Geometric Constraints II

Sep 5 ,7 and 12

(Dr.Sitharam)

(Course Overview, Background about Algorithms, Complexity, NP completeness,Geometric Complexity)

What is an Algorithm ?

An algorithm is a **well defined** procedure that takes some value/set as **input** and produces some value/set as **output**. It should **halt** in a finite number of steps and be **efficient**. (in time/space/communication) and lastly it should be **correct**

Deterministic and non-deterministic:

Deterministic algorithms solve the problem with exact decision at every step of the algorithm whereas non-deterministic algorithms solve problems via guessing although typical guesses are made more accurate through the use of heuristics.

* Do some reading in Big O notation if you aren't already familiar

P = NP ?

what is a polynomial time algorithm ?

what is a non-polynomial time algorithm ? (consists of problems that are verifiable in polynomial time, in other words a polynomial time verifier exists for the problem)

The class NPC consists of problems that are NP and can be reduced to another NP problem using a polynomial time algorithm: refer corman pg:967)

Decision Problems: If you can solve a decision version of a problem in polynomial time, it can be shown that the original version of the problem can be solved in polynomial time. Hence we deal with decision version of the problems since they are much easier to deal with.

some examples:
The clique problem:
$\{(G, K)$:G has a complete subgraph of size k$\}$
another version:
$\{(G, K) : \exists S \subseteq V(G) s.t\, |S| \geq$k and S is Clique in G$\}$

max clique:
$\{(G, K)$:k is the size of the max clique in G$\}$

Status as on date: No one knows whether P is in NP, ie, whether if you have a polynomial time algorithm for an NP problem. The relationship between NP and Co-NP is also not clear. These ideas are summed up best in the diagram in page 982 of the corman book.

Problems to think about ...

- What is the lower bound on any algorithm for sorting
- Show that sorting is reducible to Element Distinctness.
- note: ED = $\{(x_1....x_n) \in N : \forall i \neq j x_i \neq x_j\}$
- Show that Linear Programming is in NP $\cap$ Co.NP.

About Sorting Lower bound: We will outline a proof for the sorting lower bound here. Following that we will outline a geometric proof, which should give a taste of how to use geometric intuitions in regular problems that we encounter.

Sorting lower bound Proof outline: Essentially when we are sorting we are searching through a space of n! permutations to find the one we are looking for. Any algorithm that searches using comparison, will divide this space into at most half. (Binary Search). Searching like this, to reach a unique permutation, we need at least log(n!) steps. To calculate log(n!) we can use the stirlings approximation for n! Thus $log(n!) \approx nlog(n) - n + log(n)/2 + log(2 * pi)/2$

Geometric Proof for Element Distinctness Lower bound: This is problem is an example of reducing computational complexity of a problem into complexity of recognising a geometric set. This way you get an idea about the original problem you have.

Consider n space, all the points have n co-ordinates. Consider all the points that have a specific 2 co-ordinates equal, you can see that they are all in a hyper plane (eg. in 3 space, if the x and y co-ordinates are equal its a plane parallel to the xy plane). Now, using this idea, we can reduce the element distinctness problem to this problem : Namely that of finding if the co-ordinates is on one of these planes or not. (If the points is not in one of these planes, then it will have all its elements as distinct.).Now we have reduced the original problem into one of recognizing membership in a geometric set, the one defined by the intersection of the plane.

There are $\binom{N}{2}$ planes and they will divide the space into n! regions. We construct a decision tree. Imagine that the leaves of this tree imply a decision about where the point is receding. We need at least n! leaves, and the paths leading to them must be unique. Also note that the height of the decision tree is the complexity of your problem. At every level, we decide which path to follow. So that is one comparison for each level. So the minimum height of such a tree would be the limit on the computational complexity of the problem. This will be equal to log(n!) which is also the lower bound for the ED problem.

This course will be dealing with:

- 1. Exact embeddings for realization problem * Low Dimension (Dimension of metric space) * Algebraic Methods
- 3. Approximate realization or embeddings (Approximate as in with distortions)
- 2. Give a min dimensional embedding(normed space R_n equipped with L_2).

The Communication Complexity Problem: Alice receives an n-bit string x and Bob another n-bit string y, and the goal is for one of them (say Bob) to compute a certain function f(x,y) with the least amount of communication between them.

*Refer [2] for Matrix representation of the problem

We can represent this in the form of a matrix. $f_r : \{0,1\}^n X \{0,1\}^n - > \Re$
Q1.

1.Rank(f_r) is small. 2.$Sign(f_R(x,y)) = Sign(f(x,y))$
How to find f_r such that both 1, and 2 hold ?

Q2.

(p a point, and h a hyperplane) Given a matrix, $H(p.h) = 1$ if p is to left of h. $H(p,h) = -1$ if p is to the right of h. Find a realization of this in $\Re^d$ for the min possible d.

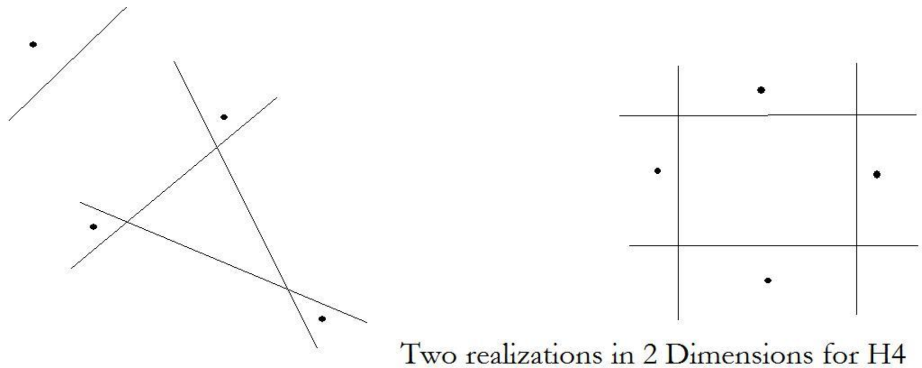
Definition: Hadamard Matrix: All the entries are ± 1 and the rows are orthogonal.

Problem: What is the minimum possible dimension d(n) such that $\exists$ n points and n hyperplanes such that every pair of points is separated by exactly half the hyperplanes. You can embed 2 planes with 2 points in dimension 2.

Above we show 2 methods of doing it.

Homework:

" Q1.What is the min dim into which H_8 can be embedded ?



Two realizations in 2 Dimensions for H4

H_8 is just $\begin{pmatrix} H_4 & H_4 \\ H_4 & -H_4 \end{pmatrix}$

Q2.What is the cut metric of a graph ?

[1] Refer to the thesis of Pranav at http://www.cise.ufl.edu/ ppd/mythesis.pdf.
[2] http://en.wikipedia.org/wiki/Communication_complexity