geometry.

Recent advances in Complexity CIS 6930/CIS 4930 October 14, 2008

Lecture Lecture 15

Lecturer: Jialong Cheng Scribe: Jialong Cheng

1 Introduction

A fundamental problem in distance geometry is to determine when the distances between certain pairs of vertices of a finite configuration in Euclidean space $\mathbb{E}^d$ determine it up to congruence.

From [1], a *configuration* is a finite collection of n labeled points, $\mathbf{p} = (p_1, \ldots, p_n)$, where each $p_i \in \mathbb{E}^d$, for $1 \leq i \leq n$. A graph G will always be finite, undirected, and without loops or multiple edges. A *bar framework* in $\mathbb{E}^d$ is a graph G with n vertices together with a corresponding configuration $\mathbf{p} = (p_1, \ldots, p_n)$ in $\mathbb{E}^d$, and is denoted by $G(\mathbf{p})$.

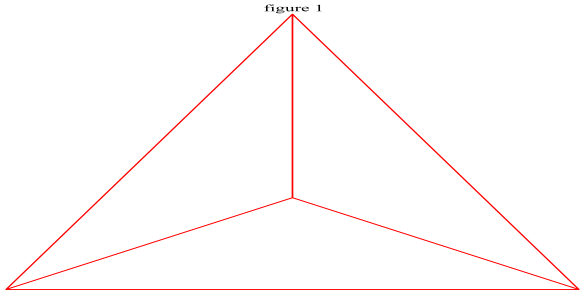
Note that a configuration can also be viewed as a map from the set of vertices V to a d-dimension Euclidean space $\mathbb{E}^d$.

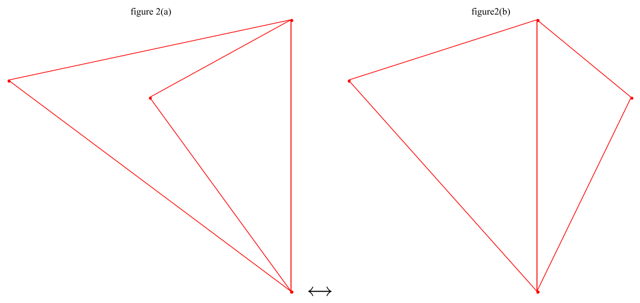
1.1 Global rigidity

We say that two frameworks $G(\mathbf{p})$ and $G(\mathbf{q})$ are *equivalent*, written as $G(\mathbf{p}) \equiv G(\mathbf{q})$, if when $\{i, j\}$ forms an edges of G, then $|p_i - p_j| = |q_i - q_j|$. We say that a configuration $\mathbf{p} = (p_1, \ldots, p_n)$ is *congruent* to $\mathbf{q} = (q_1, \ldots, q_n)$, denoted by $\mathbf{p} \equiv \mathbf{q}$, if for all $\{i, j\}$ in $\{1, \ldots, n\}$, $|p_i - p_j| = |q_i - q_j|$. A framework $G(\mathbf{p})$ is called *globally rigid* in $\mathbb{E}^d$ if $G(\mathbf{p}) \equiv G(\mathbf{q})$ implies $\mathbf{p} \equiv \mathbf{q}$. The congruence class of P in R^{3V} is denoted by [[p]]. The isometry class of P in R^{3V} is denoted by [p]. Note that equivalent is also called isometric.

Here I will stop to give several examples of global rigidity(figures).

figure 1





Note that the term **globally rigid** is a new version of the term **uniquely realized**.

A framework $G(\mathbf{p})$ in $\mathbb{E}^d$ is said to be *rigid* if there is an $\epsilon > 0$ such that for any other configuration $\mathbf{q}$ in $\mathbb{E}^d$, where $|\mathbf{p} - \mathbf{q}| < \epsilon$ and $G(\mathbf{p}) \equiv G(\mathbf{q})$, then $\mathbf{p} \equiv \mathbf{q}$.

1.2 Generic global rigidity

A set $A = (\alpha_1, \ldots, \alpha_m)$ of distinct real numbers is said to be *algebraically dependent* if there is a non-zero polynomial $h(x_1, \ldots, x_m)$ with integer coefficients such that $h(\alpha_1, \ldots, \alpha_m) = 0$. If a is not algebraically dependent, then it is called *generic*. If a configuration $\mathbf{p} = (p_1, \ldots, p_n)$ such that its dn coordinates are generic, we say $\mathbf{p}$ is generic.

Informally, a constraint graph is said to be *generically rigid* if it is rigid(does not flex or has only finitely many non-congruent, isolated solutions) for most of choices of coefficients of the system.

Formally, a property is said to hold *generically* for polynomials $f_1, \ldots, f_n$ if there is a nonzero polynomial P in the coefficients of f_i such that this property holds for all $f_1, \ldots, f_n$ for which P does not vanish.

Thus the system E is generically rigid if there is a nonzero polynomial P in the coefficients of equations of E–or the parameters of the constraint system–such that E is solvable when P does not vanish.

For a given graph G, when $G(\mathbf{p})$ is globally rigid for all generic configurations $\mathbf{p}$ in $\mathbb{E}^d$ we say that G itself is *generically globally rigid*.

References

[1] Robert Connelly, Generic global rigidity, Discrete Comput. Geom 33 (2005) no. 4, 549-563

[2] Bruce Hendrickson, Conditions for unique graph realizations, SIAM J. Comput. 21(1992), No. 1, 65-84

[3] Herman Gluck, Almost all simply connected closed surfaces are rigid, Geometric topology, pp. 225-239, Lecture Notes in Math., Vol. 438, Springer, Berlin, 1975. MR 53 #4074

[4] Steven J. Gortler, Alexander D. Healy, Dylan P. Thurston. Characterizing Generic Global Rigidity. arXiv:0710.0926v3, 2008

Recent advances in Complexity CIS 6930/CIS 4930 October 16, 2008

Lecture Lecture 16

Lecturer: Jialong Cheng Scribe: Jialong Cheng

1 Infinitesimal rigidity[1]

Before we go to Hendrickson's conditions for generic global rigidity, we first prove a theorem that infinitesimal rigidity implies rigidity.

Definition 1.1 1. *A family $(\delta_1, \ldots, \delta_v)$ of vectors in R^3 will be called an **infinitesimal isometric perturbation** of the polyhedron $P : K \to R^3$ if*

$$(p_i - p_j) \cdot (\delta_i - \delta_j) = 0, \textit{for all } (i, j) \in E$$

where the dot indicates the inner product in R^3.

2. *Suppose that t and r are any two vectors in R^3, and let $\delta_i = t + (p_i \times r)$ for $1 \leq i \leq V$. Then*

$$(p_i - p_j) \cdot (\delta_i - \delta_j) = (p_i - p_j) \cdot ((p_i - p_j) \times r) = 0,$$

*for all i, j, whether or not corresponding to an edge of K. Such a choice of $(\delta_1, \ldots, \delta_v)$ is therefore certainly an infinitesimal isometric perturbation, and is called an **infinitesimal congruence**.*

3. *An infinitesimal isometric perturbation which is not an infinitesimal congruence is called an **infinitesimal flexing**. If P admits an infinitesimal flexing, it is **infinitesimally flexible**; otherwise it is **infinitesimally rigid**.*

From this definition, we know that the only perturbation allowed in the direction of $p_i - p_j$ is a translation.

Homework:

- Try to find a framework that is rigid but not infinitesimally rigid.

To be continued...

References

[1] Bruce Hendrickson, Conditions for unique graph realizations, SIAM J. Comput. 21(1992), No. 1, 65-84

Geometric Complexity and Applications CIS 6930	September 23/08
Lecturer: Ugandhar & Venkat	Scribe: Ugandhar

Introduction

Polynomial algorithms are not known for any of the NP-hard problems. These problems (NP-hard) are not uncommon; they are pretty much frequent in day to day applications. As we were not able to find polynomial algorithms for these problems, it would be quite useful if one can provide a solution close to optimum with constant error factor but runs in polynomial time. In the following sections we discuss €-approximation algorithms for some NP-hard problems. And finally conclude with an example that has a constant approximation only when P = NP.

Threshold € : M is called € approximation algorithm (0<=€<=1) if for all x |C(M(x)) – OPT(x)|/max{OPT(x),C(M(x))} = €, where C(M(x)) is the approximation solution for x and OPT(x) is the optimum solution for x.

For Maximization: C(M(x)) >= (1-€) OPT(x) For minimization: C(M(x) <= 1/(1-€) OPT(x)

Approximation threshold for a problem A is the greatest lower bound of € such that A has an € approximation algorithm.

Vertex Cover: ***A vertex cover for an undirected graph G = (V,E) is a subset S vertices such that each edge in graph G has at least one endpoint as vertex in S.***

Algorithm: Start with an empty set of edges, S. Add an edge to set S (remove from graph G) such that it is not adjacent to a any edge in the set S. keep picking edges from G until there are no edges that are not adjacent to any of the edges in the set S. Vertices in the set S (end pts of edges in set S) cover all the edges in the initial graph G. These vertices cover the whole graph because all the remaining edges are adjacent to these edges (termination condition) which mean they share a vertex. This runs in polynomial time as it takes polynomial time to check weather an edge is adjacent to a set of edges or not.

Optimum vertex cover should cover all the edges in the set S. All the edges in the set S are disjoint. Optimum vertex cover should contain at least one vertex from each edge in S to cover all the edges in S. So we can conclude that 2*opt_vertex_cover >= vertex cover . € = 1/2

Max-Cut: ***Partition an undirected graph G into two sets of vertices such that the number of edges between these two sets is max.***

Divide the set of vertices arbitrarily into two set of vertices. In every step scan through all the vertices in the graph and add/ delete a vertex to S1, if (adding: moving from s2 to s1; deleting: moving from s1 to s2)) adding/deleting from S1 increases the number of edges between S1 and S2. Repeat the above step until there is no vertex to add /delete such that edge count increases. Max cut cuts through max number of edges in a graph. This algorithm runs in polynomial time because at every step we increment the number of edges passing through between sets by 1. At max there can be |E| number of edges between S1 and S2 |E| <= n^2.

Consider a Decomposition of V into four disjoint subsets V = V1, V2, V3, V4, such that the optimum partition obtained by our approximation algorithm is (V1 U V2, V3 U V4), where as the optimum partition is (V1 U V3, V2 U V4)[in case optimum is other combinations we can prove similarly]. Let eij, 1<= i <= j <= 4 be the number of edges between node sets Vi and Vj. For each vertex in V1 edges to V1 and V2 are outnumbered by those to V3 and V4(termination condition guarantees this). Consider all the nodes in V1 we obtain 2 e11 + e12 <= e13 + e14 which is equal to e12 <= e13 + e14. Similarly we obtain the following e12 <= e23+ e24, e34 <= e23 + e13, e34 <= e14 + e24. By adding all these inequalities and dividing by 2 we obtain e12 + e34 <= e14 + e23 + e13 + e24. Add the following inequality to above inequality: e14 + e23 <= e14 + e23 + e13 + e24. The result is e12 + e14 + e23 + e34 <= 2*(e14 + e23 + e13 + e24). € = ½

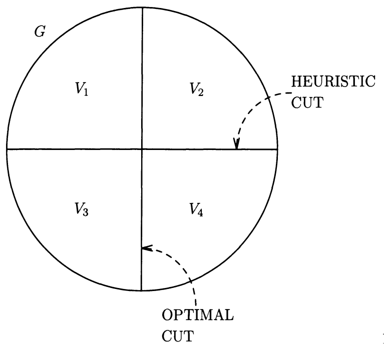
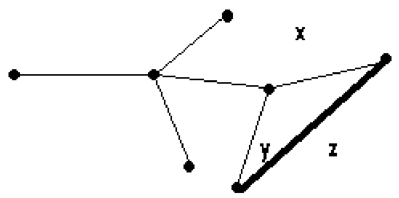


fig from [1]

TSP (Euclidian): Given an undirected complete Graph G, find an ordering of vertices s.t the length of the route traversed is min. (The weight on the edges satisfies Triangular property)

Algorithm: Find the MST of the given graph. Traverse from the start vertex along the MST, in some cases to reach an another vertex v traversing through an MST edge leads to repeating certain vertex, then take a non-MST vertex to visit v and continue in the MST and again take a non-MST path if again the above case arises. Non-MST edge we take in the above procedure forms a triangle or quadrilateral with MST edges, sum of two MST is greater than the non-MST edge (Triangle inequality) / sum of three MST is greater than one non-MST edge (Triangle inequality). As the graph is complete we can always find a non-MST edge. If we replace all the non-MST edges in our route with the sum of the MST edge with which it formed a triangle or the sum of three edges with which it forms a quadrilateral, then we obtain each weight of the MST edge twice. It can be written as 2*MST-Weight. As the non-MST weight is less than the two MST-weights/ three MST-weights. The weight of the path found by our algorithm is less than equal to 2*MST-Weight. Any TSP path cannot be better than an MST (TSP embeds an MST). So we can conclude 2*optimum-Path>= Path. [€ = ½]



(Let x, y be MST edge and z be the non-MST edge. Triangular inequality x+y >= z)

KnapSack: Given n weights wi i= 1..n and values vi i=1..n find subset S <= {1,2,,,,n} such that [sum(wi), i belongs to S]<= W) and [sum(vi), i belongs to S] is the largest value.

Let V = max {v1,,,vn} be the max value and let us define for each i= 0,1…n and 0<=v<= nV the quantity W(i,v) to be the minimum weight attainable by selecting some among the i first items, so their value is exactly v. Start with W(0,v)= Infinity for all i and v

$$W(i+1,v) = \min\{W(i,v), W(i, v - v_{i+1}) + w_{i+1}\}.$$ [1]

The above recursive equation is the result of Dynamic Programming solution to the above problem. It can be treated as a table containing rows 1 to n and columns from 1 to nV. Row i contains min possible weights for all the values (1 to nV) vi's can be from any of 0 to i. To compute all the values in the table it will take O((n^2) V) where V is a more than polynomial in the order of input n. It is psudopolynomial. We are interested in approximation algorithm so we can discard the last few bits of values thus gaining speed by trading accuracy. Making all the b bits zero in every vi creates new value v`i = vi - 2^b. As all the b trailing bits are zero we can right shift by b bits and run the above algorithm on these new v`i values. Now the same algo runs in O ((n^2) V/2^b) to find the optimum set S`. The solution obtained S` is different from S. The following inequalities bring about the relationship b/t them

$$\sum_{i\in S} v_i \geq \sum_{i\in S'} v_i \geq \sum_{i\in S'} v_i' \geq \sum_{i\in S} v_i' \geq \sum_{i\in S}(v_i - 2^b) \geq \sum_{i\in S} v_i - n2^b$$ [1]

First inequality is correct because S is optimal for vi's. Second inequality is true as vi >= v`i. third inequality is true as S` is optimal for v`i . Fourth is by definition of approximation algorithm. Fifth the summation ranges from 1 to |S|. |S| <= n. so the result. The following inequality shows that the approximation algorithm result value is lower by n*2^b . The deviation from optimum is at most € = n*(2^b)/V . Given any € > 0 we can truncate last b bits b = log (€ V/n) and arrive at an €-approximation algorithm whose runtime is O(n^3 /€) , polynomial. The greatest lower bound of epsilons for this is 0. [€ = 0]

TSP (non-Euclidian): Same definition as the above except that triangular inequality doesn't apply to edge weights. This TSP has a constant factor polynomial approximation algorithm only if P = NP. We will prove this by reducing Hamiltonian circuit problem to k factor TSP approximation algorithm.

Hamiltonian circuit problem: {G: there exists an ordering of vertices of G v1…vn s.t (vi, v(i+1 mod n)) belongs to E(G) and v1,vn belongs to E(G)}

TSP = {(G,Val):G is complete graph , there exists an ordering of vertices of G v1,,,,vn Sum(vi,v(i+1 mod n)) <= Val; }

Corresponding optimization problem for TSP is min (Sum wt(vi,v(i+1 mod n))) = Tk [min TSP route].

A polynomial const factor approx algorithm for min TSP will find a TSP route T1 s.t wt(T1) <= k*wt(Tk)

We can reduce Hamiltonian circuit problem to finding a constant factor polynomial approximation to TSP by the following reduction.

Assign the edge weights 1 unit to all Hamiltonian Graph(G) edges. Add edges to this graph to make it a complete graph (G1). Let the edge weights for each of these new edges be k|v| (K is constant factor for TSP approx algo and v is the number of vertices in the graph). If k-constant factor TSP route algorithm comes up with an edge weight <= kv (inclusion of at least one non-edge of G makes it, v-1+kv=v(k+1)-1 which is greater than kv at least for v >=2) then we can claim that these edges are part of G and hence a Hamiltonian circuit exists. If greater than kv we can say no Hamiltonian circuit exists. But we already know that Hamiltonian circuit is NP-hard, which means a polynomial solution to this exists only if P = NP.

References

[1] C.H. Papadimitriou, Computational Complexity, Addison Wesley, 1993.

Geometric Complexity and Applications CIS 6930	Oct 2 – Oct 14, 2008
Lecturer: Ugandhar	Scribe: Ugandhar

Introduction: IP is class of problems solvable by interactive proof system. It is introduced by Goldwasser et al 1985. Interactive proof system is an abstract machine that models computation as the exchange of messages between two parties. One party is all powerful with unbounded resources called prover and the other has bounded computational power known as Verifier. NP Class can be modeled as Interactive proof system. Verifier for NP is deterministic polynomial-time machine. In this chapter we shall see that inclusion of randomness in a deterministic polynomial-time machine enhances its capacity. We will discuss some important Random classes that can be recognized by the probabilistic polynomial time machine. Finally we present introduction to Interactive proof systems which allows probabilistic verifiers, Zero knowledge proof system and conclude with IP = PSAPCE proof.

Randomized Polynomial Class:

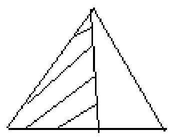
Every NP problem can be verified in Polynomial time. This is possible by checking the input instance with given certificate. NP problems can be thought of as having a binary tree structure. The path from root to leaf constitutes a certificate. It is an accepting certificate if an input instance x when verified on y it resulted in acceptance. This verification step runs in deterministic polynomial time. We can achieve a mapping between sequence of coin tosses and certificates.

Randomized Polynomial Class: x belongs to set X if at least ½ the certificates are accepting. x doesn't belong to X if all the certificates are rejecting.

x € X => there exists >= ½ y's |y| < |x|^k (x,y) € S {Pr(accep) >= ½}

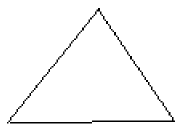
x dosen't belong X => for all y's |y|<= |x|^k (x,y) dosen't belong to S {Pr(reject) = 1}

x belongs to X



>=1/2 accepting paths

x doesn't belong to X



all are rejecting paths

Eg: Given a problem in straight line program format find if the co-efficients of the resulting monomial is not identically zero over rational.

Let straight line equation ax + by +c be the input. Let the program have the following series of steps p = input; p = p^2; p = p*z; …

We can have polynomial number of such steps. We would be interested in knowing if the final monomial is zero i.e all its co-efficients are zero.

If we try by expanding the polynomial in each step then the resulting polynomial would be in exponential in length. Every square doubles the length. So a polynomial number of squaring steps will easily make the resulting polynomial length exponential. So we cannot calculate by expanding at each step to find the co-efficients of monomial. Instead we can do the following, which runs in polynomial time.

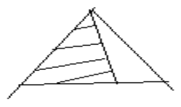
Randomly select rational numbers for each variable used in the polynomial. Substitute these values and evaluate the polynomial in every step to some number. Every step substation and evaluation takes poly time. There are polynomial number of such steps. We can compute the value in polynomial time. Suppose if the values of the variable selected are the roots of the polynomial then even if the co-efficient of the resulting polynomial is not zero our evaluation as described in the above results in zero. The probability of this happening for any random value chosen from rational is very small because for any polynomial the ratio of number of roots and the number system on which the polynomial is defined is negligible. So with probability >= ½ we can find if the monomial form is not identically zero. If the resulting monomial is identically zero then for all the inputs the multivariate polynomial returns zero. So for all the cases it rejects if monomial is identically zero. This algorithm has the structure of RP because of the following acceptance criteria. P is the monomial. V is the randomly chosen tuple of values for the variables.

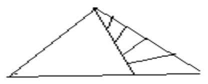
$P \neq 0$ Pr(P(V) != 0) >= ½

$P \equiv 0$ Pr(P(V) =0) = 1

Bounded probabilistic polynomial (BPP): x € X => >=1/2+ e of y's |y| € poly(|x|) (x,y) € S

x doesn't belong to X => >=1/2+ e of y's |y| € poly(|x|) (x,y) doesn't belong to S





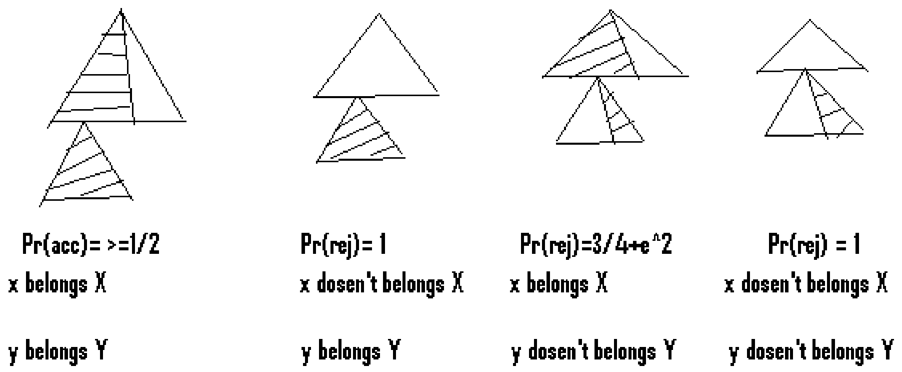
>=1/2 +e accepting path >= ½ +e rejecting path

BPP is closed under complementation as both the accepting and rejecting are symmetric.

Eg: (Px,Py) € P {(Px,Py):Px identically not zero and Py identically zero}

R.P is in Bpp as Pr(acceptance) is >=1/2 by boosting falls in >= ½+e, Pr(rejection) is 1 which falls in >=½ + e ; Co-RP is in BPP as Pr(acceptance) = 1 which falls in >=1/2 +e ; Pr(rejection) >= ½ by boosting falls in >= ½ +e .

X is R.P; Y is Co-RP, in order to prove that above example is BPP, I need to show that Px and Py is recognizable by BPP. Let Px be x in & Py be y in the below diagram. Run X for every accepting path Run Y and accept if both accept. Accept if both X and Y accepts. The below diagram proves that Bpp can recognize Px and Py. So Px and Py can be recognized by Bpp machine.



Zero knowledge proof System:

In cryptography, a **zero-knowledge proof** or **zero-knowledge protocol** is an interactive method for one party to prove to another that a (usually mathematical) statement is true, without revealing anything other than the veracity of the statement.

A zero-knowledge proof must satisfy three properties:

1. **Completeness**: if the statement is true, the honest verifier (that is, one following the protocol properly) will be convinced of this fact by an honest prover.
2. **Soundness**: if the statement is false, no cheating prover can convince the honest verifier that it is true, except with some small probability.
3. **Zero-knowledge**: if the statement is true, no cheating verifier learns anything other than this fact. This is formalized by showing that every cheating verifier has some *simulator* that, given only the statement to be proven (and no access to the prover), can produce a transcript that "looks like" an interaction between the honest prover and the cheating verifier.

   For more refer [3]

The below is an example of a problem in Co-NP that can be solved using Zero knowledge proof System.

Graph Non-Isomorphism: I <G1,G2> € G {(G1,G2): G1 is not isomorphic to G2} . This problem is in Co-NP because the complement of this problem is in NP. NP problem is Graph Isomorphism. We can check Graph Isomorphism by a certificate, in this case permutation that can convert one of the graphs to the other. So given a permutation (certificate) we can say if the set <G1,G2> is isomorphic .

Zero-knowledge proof protocol for Graph Non-Isomorphism:

The verifier randomly generates a number between 1,2 . Base on the random number x, it permutes Gx. The verifier sends P(Gx) to the prover. Verifier expects prover to send the random number it chose, to be convinced of Graph Non-Isomorphism. As prover is all powerful with unbounded computational resource, it should be able to find the random number. Prover compares P(Gx) with G1 and G2. If G1 and G2 are Non-isomorphic then prover can identify the random number x correctly. So if Graph is Non-Isomorphic then probability of acceptance is 1. If graphs are isomorphic then prover cannot uniquely identify the source of P(G(x)). So it randomly guesses number between 1,2 and sends the it to verifier. If graphs are isomorphic then with50-50% changes verifier will be convinced that they are isomorphic.

Interactive Proof systems: All interactive proof systems have two requirements:

- **Completeness**: if the statement is true, the honest verifier (that is, one following the protocol properly) will be convinced of this fact by an honest prover.
- **Soundness**: if the statement is false, no prover, even if it doesn't follow the protocol, can convince the honest verifier that it is true, except with some small probability.

Arthur-Merlin, Merlin-Arthur protocol and IP: AM is the first concept of computation through interaction. It was published by László Babai in "Trading group theory for randomness", a paper defining the *Arthur-Merlin* (AM) class hierarchy. Arthur-Merlin protocol is an interactive proof system in which the verifier's coin tosses are constrained to be public. Arthur (the verifier) is a probabilistic, polynomial-time machine, while Merlin (the prover) has unbounded resources The complexity class AM[k] is the set of problems that can be decided in polynomial time by an Arthur-Merlin protocol, with k queries and responses.

The complexity class **MA** is the set of decision problems that can be decided in polynomial time with (unlimited) communication only from Merlin to Arthur. In particular, **MA** is contained in the intersection of Σ_2P and Π_2P and **AM** is contained in Π_2P. Later it was proved by Goldwasser and Sipser that all languages with constant-length interactive proofs with private coins also have interactive proofs with public coins with two additional rounds. Private coins may not be helpful, but more turns are: if we allow the probabilistic verifier machine and the all-powerful prover to interact for a polynomial number of rounds, we can solve the class of problems **IP**. **(**For more refer [6], [7])

IP: We say (P, V) decides a language L if the following is true for each string in x:

if x belongs to L => Pr(accept) by (P,V) is at least 1- 1/(2^|x|)

if x dosen't belong to L => Pr(accept) by (P`,V) is at most 1/(2^|x|)

Proof: **IP = PSPACE**

a) PSPACE <= IP: We can prove PSPACE <= IP if we can provide an interactive protocol for PSPACE – complete problem QSAT. QSAT: Given a Boolean expression Φ in conjunctive normal form, with Boolean variables x1,,, xn is it true that for both true values for the variables x1 there is a truth value for the variable x2 such that for both the truth values for the variable x3 and so on up to xn, Φ is satisfied by the overall truth assignment?
$\exists x_1 \forall x_2 \exists x_3 \ldots Q_n x_n \quad \phi?$

Let $\phi = \forall x \exists y (x \vee y) \wedge \forall z ((x \wedge z) \vee (y \wedge \neg z)) \vee \exists w (z \vee (y \wedge \neg w)).$ be the Quantified Boolean expression Φ. We can observe that the above expression is not in prenex form. As we proceed, we would understand why the above form (simplex form) is important for our proof.

Claim: Any quantified Boolean formula Φ can be transformed into logarithmic space to an equivalent simple expression. Consider a universal quantifier $\forall y$ and a variable x quantified before $\forall y$ and used after $\forall y$. Φ is of the form $\cdots Qx \ldots \forall y \psi(x)$. We transform Φ as follows $\ldots Qx \ldots \forall y \exists x' ((x \wedge x') \vee (\neg x \wedge \neg x')) \wedge \psi(x').$ To not let x be used after the second for-all quantifier we restrict its usage between before the second for-all by defining a new variable and quantifying it after the first for-all. So new variable can replace x used after the second for-all. In this way we restrain the usage of a variable and its quantification to at most separation of one for all quantifier. We fix similarly at each for-all quantifiers. At each for-all quantifier there can O(n) fixes and the number of for-all quantifiers can be O(n) . So the above described procedure runs in O (n^2). In O (n^2) steps the above expression becomes simple.

Arithmetization: The protocol to proves QSAT satisfiability by converting QSAT expression into equivalent arithmetic expression and then convinces Verifier that the Arithmetized value is not zero. To arithmetize Φ we replace Boolean variables with integer variables , with the convention that x =0 means x is false, x =1 means x is true , $\neg x$ is translated to 1-x, v replaced with + and ^ is replaced with * . The result zero for the arithmetic expression means false and the result greater than 0 means true for the corresponding QSAT. By the above conversion procedure the above QSAT equation becomes
$A_\phi = \prod_{x=0}^{1} \sum_{y=0}^{1} [(x+y) \cdot \prod_{z=0}^{1} [(x \cdot z + y \cdot (1-z)) + \sum_{w=0}^{1} (z + y \cdot (1-w))]].$ [1]AΦ has no free variables.

$$\prod_{x_1=0}^{1}\prod_{x_2=0}^{1}\cdots\prod_{x_k=0}^{1}\sum_{y_1=0}^{1}\sum_{y_2=0}^{1}(y_1+y_2)$$ [1]

Prover can calculate the integer value for the above arithmetic expression and send the value to verifier. The value for the above expression is of the order 4^(2^k). The value computed by prover can be of the order 2^(2^n). The binary representation of this would be exponential. If it is exponential then verifier cannot receive it as it has only polynomial space. To counter this problem prover can send the modulus of prime p, (2^n <= p <= 2^(3*n)). But have to make sure that we have set of prime's p so that for any positive value of AΦ, AΦ mod p will be greater than 0.If that is the case then we can send the prime number p and the AΦ mod p to the verifier to prove that AΦ is not zero.

Claim: If the value of expression AΦ, of length n, is non-zero then there is a prime p between 2^n and 2 ^ (3*n) such that AΦ ≠ 0 mod p.
The value of AΦ is no more than 2^(2^n) because each ∑ doubles the value and each ∏ squares the value. There are only n operations ,and at most the value can be no larger than 2^(2^n)[worst case all the n values are ∏, even then the value AΦ will be of order 2^(2^n].

Number of primes up to n is at least square root (n)

$$n\prod_{i=2}^{\lfloor\sqrt{n}\rfloor}\frac{i-1}{i}\geq\sqrt{n}.$$ [1]

By the above equation we can obtain the number of primes between 2^n and 2^(3*n) to be 2^n .

Let AΦ is divisible by all the primes p , p lies between 2^n <= p <= 2^(3*n). In that case the value of AΦ should be of the order of at least (2^n*(2^n)) which is more than the value AΦ can attain. Hence there exists p for every AΦ greater than zero s.t AΦ mod p is greater than zero.

$$A_\phi=\prod_{x=0}^{1}\sum_{v=0}^{1}[(x+y)\cdot\prod_{z=0}^{1}[(x\cdot z+y\cdot(1-z))+\sum_{w=0}^{1}(z+y\cdot(1-w))]].$$ [1]

Now the protocol for the above expression is as follows:

P $\xrightarrow{\text{(A}\Phi\text{ mod p , p) : (5,13)}}$ V a = 5 , p =13

$\xleftarrow[\text{V req P(x) in }\prod\text{ P(x)}]{}$

P $\xrightarrow{\text{Ax = } 2x^2+8x+6}$ V Ax(0).Ax(1) = a mod p ; a = Ax(9) mod p = 6 mod p ;

$\xleftarrow[\text{V req to delete }\prod\text{; V req P(y) in }\sum\text{P(y); x = 9}]{}$

$$\sum_{y=0}^{1}[(9+y)\cdot\prod_{z=0}^{1}[(9\cdot z+y\cdot(1-z))+\sum_{w=0}^{1}(z+y\cdot(1-w))]].$$ [1]

P $\xrightarrow{\text{Ay = } 2y^3+y^2+3y}$ V Ay(0) + Ay(1) = a mod p ; a = Ay(3) mod p = 7 mod p;
12.x = a mod p , x = 6 mod p, a = x;

$\xleftarrow[\text{V req to delete }\sum\text{; V req P (z) in }\prod\text{P (z); y = 3}]{}$

$$A=\prod_{z=0}^{1}[(9\cdot z+3\cdot(1-z))+\sum_{w=0}^{1}(z+3\cdot(1-w))]$$

P $\xrightarrow{\text{Az = } 8z+6}$ V Az(0)+Az(1) = a mod p; a = Az(7)mod p= 7 modp;
6+y = 10 mod p, y = 4mod p, a = y;

$\xleftarrow[\text{V req to delete }\prod\text{ ; V req P(w) in }\sum\text{P(w); z = 7}]{}$

$$A=\sum_{w=0}^{1}(7+3\cdot(1-w))$$

P $\xrightarrow{\text{Aw = } 10-3\cdot w}$ V Check if Aw (0) +Aw (1) is 4 mod p; verify if the poly after deleting ∑ is same as the polynomial sent by the prover. If both are yes then accept else reject.

The protocol starts when prover communicates the value of AΦ mod p, and p to the verifier. Verifier checks if the value is 0 or not and stores AΦ mod p as 'a' and prime as p. Then the verifier requests the p(x) in AΦ = ∏P(x). The prover computes this value p(x) by evaluating all the symbols except the last for ∏. This polynomial cannot be computed by V because the co-efficients can of the order of 2^(2^n) . But the length of the poly(x) will be of the order of x because the x quantification and usage is separated at most by one for all quantifier. So the max degree x can get is 2n which is polynomial. This is why we need to convert prenex to simplex form. Then the verifier calculates Ax (0).Ax (1) based on the Quantifier used. It compares this value with 'a' supplied earlier. If both are same then verifier requests to deleted the last quantifier in AΦ and substitute the free variables with a random value mod p generated by the verifier. In the above case that value is 9. Also verifier calculates new AΦ mod p by substituting 9 in Ax i.e a = Ax (9) mod p. In the next communication by prover it sends the poly(y) in ∑poly(y) of AΦ after deleting last ∏ and substitution 9 for x. Again verifier compares the value of Ay (0) + Ay (1) mod p with 'a'. Like this the protocol continues. In the last communication step by prover, prover sends the polynomial that is obtained by stripping all the quantifiers and substituting the free values by the values generated by V. This polynomial is compared with the polynomial V gets by stripping all the quantifiers and substitution its random values. If both are same and the poly equals the value 'a' computed in the earlier step then our machine accepts else it rejects. It is clear from the protocol that if AΦ is satisfiable IP accepts with Pr(1). If AΦ is equal to 0 then IP accepts with probability (1-2n/2^n)^n .

Claim: if AΦ = 0 and still the prover stats with a non-zero value then with probability (1-2n/2^n)^(i-1) the value claimed in the ith round is wrong

Base case: First value claimed by AΦ is non-zero, so definitely wrong with probability 1.

By induction we know i-1 step is wrong with probability (1- 2n/2^n)^(i-2) . Suppose it is wrong then the value produced by the ith round that is Ax(0) + Ax(1) / Ax(0).Ax(1) mod p should be that wrong value. Prover must supply a wrong polynomial to satisfy the wrong value. Let C(x) be the correct polynomial at each step (polynomial that would have been produced if proved started with 0). There is a possibility that the wrong value supplied and the correct polynomial evaluates to the same value on certain random values. If it evaluates to same values then prover can continue from this step till the end like our protocol with all the correct values and correct polynomials. This instance of wrong polynomial and correct polynomial evaluating to the same value can deceive the verifier. Both polynomials evaluate to same values can be expressed as C(x)-A(x)=0{C(x) is the correct polynomial; A(x) is the wrong polynomial}. Roots of this polynomial are the random values at which both have same values. The probability of this happening is 2n/2^n (random values mod p is between 0 to 2^n. degree of poly 2n gas at most 2n roots) . So if i-1 th value is wrong then probability that ith value is wrong is (1- 2n/2^n)^(i-1). **(**For more refer [1])

b) IP <= PSPACE

IP definition states: if x belongs to X there exists a prover P such that Pr(V ↔ P accepts) >= 1- 1/(2^|x|)
If x dosen't belong to X for all provers P` Pr(V ↔ P` rejects) >= 1- 1/(2^|x|)
To prove this we construct a PSPACE machine that simulates V. For any string w we define
$\Pr[V \text{ accepts } w] = \max_P \Pr[V \leftrightarrow P \text{ accepts } w]$. **(Eq 1)** i.e max of acceptance probabilities over all possible P's. Pr values should be atleast 1- 1/(2^|x|)if w is in A, at most 1/(2^|x|) if not. Let Mj denote the message history m1#m2#m3#..mj . To Generalize interaction of V and P , we start with an arbitrary message stream Mj . It can be written as $(V\leftrightarrow P)(w,r,M_j) = accept$ if Mj with messages m(j+1) through mp accepts so

$j \le i < p$, where i is even, $V(w,r,M_i) = m_{i+1}$ → Message sent by verifier

$j \le i < p$, where i is odd, $P(w,r,M_i) = m_{i+1}$ → Message sent by prover

Final message mp is accept/reject

Further generalizing, Probability of acceptance of w starting at Mj(configuration) given the polynomial length of random string :
$Pr[V \leftrightarrow P \text{ accepts } w \text{ starting at } M_j] = Pr[(V \leftrightarrow P)(w,r,M_j) = accept]$
Writing the above statement in terms of Eq 1:
$\Pr[V \text{ accepts } w \text{ starting at } M_j] = \max_P \Pr[V\leftrightarrow P \text{ accepts } w \text{ starting at } M_j]$.

Ffor every $0 \le j \le p$ and every message history M_j, we inductively define the function N_{M_j}:

$$N_{M_j} = \begin{cases} 0 : j = p \text{ and } m_p = reject \\ 1 : j = p \text{ and } m_p = accept \\ max_{m_{j+1}} N_{M_{j+1}} : j < p \text{ and } j \text{ is odd} \\ wt-avg_{m_{j+1}} N_{M_{j+1}} : j < p \text{ and } j \text{ is even} \end{cases}$$

N_{M_j} : Probability that V accepts w starting at Mj. It is defined recursively. Base case (N_{M_p}) is either 1 or 0 as at end of rounds of interaction w is either accepted or rejected. Let M_j be the configuration at prover. $P(w,r,M_j) = m_{j+1}$, prover sends a message m_{j+1} to the verifier such that it maximizes number of outcomes of the verifier coin tosses that are consistent with M_j and leads the verifier to

accept when subsequent prover moves are determined. Given the configuration Mj, m_{j+1} is chosen such that it generates max probability of acceptance in N_{Mj+1} among all the possible m_{j+1} that can be sent to the verifier.
Let Mj be the configuration of the Verifier. Probability that message m_{j+1} is sent to the prover is: weighted avg of probabilities of m_{j+1} across various random numbers. Probability of acceptance at Mj is probability of Acceptance of N_{Mj+1}*probability of message of m_{j+1} where the term $wt-avg_{m_{j+1}}N_{M_{j+1}}$is defined as follows: $wt-avg_{m_{j+1}}N_{M_{j+1}} = \sum_{m_{j+1}}(Pr_r[V(w,r,M_j)])$ i.e. probability that verifier sent message mj+1 which is equal to the summation of probabilities of all possible m(j+1). $\Pr_r$ is probability taken over all strings random strings 'r' that are consistent with Mj. If no such r exists then define the probability to be zero.

M_0 is empty message stream. We can calculate N_{M_0} in polynomial space by recursively calculating N_{M_j} for every j and Mj . Depth of the recursion is polynomial(interaction between P and V is polynomial) so only polynomial space is sufficient. By induction we can show $0 \leq j \leq p$and every M_j, $N_{M_j} = Pr[V\ accepts\ w\ starting\ at M_j]$, and we will do this using induction on j.

Base case $j = p$. m_p is either accept or reject, if m_p is accept, N_{M_p} is 1, so Pr[V accepts w starting at M_j] = 1 since the message stream indicates acceptance. If m_p is reject, N_{M_p} is 1, so Pr[V accepts w starting at M_j] = 0

Inductive hypothesis, we assume $N_{M_{j+1}} = Pr[V\ accepts\ w\ starting\ at\ j+1]$true for some $j+1 \leq p$ and some message sequence M_{j+1}

Let m_{j+1} is a message from V to P. By the definition of N_{M_j}, $N_{M_j} = \sum_{m_{j+1}}(Pr_r[V(w,r,M_j) = m_{j+1}]N_{M_{j+1}})$. Then, by the inductive hypothesis, we get $\sum_{m_{j+1}}(Pr_r[V(w,r,M_j) = m_{j+1}] * Pr[V\ accepts\ w\ starting\ at\ M_{j+1}])$. This is equal to $Pr[V\ accepts\ w\ starting\ at\ M_j]$.

Let m_{j+1} is a message from P to V. By definition, $N_{M_j} = max_{m_{j+1}}N_{M_{j+1}}$. by the inductive hypothesis, this equals $max_{m_{j+1}} * Pr[V\ accepts\ w\ starting\ at\ M_{j+1}]$.

$max_{m_{j+1}}Pr[V\ accepts\ w\ starting\ at\ M_{j+1}] \leq Pr[V\ accepts\ w\ starting\ at\ M_j]$ Prover always sends a msg to verifier such that N_{Mj+1} is maximized

$max_{m_{j+1}}Pr[V\ accepts\ w\ starting\ at\ M_{j+1}] \geq Pr[V\ accepts\ w\ starting\ at\ M_j]$ Prover cannot do any better than send that same message.

Thus, $max_{m_{j+1}} * Pr[V\ accepts\ w\ starting\ at\ M_{j+1}]$ = N_{Mj} . Hence **IP <=PSPACE**

In the above proof we constructed a polynomial space machine that uses the best prover *P* (max over P) for a particular string *w* in language *A*. We use this best prover in place of a prover with random input bits because we are able to try every set of random input bits in polynomial . Since we have simulated an interactive proof system with a polynomial space machine, we have shown that **IP <=PSPACE .** *Refer for more at [2],[4],[5]*

IP <= PSPACE is true and PSPACE <= IP is true => IP = PSPACE

References

[1] C.H. Papadimitriou, Computational Complexity, Addison Wesley, 1993.

[2] IP = PSPACE proof at http://en.wikipedia.org/wiki/IP_(complexity)

[3] Zero-Knowledge proof system at http://en.wikipedia.org/wiki/Zero-knowledge_proof

[4] Michael Fredric Sipser, Introduction to the Theory of Computation, Thomson

[5]Notes on Probabilistic proof systems at www.wisdom.weizmann.ac.il/~oded/CC/r5.ps by Oded Goldreich

[6] Introduction to AM at http://en.wikipedia.org/wiki/Arthur-Merlin_protocol

[7] Introduction to AM, MA, IP http://en.wikipedia.org/wiki/Interactive_proof_system

Geometric Complexity and Applications CIS 6930 September 9-26, 2008

Lecture on PCP theorem and approximability

Lecturer: Venkatakrishnan Ramaswamy Scribe: Venkatakrishnan Ramaswamy

1 Introduction

In the first part of these notes, we study *probabilistically checkable proof systems (PCPs)* and look at the celebrated PCP theorem which provides a probabilistic characterization of the class NP. We use the PCP theorem to prove some hardness of approximation results.

In the second part, we describe a type of reduction called *L-reductions*, which unlike the many-one polynomial-time reductions used in the theory of NP-completeness, preserve approximation ratios. We use these reductions in developing a complexity theory of approximability and show the existence of a large class of problems that do not have a PTAS, unless P=NP.

The material in the first part and the second part follows respectively the material in [1] and [2]. The reader will frequently be referred to these books for details that are not central to the exposition here.

2 The PCP Theorem

The PCP Theorem provides a powerful probabilistic characterization of the class NP. In this section, we will discuss probabilistic proof verifiers, state the PCP theorem, prove the easy direction of the inclusion, and present some intuition. We then use the PCP theorem to prove a hardness of approximation result for MAX k-FUNCTION SAT. This result will be used in the next section to show that no PTAS exist for a large class of problems, unless P=NP.

Recall that NP is usually defined to be the class of decision problems, whose "Yes" instances have short (polynomial in the length of the input) witnesses that can be verified in polynomial time. Informally, the PCP theorem states that for these problems, in fact, there exist witnesses which can be verified with high probability by only looking at a constant number of randomly chosen bits. The witness string is often called a proof in the literature. We will use these terms synonymously.

A *probabilistically checkable proof system* is described by two parameters namely the number of random bits it needs, and the number of bits of the proof it queries. The PCP theorem uses $O(\log n)$ number of random bits and a constant number of query bits.

The *verifier* for the PCP is a polynomial-time Turing machine, which apart from its input tape has a tape that provides random bits and another tape that has the proof on it. The proof tape is random access, i.e. any bit of the proof can be read in constant time. Clearly, which bits of the proof are queried is then a function of the input and the random bits. The turing machine either halts and accepts the proof or halts and rejects the proof.

Definition 1 *A language $L \in PCP(r(n), q(n))$ if there is a verifier V and constants c and k, such that on input x, V obtains a random string of length $cr(n)$ and queries $kq(n)$ bits of the proof. Further,*

- *If $x \in L$, then there is a proof y that makes V accept with probability 1,*
- *If $x \notin L$, then for every proof y, V accepts with probability $< \frac{1}{2}$,*

  where the probabilities are over the random string.

The probability of accepting in case $x \notin L$ is called the *error probability.*

Intuitively, a PCP verifier is analogous to a lazy but competent TA who is grading proofs written by students. Since the TA is lazy, she only checks a small part of each student's proof and based on the correctness of that small part decides if the proof is right or wrong. Since the TA is competent, if the entire proof is right she will never fail to be convinced by any part of it. However, if the proof is incorrect, it might be the case that she happened to look at a correct portion of the proof and erroneously concluded that the entire proof is right. The error probability quantifies the chance of this happening. This is the intuitive reason for the one-sided error in the definition above.

It is immediate from the definition that NP=PCP($0, poly(n)$). This is because a deterministic verifier requires zero random bits and queries atmost a polynomial number of bits of the proof and has error probability zero (which is smaller than 1/2). The PCP Theorem offers another characterization of NP.

Theorem 2 (PCP Theorem) *NP=PCP*$(\log n, 1)$.

The inclusion PCP$(\log n, 1) \subseteq$ NP is easy to prove. Consider a language $L \in$ PCP$(\log n, 1)$. It thus has a PCP verifier. The claim is that using this PCP verifier we can construct a deterministic verifier. Both these verifiers receive the input instance and a proof, and seek to verify the proof for the input instance. Additionally the PCP verifier uses $\log n$ random bits and gives answers that are true in probability over this random string. What we do is to simply run the PCP verifier over every possible "random" bit string and accept if and only if the PCP verifier accepts for every random string (while keeping the input instance and the proof the same). Since the random string is of length $\log n$ and the PCP verifier runs in polynomial time, the resulting deterministic verifier also runs in polynomial time.

The other inclusion NP $\subseteq$ PCP$(\log n, 1)$ is much harder to prove and will not be discussed in these notes. The interested reader is referred to [3] for details.

Next, we describe a weaker probabilistic verifier (than the PCP verifier) for 3SAT, which should give the reader a sense of how strong the PCP Theorem is. Consider a 3-CNF SAT boolean formula with m clauses. 3-SAT is in the class NP. Suppose we are given a certificate, which is an assignment to the variables and $O(\log n)$ bits. The weak verifier does the following: It randomly select one of the clauses. Since, the number of clauses is atmost the number of variables, $\log n$ random bits are sufficient to make this choice. It now queries the truth values of the variables present in the clause chosen, from the proof (the query size is therefore atmost 3 bits). It substitutes the truth values from the proof onto the clause and checks if the clause evaluates to true. If this is so, it halts and accepts, else it halts and rejects. If the formula is satisfiable and the certificate is right (there is atleast one such certificate), then this verifier will accept with probability 1, since an assignment that satisfies the formula must make each clause evaluate to true. However, if the formula is unsatisfiable, then each certificate makes atleast one clause evaluate to false. We choose one such clause with probability atleast $1/m$. Therefore, our error probability for unsatisfiable instances for this weaker verifier is atmost $1 - \frac{1}{m}$. The breakthrough with the PCP theorem is that it manages to get this error probability down to atmost $1/2$, while still querying a constant number of bits from the proof and using $O(\log n)$ random bits.

Next, we describe the MAX k-FUNCTION SAT problem and prove a hardness of approximation result for it using the PCP Theorem.

Problem 1 (MAX k-FUNCTION SAT) *Given n Boolean variables $x_1, \ldots, x_n$ and m functions $f_1, \ldots f_m$ each of which is a function of k of the boolean variables, find a truth assignment to $x_1, \ldots, x_n$ that maximizes the number of functions satisfied. Here k is a fixed constant (not part of input).*

Lemma 3 *There exists a constant k for which there is a polynomial-time reduction from SAT to MAX k-FUNCTION SAT that transforms a boolean formula ϕ to an instance I of MAX k-FUNCTION SAT such that,*

- *If ϕ is satisfiable, $OPT(I) = m$ and*
- *If ϕ is not satisfiable, then $OPT(I) < \frac{1}{2}m$*

Sketch of Proof: Since SAT$\in$ NP, it has a PCP($\log n$, 1) verifier, P. Since the random bit string used by P has length atmost $c \log n$, we can go over all possible random bit strings, of which there are atmost n^c. On being presented a random bit-string and ϕ, P queries atmost q bits of the proof. The instance I of MAX-k-FUNCTION SAT is constructed as follows: It has number of variables equal to the length of the proof (which can be atmost qn^c). Also, we have $k = q$. The idea is that for every "random" bit string, since ϕ is fixed, whether V accepts or rejects is a boolean function of just the bits of the proof queried. We construct this function. Since the number of bits q is a constant, we can run through all the 2^q bit-strings to construct the function in constant time. We thus have a function corresponding to every "random" bit-string. If ϕ is satisfiable, there exists a proof that makes the verifier accept for every random bit-string. Therefore there is an assignment to the variables of I, that makes each of the functions evaluate to true. If ϕ is not satisfiable, every proof leads to the verifier accepting on atmost half of the bit strings. That is, for every assignment to variables of I, less than half the functions evaluate to true. ■

3 Approximation and Complexity

In this section, we first describe a type of reduction called *L-reductions*, which unlike the many-one polynomial-time reductions used in the theory of NP-completeness, preserve approximation ratios. We then define a complexity class for optimization problems called MAXSNP, which is motivated by a structural characterization of NP from finite model theory (Fagin's theorem). We list several problems that are known to be MAXSNP-complete (under L-reductions). We then show that every problem in this class has approximation threshold strictly greater than zero. Finally, we show that none of the MAXSNP-complete problems have a PTAS, unless P=NP, which is the strongest result that we could expect to hold.

3.1 L-Reductions

We observe that the many-one polynomial time reductions do not preserve approximation ratios of approximation algorithms. We thus construct another type of polynomial-time reduction for optimization problems which preserve approximation ratios of approximation algorithms across the reduction and which are transitive, i.e. If problem A reduced to problem B and problem B reduced to Problem C, then there exists a reduction from A to C with the same properties.

Definition 4 (L-Reduction) *An L-Reduction from an optimization problem A to B is a pair of functions (R, S) both computable in logarithmic space, such that if x is an instance of A, then $R(x)$ is an instance of B and if y is an answer (feasible solution) of $R(x)$, then $S(y)$ is an answer of x. Furthermore,*

1. *If x is an instance of A with optimum cost OPT(x), then $R(x)$ is an instance of B with optimum cost satisfying*

   $OPT(R(x)) \leq \alpha \cdot OPT(x)$,
   where α is a positive constant

2. *If s is any feasible solution of $R(x)$, then $S(s)$ is a feasible solution of x such that,*

   $|OPT(x) - c(S(s))| \leq \beta \cdot |OPT(R(x)) - c(s)|$,
   where β is another positive constant particular to the reduction. In both cases, $c(\cdot)$ is used to denote the cost of of feasible solutions.

Observe that from the second property, we have that if s is the optimum solution of $R(x)$, then $S(s)$ is the optimum solution of x.

The following two propositions are an easy consequence of the definition.

Proposition 5 *If (R,S) is an L-reduction from problem A to problem B, and (R', S') is an L-reduction from B to C, then their composition $(R \cdot R', S' \cdot S)$ is an L-reduction from A to C.*

Proposition 6 *If there is an L-reduction (R,S) from A to B with constants α and β, and there is a polynomial-time ϵ-approximation algorithm for B, then there is a polynomial-time $\frac{\alpha\beta\epsilon}{1-\epsilon}$-approximation algorithm for A.*

The last proposition gives rise to the following corollary:

Corollary 7 *If there is an L-reduction from A to B and there is a PTAS for B, then there is a PTAS for A.*

The proofs are left as an exercise (and are also available in [2]).

3.2 The Class MAXSNP

Fagin's Theorem from finite-model theory [2] provides a structural characterization of the class NP. It states that all graph-theoretic properties in NP can be expressed in existential second-order logic. Strict NP or SNP is a subset of NP, which consists of properties expressible as

$\exists S \forall x_1 \forall x_2 \ldots \forall x_k \phi(S, G, x_1, \ldots, x_k)$,

where ϕ is a quantifier-free First-Order expression involving the variables x_i and the structures G (the input) and S. SNP contains decision problems. however, we are interested in defining a class of optimization problems. We can achieve this by slightly modifying the definition. SNP consists of structures G, for which $\exists S \forall x_1 \forall x_2 \ldots \forall x_k \phi$ is true. Now, we can relax this requirement a little bit. Instead of requiring ϕ to hold for all k-tuples, we only ask for the S that makes it hold for the largest possible number of tuples. This immediately gives us an optimization problem.

We now define the class MAXSNP$_0$ of optimization problems: Problem A in this class is defined in terms of the expression,

$$\max_S |\{(x_1, \ldots, x_k) \in V^k : \phi(G_1, \ldots, G_m, S, x_1, \ldots x_k)\}|$$

The input to a problem A is a set of relations $G_1, \ldots, G_m$ over a finite universe V. We seek a relation $S \subseteq V^r$, such that the number of k-tuples $(x_1, \ldots, x_k)$ for which ϕ holds is as large as possible.

Finally, we define the class MAXSNP to be the class of all optimization problems that are L-reducible to a problem in MAXSNP$_0$.

Example 1 *The problem MAX-CUT is in MAXSNP$_0$. and therefore in MAXSNP. This is because MAX-CUT can be written as follows:*

$$\max_{S \subseteq V} |\{(x, y) : ((G(x, y) \vee G(y, x)) \wedge S(x) \wedge \neg S(y))\}|.$$

Here, the graph is represented as a directed graph. The problem asks for the subset S of nodes that maximize the number of edges that enter S or leave S, that is the maximum cut in the underlying directed graph.

The following theorem shows that every problem in MAXSNP has nonzero approximation threshold.

Theorem 8 *Let A be a problem in MAXSNP$_0$. Suppose that A has the form $\max_s |\{(x_1, x_2, \ldots, x_k) : \phi\}|$. Then A has a $(1-2^{-k_\phi})$-approximation algorithm, where k_ϕ is the number of atomic expressions in ϕ that involve S.*

Sketch of Proof: The proof consists of two parts. In the first part, we use the structural characterization of each problem in MAXSNP$_0$ to come up with a corresponding MAX-k-FUNCTION SAT instance. In the second part, we provide an approximation algorithm for MAX-k-FUNCTION SAT.

Consider an instance of A. We wish to find that S that maximizes the number of tuples that evaluate to true. For the MAX-k-FUNCTION SAT instance, we basically construct a formula for every possible tuple. (Since k is a constant, there are atmost polynomially many formulae). For each tuple, given that G

and the tuple are fixed, the only variables left in ϕ are of the form $S(x_i)$, where $S(\cdot)$ is the indicator function for S. Let k_ϕ be the number of times that $S(\cdot)$ appears in ϕ. We thus have a MAX-k-FUNCTION SAT instance where $k = k_\phi$. The optimum assignment to this instance is clearly the indicator function for the optimum S of the instance of A in question.

Now given MAX-k-FUNCTION SAT instance, here is a $(1 - 2^{-k})$ approximation algorithm: Partition the variables into sets of size k. For each set in the partition, consider the functions that can be evaluated by an assignment to the corresponding k variables. By the pigeonhole principle, there is an assignment to these k variables that makes atleast 2^{-k} of these formulae evaluate to true. Find such as assignment by going over the 2^k possible assignments to these variables (This can be done in constant time, since k is a constant). Since at each step, we make atleast 2^{-k} of the formulae evaluate to true, the algorithm has an approximation ratio of 2^{-k} and is therefore a $(1 - 2^{-k_\phi})$-approximation algorithm.

The result follows. ■

3.2.1 MAXSNP-Completeness

Definition 9 *A problem in MAXSNP is MAXSNP-complete if all problems in MAXSNP L-reduce to it.*

Theorem 10 *MAX3SAT is MAXSNP-complete.*

Proof Idea: The proof uses the construction in Theorem 8 to first obtain an instance of MAX-k-FUNCTION SAT, for some k. This is then L-reduced to a 3-CNF SAT instance. [2] has the complete proof.

[2] proves that several important problems are MAXSNP-complete. These include MAX-CUT, 4-DEGREE INDEPENDENT SET and 4-DEGREE NODE COVER among others.

Next, we prove that no MAXSNP-complete problem has a PTAS, unless P=NP. This uses the inapproximability result for MAX-k-FUNCTION SAT, which was proved in Section 1 using the PCP Theorem.

Theorem 11 *No MAXSNP-complete problem has a PTAS, unless P=NP.*

Proof: We show this result by showing the existence of a problem in MAXSNP that does not have a PTAS. This immediately implies the required result, because if a MAXSNP-complete problem has a PTAS, then every problem in MAXSNP has a PTAS.

Recall that Lemma 3, proved that there exists a constant k, for which MAX-k-FUNCTION SAT has a no 1/2-factor approximation algorithm (and hence no PTAS). Now, for every constant k, we can construct a problem in MAXSNP (using the structural characterization of MAXSNP$_0$.), which has number of atomic expressions equal to k. By the construction in Theorem 8, we can reduce this problem to MAX-k-FUNCTION SAT. Lemma 3 applies for atleast one such problem. This proves the result.

■

References

[1] V.V. Vazirani, *Approximation Algorithms*, Springer, 2001.

[2] C.H. Papadimitriou, *Computational Complexity*, Addison Wesley, 1993.

[3] S. Arora, B. Barak, *Complexity Theory: A Modern Approach*, Ca mbridge University Press, expected in 2009. (A draft is available at `http://www.cs.princeton.edu/theory/complexity/` .

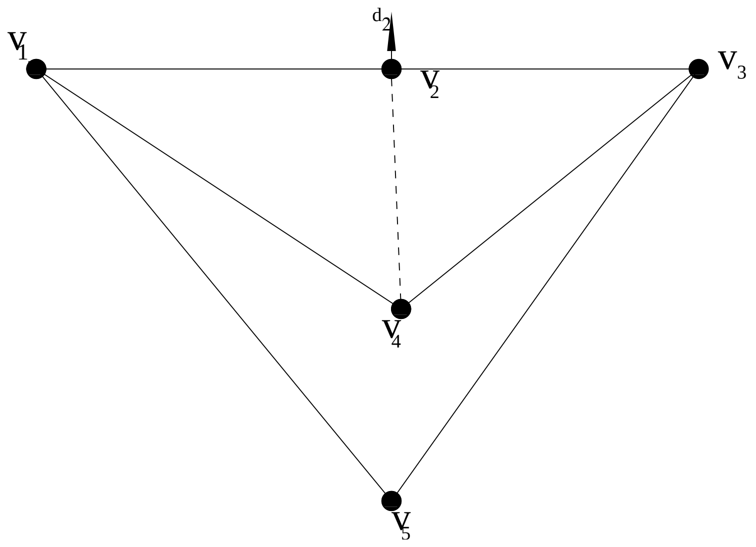
Recent advances in Complexity CIS 6930/CIS 4930 October 21, 2008

Lecture 17

Lecturer: Jialong Cheng Scribe: Jialong Cheng

1 Infinitesimal rigidity implies rigidity

First, the following is an example that shows a framework can be rigid but not infinitesimally rigid.



In this figure, the velocity d_2 on vertex v_2 is perpendicular to the edge v_1v_2 and v_2v_3. Perturbations on other vertices are zero. Hence it is an isometric perturbation. But the nonedge v_2v_4 will not keep the inner product to be zero.

$$L : R^{vd} \rightarrow R^{e}$$
$$L(\delta_1, \ldots, \delta_V) = (\ldots, (p_i - p_j) \cdot (\delta_i - \delta_j), \ldots),$$

where the component displayed on the right corresponds to the edge v_iv_j of K, with $i < j$.

It is easy to see that the infinitesimal congruences form a $d(d+1)/2$-dimensional subspace of the kernel of L. In this case, P is infinitesimally rigid if and only if $dim(kerL) = d(d+1)/2$.

A **manifold** is a mathematical space in which every point has a neighborhood which resembles Euclidean space, but in which the global structure may be more complicated. In discussing manifolds, the idea of dimension is important. For example, lines are one-dimensional, and planes two-dimensional.

In a one-dimensional manifold (or one-manifold), every point has a neighborhood that looks like a segment of a line. Examples of one-manifolds include a line, a circle, and two separate circles. In a two-manifold, every point has a neighborhood that looks like a disk. Examples include a plane, the surface of a sphere, and the surface of a torus. The trivial example of an n-dimensional manifold is R^n.

The following is the crucial part of this section:

Theorem 1.1 *Infinitesimal rigidity implies rigidity.*

Proof: Consider smooth map f:

$$f : R^{3V} \to R^{3V-6}$$
$$f(p_1, \ldots, p_V) = (\ldots, (p_i - p_j) \cdot (p_i - p_j), \ldots), \quad (i,j) \in E, i < j.$$

We can easily see that

$$df_p(\delta_1, \ldots, \delta_V) = 2(\ldots, (p_i - p_j) \cdot (\delta_i - \delta_j), \ldots).$$

Hence $df_p = 2L$. Therefore

$$\begin{aligned} \text{P is infinitesimally rigid} &\Leftrightarrow \quad dim(kerL) = \\ &\Leftrightarrow \quad \text{L is onto} \\ &\Leftrightarrow \quad df_p \text{ is onto} \\ &\Leftrightarrow \quad \text{P is a regular point of f.} \end{aligned}$$

So by the implicit function theorem, if P is a regular point of f, $f^{-1}f(P)$ is a 6-dimensional manifold near P. But $f^{-1}f(P) = [P]$. Since P is infinitesimally rigid, it can not degenerate to a subset of a line and therefore, [[P]] is also a 6-dimensional manifold near P. Hence [P] and [[P]] coincide near P. As the definition says, P is rigid. ■

The *rigid matrix* has $m$rows and nd columns. Each row corresponds to and edge while each column corresponds to a coordinate of a vertex. The non-zero values are the differences in the coordinate values for the two vertices.

$$R(\mathbf{p}) = \begin{bmatrix} & & & & \ldots & & & & \\ 0 & \ldots & (p_i - p_j) & \ldots & 0 & \ldots & (p_j - p_i) & \ldots & 0 \\ & & & & \ldots & & & & \end{bmatrix}$$

For example, consider K_3, the complete graph on three vertices,positioned in R^2. If the realization maps the vertices to locations $(0,1), (-1,0)$, and $(1,0)$, the matrix would be:

$$\begin{bmatrix} 1 & 1 & -1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 1 & -1 \\ 0 & 0 & -2 & 0 & 2 & 0 \end{bmatrix}$$

Recent advances in Complexity CIS 6930/CIS 4930 October 23, 2008

Lecture 18

Lecturer: Jialong Cheng Scribe: Jialong Cheng

1 Generic Rigidity

Theorem 1.1 *If a graph G has a infinitesimally rigid framework, then all its generic frameworks are rigid.*

Since infinitesimal rigidity implies rigidity, we can prove the following lemma to show the correctness of the theorem.

Lemma 1.1 *If a framework $G(p)$ is infinitesimally rigid, then $G(q)$ is infinitesimally rigid for all generic realization q.*

Proof: Denote $V(G)$ to be the set of infinitesimal motions of a framework, D to be the set of infinitesimal motions of a complete graph. Let dn be the number of dependent relations among the rows of $R(p)$. Let df be the degree of freedom, which is the dimension of $V(G)$ minus the dimension of D. Then we have $dim(V(G)) = vd - e + dn(G)$, and $dim(D) = \binom{d+1}{2}$. So,

$$\begin{aligned} df &= dim(V(G)) - dim(D) \\ &= [vd - e + dn] - \binom{d+1}{2} \end{aligned}$$

We can see easily that for a given graph, $df - dn$ is a constant(corresponds to d, v, and e). So $df_p - dn_p = df_q - dn_q$. Since q is a generic realization, $dn_p \leq dn_q$. And since $G(p)$ is infinitesimally rigid, so $df_p = 0$. Hence $df_p \leq df_q \leq 0$. So $df_q = 0$, which means that $G(q)$ is infinitesimally rigid. ■

Recent advances in Complexity CIS 6930/CIS 4930 November 4, 2008

Lecture 21

Lecturer: Jialong Cheng Scribe: Jialong Cheng

1 Thurston's Theorem [1]

Definition 1.1 *A **stress kernel** $K(\rho)$ of a framework ρ of Γ is the intersection of the null space of all its stress matrices: $K(\rho) := \bigcap_{\Omega \in S(\rho)} ker\Omega$. This is the same as the kernel of a generic stress matrix in $S(\rho)$. It is also isomorphic to the space of frameworks of Γ in $\boldsymbol{E}^1$ which satisfy all the stresses in $S(\rho)$(and maybe more). As has been noted before, for a generic framework ρ, $dimK(\rho)$ does not depend on the particular generic framework $\rho \in C^d(\Gamma)$, only on the graph Γ and the dimension d. Let k be this dimension.*

Definition 1.2 *A graph has a **minimal stress kernel** in $\boldsymbol{E}^d$ if $k = d + 1$.*

In fact, having a minimal stress kernel is equivalent to generic global rigidity.

Proposition 1.1 *The rank of G and $G \circ \ell$ is $vd - kd$, and the fiber of $G \circ \ell$ at a generic framework ρ is isomorphic to an open subset of $K(\rho)^d$.*

Definition 1.3 *A graph Γ has **maximal Gauss rank** in $\boldsymbol{E}^d$ if $rankG = vd - (d + 1)d$.*

The following is the main result of the paper, which is converse to Connelly's criterion.

Theorem 1.1 (Main Result) *If a graph Γ with $d + 2$ or more vertices does not have maximal Gauss rank in $\boldsymbol{E}^d$, then any generic framework $\rho \in C^d(\Gamma)$ is not globally rigid.*

The basic strategy used here is similar in spirit to the approach used by Hendrickson to show "redundant rigidity". Given a graph Γ which does not have maximal Gauss rank in $\mathbf{E}^d$, we construct spaces X and Y and a map $f : X \rightarrow Y$ so that the preimage in X of a point in Y correspond to incongruent frameworks with the same edge lengths, and we then show that the "degrees mod two" of this map is well defined and equal to 0. This degree is equal to the number of preimages at a regular value, modulo 2. Thus for such a map, any regular value in the image must have at least one more preimage, which represents a distinct framework in $C^d(\Gamma)$ with the same edge lengths, thus contradicting global rigidity in $\mathbf{E}^d$.

1.1 The domain

Definition 1.4 *For a framework $\rho \in C^d(\Gamma)$, the space of* ***stress satisfiers*** *is the space of all d-dimensional frameworks that satisfy all of the stresses in $S(\rho)$.*

References

[1] Steven J. Gortler, Alexander D. Healy, Dylan P. Thurston. Characterizing Generic Global Rigidity. arXiv:0710.0926v3, 2008

Recent advances in Complexity CIS 6930/CIS 4930 November 6, 2008

Lecture 22

Lecturer: Jialong Cheng Scribe: Jialong Cheng

1 Thurston's Theorem [1] continue..

The following is the theorem that we need to prove:

Theorem 1.1 (Main Result) *If a graph Γ with $d+2$ or more vertices does not have maximal Gauss rank in $\boldsymbol{E}^d$, then any generic framework $\rho \in C^d(\Gamma)$ is not globally rigid.*

In the last lecture we proved that $A(\rho)/Eucl(d)$ is a smooth stratified space with singularities of codimension at least 2. Now we give the definition of domain X:

$$X := \{x \in A(\rho) | \ell(x) \neq \ell(y) \text{ for any } y \in A(\rho) \text{ with non-trivial stabilizer } \}/Eucl(d). \tag{1}$$

Then we prove that X is non-empty, and in particular contains $[\rho]$.

1.1 The range

Consider the image $\ell(A(\rho))$ in $\mathbf{R}^e$ of squared edge lengths of elements in $A(\rho)$; call this space $B(\rho)$. We need the following definition:

Definition 1.1 *A semi-algebraic set S in $\boldsymbol{R}^e$ is **flat** if it is contained in a linear space of the same dimension as S.*

The main step of this subsection is to show that $B(\rho)$ is flat. First we consider $B(\rho)$ as a subset of the semi-algebraic set $M := \ell(C^d(\Gamma))$. Denote the subset of M of smooth points with the exact same tangent space as that at $\ell(\rho)$ as $B^\circ(\rho)$. In other words, $B^\circ(\rho)$is the fiber of the Gauss map G at $G(\ell(\rho))$. Similarly we denote $A^\circ(\rho)$ by the fiber of $G \circ \rho$. We have

Lemma 1.1 *Suppose that Γ is a graph and ρ is a generic framework in $C^d(\Gamma)$. Then $\overline{A^\circ(\rho)} = A(\rho)$ and $\overline{B^\circ(\rho)} = B(\rho)$.*

We now define $L(\rho)$ to be the linear space that contains $B(\rho)$ and is of the same dimension as $B(\rho)$. Then we have

$$Y := \{y \in L(\rho) | y \neq \ell(x) \text{ for any } x \in A(\rho) \text{ with non-trivial stabilizer } \}. \tag{2}$$

At this point we can prove that Y is a connected manifold with the same dimension as X.

1.2 The map

Define the map f as simply the restriction of ℓ to our spaces X and Y.

Definition 1.2 *A* ***proper map*** $f : X \to Y$ *between topological spaces is a continuous map so that the inverse image of a compact set is compact.*

Then we prove that f is a smooth proper map with mod-two degree of 0. Also, it is shown that $\ell(\rho)$ is a regular value of f. Then we can give the following proof of the theorem 1.1:

Proof: Since f is a smooth proper map with mod-two degree of 0 and $\ell(\rho)$ is a regular value of f, there must be an even number of points in $f^{-1}(f([\rho]))$. As shown before, $[\rho]$ is in the domain of f. So there must be another point in $A(\rho)/Eucl(d)$ that has the same edge lengths as ρ. ■

References

[1] Steven J. Gortler, Alexander D. Healy, Dylan P. Thurston. Characterizing Generic Global Rigidity. arXiv:0710.0926v3, 2008

Geometric Complexity and Applications CIS 6930 December 02 - December .., 2008

Aysegul Ozkan

The Sign Rank Of AC^0

Main result. Let $f_m(x,y) = \wedge_{i=1}^{m} \vee_{j=1}^{m^2} (x_{ij} \wedge y_{ij})$. Then the matrix $[f_m(x,y)]_{x,y}$ has sign-rank $2^{\Omega(m)}$.

The **sign-rank** of a matrix $A = [A_{ij}]$ with 1 entries is the least rank of a real matrix $B = [B_{ij}]$ with $A_{ij}B_{ij} > 0$ for all i, j.

Main result states that AC^0 contains matrices whose rank is rather stable in that it cannot be reduced below $2^{\Theta(n^{1/3})}$ by any sign-preserving changes to the matrix entries.

AC^0 is the family of polynomial-size unbounded-fanin constant-depth circuits with and, or, not gates.

Lemma 3.1 (Interpolation bound). Let $I \subset R$ be an interval of length L. Let p be a polynomial of degree $d \leq L$ such that

$$|p(x_i)| \leq 1 \text{ (i = 0, 1, . . . , d)},$$

where $x_0, x_1, ..., x_d \in I$ are some points with pairwise distances at least 1. Then

$$max_{x \in I} |p(x)| \leq 2^d \binom{L}{d}.$$

The Lagrange interpolating polynomial is the polynomial P(x) of degree $\leq (n-1)$ that passes through the n points $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, and is given by $P(x) = \sum_{j=1}^{n} y_j \prod_{k=1 k \neq j}^{n} \frac{x - x_k}{x_j - x_k}$

Lemma 3.2. Let k be an integer, $0 \leq k \leq n-1$. Let $f : \{0,1\}^n \to R$ be given with $\hat{f}(S) = 0$ for $|S| > k$. Then $|f(1^n)| \leq 2^k \binom{n}{k} max_{|x| \leq k} |f(x)|$.

Fourier Transformation:

Two functions f and g are orthogonal if $\langle f, g \rangle = 0$

For $S \subseteq [n]$, $\chi_S(x)$ is an orthonormal basis for the inner product space where inner product defined as $\langle f, g\rangle = 2^{-n}\sum_{x\in\{0,1\}^n} f(x)g(x)$ and $\chi_S(x) = (-1)^{\sum_{i\in S} x_i} = (-1)^{<x,s>}$.

$\langle \chi_S(x), \chi_S(x)\rangle = 1 \Leftrightarrow ||\chi_S(x)|| = 1$

$\langle \chi_S(x), \chi_T(x)\rangle = 0$ for $S \neq T$

Every function f(x) has a unique representation of the form $f(x) = \sum_{S\subseteq[n]} \hat{f}(S)\chi_S(x)$ where $\hat{f}(S) = \langle f, \chi_S\rangle$. $\hat{f}(S)$ are called the Fourier coefficients of f.

$\langle f, g\rangle = \sum_{S\subseteq[n]} \hat{f}(S)\hat{g}(S)$

If f(x) has degree d then $\hat{f}(S) = 0$ for $|S| > d$

Define the sets $Z = \{0, 1, 2, ..., 4m^2\}^m$, $Z^+ = \{1, 2, ..., 4m^2\}^m$. Define $F : Z \to \{-1, +1\}$ by

$$F(z) = \begin{cases} -1 & \text{if } x \in Z^+, \\ 1 & \text{otherwise.} \end{cases}$$

For $u, z \in Z$, let $\Delta(u, z) = |\{i : u_i \neq z_i\}|$ be the ordinary Hamming distance.

Lemma 3.4. Let Q be a degree-d real polynomial in m variables, where $d \le m/3$. Assume that $F(z)Q(z) \ge -1$ $(z \in Z)$. Then $|Q(z)| \le 2^{m+2d}$ at every point $z \in Z^+$ with $\Delta(u, z) < m/3$, where $u = (1^2, 3^2, 5^2, ..., (2m-1)^2) \in Z^+$.

Hamming distance between two strings of equal length is the number of positions for which the corresponding symbols are different.

Theorem 3.5. Let Q be a degree-d real polynomial in m variables, where $d \le m/3$. Assume that $F(z)Q(z) \ge -1$ $(z \in Z)$. Then $|Q(z)| \le 8^m$ $(z \in Z^+)$.

Theorem 4.1. There is a $\frac{1}{3}$m-orthogonalizing distribution μ for MP_m such that $\mu(x) \ge \frac{1}{2}8^{-m}2^{-n}$ for all inputs $x \in \{0,1\}^n$ with $MP_m(x) = -1$.

Theorem 5.1. Let $A = [A_{xy}]_{x\in X, y\in Y}$ be a real matrix with $s = |X||Y|$ entries $(A \neq 0)$. Assume that all but h of the entries of A satisfy $|A_{xy}| \ge \gamma$, where h and $\gamma > 0$ are arbitrary parameters.

Then $sign - rank(A) \geq \frac{\gamma s}{A\sqrt{s} + \gamma h}$

Forster.

The vectors $v_1, ..., v_n$ in R^r are said to be in general position if no r of them are linearly dependent. Let $U \subset R^r$ be a finite set of vectors in general position, $|U| \geq r$. Then there is a nonsingular transformation $A \in R^{r\times r}$ such that $\sum_{u\in U} \frac{1}{||Au||^2}(Au)(Au)^T = \frac{|U|}{r} I_r$

Theorem 2.4. Let X, Y be finite sets and $M = [M_{xy}]_{x\in X, y\in Y}$ a real matrix ($M \neq 0$). Put r = sign-rank(M). Then there is a matrix $R = [R_{xy}]_{x\in X, y\in Y}$ such that:

$$\text{rank(R)} = \text{r},$$

$$M \circ R \geq 0,$$

$$||R||_\infty \leq 1,$$

$$||R||_F = \sqrt{|X||Y|/r}$$

Main result

Let n and N be positive integers with $n|N$. Split [N] into n contiguous blocks, with N/n elements each. Let V(N, n) denote the family of subsets $V \subseteq [N]$ that have exactly one element from each of these blocks (in particular, $|V| = n$). Clearly, $|V(N,n)| = (N/n)^n$. For a bit string $x \in \{0,1\}^N$ and a set $V \in V(N,n)$, define the projection of x onto V by $x|_V = (x_{i1}, x_{i2}, ..., x_{in}) \in \{0,1\}^n$, where $i_1 < i_2 < \ < i_n$ are the elements of V.

For $\phi : \{0,1\}^n \to R$, the (N, n, ϕ)-**pattern matrix** is the real matrix A given by

$$A = [\phi(x|_V \oplus w)]_{x\in 0,1^N, (V,w)\in V(N,n)\times 0,1^n}$$

Geometric Complexity And Applications CIS 6930
Automated theorem Proving (Wu-Ritt)

Speaker : Ugandhar Reddy Scribe: Ugandhar Reddy

Introduction: As the name implies automated theorem proving is about verifying the Theorem statement. It is a way to verify whether the given hypothesis can generated from the conclusion. In gist, given a theorem statement we translate the hypothesis and conclusion of the theorem into a series of multivariant polynomials. Then automated theorem proving (which ever method-employed) verifies the theorem by showing that the conclusion polynomial can be generated from the hypothesis polynomials . In the below section we discuss about one such method i.e Wu-Ritt . In our earlier discussion we presented Grobner basis (polynomial Calculus) which also can verify geometric theorems. But it is inefficient compared to Wu-Ritt method which is discussed below.

In this chapter we shall present Wu's algorithm and demonstrate it on Appaloneous theorem.

The following are two multivariant polynomials written in terms of single variable y. The co-efficents ci , di in the below polynomials are monomials in k[x1...x(n-1)]

$$\begin{aligned} f &= c_p y^p + \ldots + c_1 y + c_0, \\ g &= d_m y^m + \ldots + d_1 y + d_0, \end{aligned}$$ [1]

Let us assume m<=p . Then to cancel out all the terms in f that have degree of y greater than m we do the following.

Cp may not be divisible by dm . So no matter with what we multiply g we may not be able to cancel out $c_p y^p$. To overcome this, we can multiply f with d_m which will ensure the leading term of d_m*f to be divisible by d_m.. Now, with g we can cancel out the leading term of d_m*f .

LC(r,y) means leading co-efficient of r written in the form of single variable polynomial in y.

R_1 = rem(f,g,y)

$R_1 = d_m{*}f - g{*}q_1 \rightarrow$ eq 1

$q_1 = LC(f,y)*y^{p-m} \rightarrow$ eq2

We got r_1 . r_1 can have degree more than m . i.e $deg(r_1,y) > deg(g,y)$. So repeating the above procedure results in d_m*r_1 being divisible by g

We get

$R_2 = d_m*r_1 - g*q2 \rightarrow$eq 3

$q_2 = LC(r_1,y)*y^{deg(r_1,y)-m}$

Replacing r1 in dm*r1 we get $d_m*(d_m*f - g*q_1)$. So R2 becomes

$R_2 = d_m*(d_m*f - g*q_1) - g*q2$

$R2 = ((d_m)^2*f) - g*(d_m*q_1 + q_2)$

The above equation is remainder = dividend – divisor * Quotient.

This means to cancel out the terms in f which have degree greater than m (w.r.t y) we can multiply f with $(dm)^s$ and divide it by g. This yields r_s that has degree less than m . s can be at most p-m+1 . This can happen if f has all the nonzero co-efficients for every y^m to y^p . This procedure of cancelling the leading term of f by g is called pseudo-division.

If f and g are in ideal then r_i produced in the above equations is also in ideal.

From the above procedure we can learn how to convert the polynomial form of hypothesis into polynomials in triangle form.

Let h1 … h_n be the hypothesis polynomials. While using automated theorem proving to verify theorems we need to know of two kinds of variables that occur in the polynomials of the hypothesis.

In order for a geometric figure to be placed on a plane certain co-ordinates needs to be fixed. These co-ordinates can take arbitrary values (First set). There are also different set of co-ordinates that depend on these values(second set). In Wu’s method we want the hypothesis polynomial equations to be converted into triangle equations w.r.t variables in the second set.

Assume order is assigned to the second set of variables. Starting from the highest order variable consider all the hypothesis polynomials that have the highest order variable term. Then write all these polynomials as single variable equation in the highest order variable. Let the set of these single variable polynomials be S .

If S consists of one polynomial in x_n then name it as f_n`

If S is more than one and one of those polynomial has degree of 1 in x_n , then psudo-divide the rest of the polynomials in S with that polynomial. Those polynomials eliminated of x_n become f1`..$f_{n-1}$` and the polynomial with x_n is named f_n`

If S is greater than 1 and no polynomial is degree I in x_n then do the following to generate equations of the form shown in fig 1.0

Repeat the following steps until degree of x_n in S becomes zero in all except one.

a. pick $a, b \in S$ where $0 < \deg(b, x_n) \leq \deg(a, x_n)$;
b. compute the pseudoremainder $r = \mathrm{Rem}(a, b, x_n)$;
c. replace S by $(S - \{a\}) \cup \{r\}$ (leaving the hypotheses not in S unchanged), [1]

Degree of x_n decreases each time in set S. So finally we get only one polynomial that has degree greater than 0 in x_n. We can obtain a zero pseudo-remainder in this step. This doesn't always imply reducibility. In these cases this procedure doesn't work properly. There is a different version of the theorem that takes care of all the cases. Also the polynomials we are considering may not be irreducible but Wu 's method needs irreducible poly to work perfectly.

The above procedure results in the equations of S in the following form

$$\begin{aligned} f_1' &= f_1'(u_1, \dots, u_m, x_1, \dots, x_{n-1}), \\ &\vdots \\ f_{n-1}' &= f_{n-1}'(u_1, \dots, u_m, x_1, \dots, x_{n-1}), \\ f_n' &= f_n'(u_1, \dots, u_m, x_1, \dots, x_n). \end{aligned}$$

[1] fig 1.0

Repeat the same procedure by forming new S for x_{n-1} out of the polynomials f1`..$f_{n-1}$` from of previous step output . Also include those hypothesis polynomials that were not considered in the part of earlier S . (The eliminated polynomials are never considered again like 'a') . This procedure results in the following form of polynomial equations called triangular form.

$$\begin{aligned} f_1 &= f_1(u_1, \dots, u_m, x_1), \\ f_2 &= f_2(u_1, \dots, u_m, x_1, x_2), \\ &\vdots \\ f_n &= f_n(u_1, \dots, u_m, x_1, \dots, x_n) \end{aligned}$$

[1]

We are done with the first half of Wu's method. The following is second step in Wu 's method

(i) *There are nonnegative integers $s_1, \ldots, s_n$ and polynomials $A_1, \ldots, A_n$ in the ring $\mathbb{R}[u_1, \ldots, u_m, x_1, \ldots, x_n]$ such that*

$$d_1^{s_1} \cdots d_n^{s_n} g = A_1 f_1 + \cdots + A_n f_n + R_0.$$ [1]

Using the division procedure explained above start dividing g with f_n . This results in R_{n-1}

R_{n-1} = rem(g,f_n,x_n)

$R_{n-1} = d_n^{s_n} g - q_n f_n.$ [1] (f_n and g are multivariant polynomials expressed in single variable x_n)

Similarly divide R_{n-1} with f_{n-1} .

R_{n-2} = Rem(R_{n-1},f_{n-1},x_{n-1})

$$\begin{aligned} R_{n-2} &= d_{n-1}^{s_{n-1}}(d_n^{s_n} g - q_n f_n) - q_{n-1} f_{n-1} \\ &= d_{n-1}^{s_{n-1}} d_n^{s_n} g - q_{n-1} f_{n-1} - d_{n-1}^{s_{n-1}} q_n f_n. \end{aligned}$$ [1]

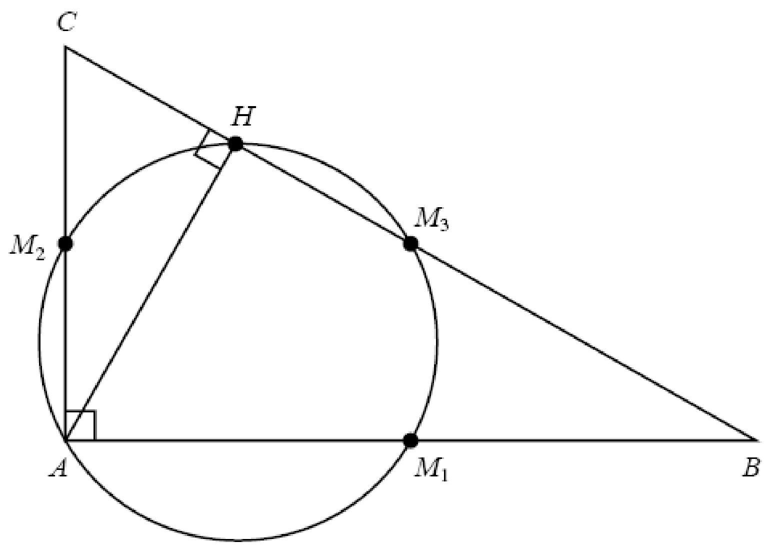
If we repeat the same way we get

$R_0 = d_1^{s_1} \cdots d_n^{s_n} g - (A_1 f_1 + \cdots + A_n f_n),$ [1]

if R_0 is zero polynomial then the above equation implies that at V(f_1 , f_2, ...f_n) [Set of points at which all of f1...fn evaluates to zero] g evaluates to zero or $d_j^{s_i}$ evaluates to zero .

Let us illustrate Wu's theorem with the following example

Theorem (The Circle Theorem of Apollonius). *Let $\triangle ABC$ be a right triangle in the plane, with right angle at A. The midpoints of the three sides and the foot of the altitude drawn from A to $\overline{BC}$ all lie on one circle.*



A at (0,0) and B at $(u_1, 0)$, $C = (0, u_2)$ [1]

No matter where we fix the diagram the above theorem satisfies. We can pick one vertex of the right triangle at origin (A). Then point C will be on y axis with zero x co-ord, point B will be on x co-ord with zero y co-ord .

Let $M_1 = (x_1, 0)$, $M_2 = (0, x_2)$, and $M_3 = (x_3, x_4)$ [1]be the mid points of the right triangle on each side. In A,B,C we fixed point A's both co-ord , fixed x co-ord of B and fixed y co-ord of X. The non-fixed co-ordinates in A,B,C can take any arbitrary value except zero. From these we can get an equation of M_1, M_2, M_3 . We get the following equations

$$h_1 = 2x_1 - u_1 = 0,$$
$$h_2 = 2x_2 - u_2 = 0,$$
$$h_3 = 2x_3 - u_1 = 0,$$
$$h_4 = 2x_4 - u_2 = 0.$$ [1]

Let us consoder the equations for for H . let H be (x_5,x_6)

$$AH \perp BC : h_5 = x_5u_1 - x_6u_2 = 0,$$
$$B, H, C \text{ collinear} : h_6 = x_5u_2 + x_6u_1 - u_1u_2 = 0.$$ [1]

Let the centre of the circle be O . Let O be (x_7,x_8)

Radius $M_1O = M_2O$

$$h_7 = (x_1 - x_7)^2 + x_8^2 - x_7^2 - (x_8 - x_2)^2 = 0.$$ [1]

Radius $M_2O = M_3O$

$$h_8 = (x_1 - x_7)^2 + (O - x_8)^2 - (x_3 - x_7)^2 - (x_4 - x_8)^2 = 0.$$ [1]

Equation fo the theorem Conclusion :

$HO = M_1O$

$$g = (x_5 - x_7)^2 + (x_6 - x_8)^2 - (x_1 - x_7)^2 - x_8^2 = 0$$ [1]

The expansion of the above equations results in the following set of equations.

$$\begin{aligned} h_1 &= 2x_1 - u_1, \\ h_2 &= 2x_2 - u_2, \\ h_3 &= 2x_3 - u_1, \\ h_4 &= 2x_4 - u_2, \\ h_5 &= u_2x_5 + u_1x_6 - u_1u_2, \\ h_6 &= x_1x_5 - u_2x_6, \\ h_7 &= x_1^2 - x_2^2 - 2x_1x_7 + 2x_2x_8, \\ h_8 &= x_1^2 - 2x_1x_7 - x_3^2 + 2x_3x_7 - x_4^2 + 2x_4x_8. \end{aligned}$$ [1]

Let us apply Step 1 of Wu's method: Triangular form

H8 , h7 are the only polynomials in x8 . That too they are of degree 1. Select any one of these as f8 and psudo-divide the other to eliminated x8 term from it. Let f8 = h8 and eliminate x8 from h7 .

$$\begin{aligned} f_7 &= \mathrm{Rem}(h_7, h_8, x_8) \\ &= (2x_1x_2 - 2x_2x_3 - 2x_1x_4)x_7 - x_1^2x_2 + x_2x_3^2 + x_1^2x_4 - x_2^2x_4 + x_2x_4^2. \end{aligned}$$ [1]

In case of x7 only f7(h7 became f7) contains x7 so f7 is f7

In case of x6 both h5 and h6 contain x6. Like procedure in case of x8 we set f6 as h6 and eliminated h5 to get the following i.e f5

$$f_5 = \mathrm{Rem}(h_5, h_6, x_6) = (u_1^2 + u_2^2)x_5 - u_1u_2^2.$$ [1]

Now the rest are all in triangular form.

Step 2: Expressing conclusion (g) in the terms of f_i . This is by successive psudo division .

Start with R_8 = g and unwound the recursive formula ($R_{i-1} = rem(R_i,f_i,x_i)$)until R_0 is reached .

$$\begin{aligned}
R_7 &= x_4x_5^2 - 2x_4x_5x_7 + x_4x_6^2 - x_4x_1^2 + 2x_4x_1x_7 + x_6x_1^2 - 2x_6x_1x_7 \\
&\quad - x_6x_3^2 + 2x_6x_3x_7 - x_6x_4^2, \\
R_6 &= x_4^2x_1x_5^2 - x_4^2x_1^2x_5 - x_4x_1x_6x_3^2 + x_4^2x_1x_6^2 - x_4^3x_1x_6 + x_4^2x_2^2x_5 \\
&\quad - x_4^2x_2^2x_1 - x_2x_4^3x_5 + x_2x_4^3x_1 - x_2x_1x_4x_5^2 - x_2x_1x_4x_6^2 \\
&\quad + x_2x_3x_4x_5^2 + x_2x_3x_4x_6^2 - x_2x_3x_4x_1^2 + x_4x_1^2x_6x_3 + x_4x_2^2x_6x_1 \\
&\quad - x_4x_2^2x_6x_3 + x_2x_1^2x_4x_5 - x_2x_3^2x_4x_5 + x_2x_3^2x_4x_1, \\
R_5 &= u_2^2x_4^2x_1x_5^2 - u_2^2x_4^2x_1^2x_5 + u_2^2x_4^2x_2^2x_5 - u_2^2x_4^2x_2^2x_1 - u_2^2x_2x_4^3x_5 \\
&\quad + u_2^2x_2x_4^3x_1 - x_4u_2^2x_2x_1x_5^2 + x_4u_2^2x_2x_3x_5^2 - x_4u_2^2x_2x_3x_1^2 \\
&\quad + x_4u_2^2x_2x_1^2x_5 - x_4u_2^2x_2x_3^2x_5 + x_4u_2^2x_2x_3^2x_1 - u_1x_5u_2x_4^3x_1 \\
&\quad + x_4u_1x_5u_2x_2^2x_1 - x_4u_1x_5u_2x_1x_3^2 - x_4u_1x_5u_2x_2^2x_3 \\
&\quad + x_4u_1x_5u_2x_1^2x_3 + u_1^2x_5^2x_4^2x_1 - x_4u_1^2x_5^2x_2x_1 + x_4u_1^2x_5^2x_2x_3, \\
R_4 &= -u_2^4x_4x_2x_3x_1^2 - u_2^4x_4^2x_2^2x_1 + u_2^4x_4x_2x_3^2x_1 + u_2^4x_4^3x_2x_1 \\
&\quad - u_2^2x_4u_1^2x_2x_3x_1^2 - u_2^2x_4^2u_1^2x_2^2x_1 + u_2^2x_4u_1^2x_2x_3^2x_1 \\
&\quad + u_2^2x_4^3u_1^2x_2x_1 - u_2^4x_4^3u_1x_2 - u_2^3x_4^3u_1^2x_1 + u_2^4x_4^2u_1x_2^2 \\
&\quad - u_2^4x_4^2u_1x_1^2 + u_2^3x_4u_1^2x_2^2x_1 - u_2^3x_4u_1^2x_1x_3^2 - u_2^4x_4u_1x_2x_3^2 \\
&\quad + u_2^4x_4u_1x_2x_1^2 - u_2^3x_4u_1^2x_2^2x_3 + u_2^3x_4u_1^2x_1^2x_3 + u_2^4x_4^2u_1^2x_1 \\
&\quad - u_2^4x_4u_1^2x_2x_1 + u_2^4x_4u_1^2x_2x_3,
\end{aligned}$$ [1]

$$\begin{aligned}
R_3 &= 4u_2^5x_2x_3^2x_1 - 4u_2^5u_1x_2x_3^2 + 4u_2^5u_1x_2x_1^2 - 4u_2^5x_2x_3x_1^2 \\
&\quad - 3u_2^5u_1^2x_2x_1 + 4u_2^5u_1^2x_2x_3 - 4u_2^4u_1^2x_1x_3^2 - 4u_2^4u_1^2x_2^2x_3 \\
&\quad + 2u_2^4u_1^2x_2^2x_1 + 4u_2^4u_1^2x_1^2x_3 - 4u_2^3u_1^2x_2x_3x_1^2 + 4u_2^3u_1^2x_2x_3^2x_1 \\
&\quad - 2u_2^6x_2^2x_1 - 2u_2^6u_1x_1^2 + 2u_2^6u_1x_2^2 + u_2^6u_1^2x_1 + u_2^7x_2x_1 \\
&\quad - u_2^7u_1x_2, \\
R_2 &= 2u_2^5u_1x_2x_1^2 - 2u_2^5u_1^2x_2x_1 + 2u_2^4u_1^2x_2^2x_1 - 2u_2^6x_2^2x_1 \\
&\quad - 2u_2^6u_1x_1^2 + 2u_2^6u_1x_2^2 + u_2^6u_1^2x_1 + u_2^7x_2x_1 - u_2^7u_1x_2 \\
&\quad + u_2^5u_1^3x_2 - 2u_2^4u_1^3x_2^2 + 2u_2^4u_1^3x_1^2 - 2u_2^3u_1^3x_2x_1^2 + u_2^3u_1^4x_2x_1 \\
&\quad - u_2^4u_1^4x_1, \\
R_1 &= -2u_2^6u_1x_1^2 - u_2^4u_1^4x_1 + u_2^6u_1^2x_1 + 2u_2^4u_1^3x_1^2, \\
R_0 &= 0.
\end{aligned}$$ [1]

We got R_0 as 0 . This implies g can have the same V(f1..fn) only if di ‘s evaluate to non zero on V(f1..fn).

For the theorem to be true we need to verify that all the di ‘s are non zero . d1 ..d4 are non zero as they are constants. The rest are below

$$\begin{aligned}
d_5 &= u_1^2 + u_2^2 \neq 0, \\
d_6 &= u_2 \neq 0, \\
d_7 &= 2x_1x_2 - 2x_2x_3 - 2x_1x_4 \neq 0, \\
d_8 &= -2x_4 \neq 0.
\end{aligned}$$ [1]

d5 is zero only if u1 and u2 are zero. In our construction we clearly stated u1 and u2 can be any value except zero. So it is true. From this argument d6 follows. d7 !=0 implies vertices of triangle are distinct. So it is also true. d8 : from h4 , we have -2 x_4 as $-u_2$. As u_2 is not zero so is d8 . These conditions imply g has V(f1..fn) as its variety.

Hence the theorem statement is true.

References:

[1] Ideals, Varieties, and Algorithms by Davis Cox , John Little , Donal O' Shea