Lecture 1
Introducing Combinatorial Design Theory

Zur Luria *

Before embarking on our journey, there are several remarks that I would like to make.

- First, let's talk about the course. We will have around 14 lessons I think, and they will all take place between 13:00 and 15:00 on Monday. To get credit for the course, you should either give a 45 minute lesson (as in a seminar), or else there will be an oral examination at the end of the semester.
- The course website and mailing list.
- Supplementary material: Of course, as in any seminar based on a paper, the most important material is the paper itself. In this case, there is an additional resource - Keevash gave a series of talks in Jerusalem in January 2015 presenting this work, and the lectures were uploaded to youtube (search for "Keevash designs"). His presentation there is clear and accurate, and probably better than we will be able to achieve in this course, and so I strongly recommend watching them.

Now that that's out of the way, we can begin.

In this lecture, we will give an overview of the field of combinatorial design theory, which involves the study of finite objects satisfying certain balance and symmetry conditions. We will introduce some basic objects from this field, and discuss the relations between them and the status of the enumeration and random generation problems for each type of object.

- **Regular graphs:** A d-regular graph is a graph where all of the vertex degrees are d. Equivalently, it is a symmetric 0-1 matrix with zeros on the main diagonal, whose rows and columns sum to d. These objects come up in innumerable settings, and they have been well studied. In particular, a long line of work involving Read, Mckay, Wormald, Bollobas, Bender and Canfield and many others established asymptotic fomulae for the number of d-regular graphs on n vertices when $d = o(\sqrt{n})$.

  For a constant d, there is a simple algorithm, called the configuration model, for sampling d-regular graphs uniformly at random. Start with nd "half edges" (v, i) where $v \in V$ and $1 \leq i \leq d$, and choose a perfect matching uniformly at random on this set. This induces a regular (possibly non-simple) graph G that may include loops and multiple edges. The chance

*Institute of Theoretical Studies, ETH, 8092 Zurich, Switzerland. zluria@gmail.com. Research supported by Dr. Max Rössler, the Walter Haefner Foundation and the ETH Foundation.

that G is simple turns out to be dependent only on d and not on n, so when d is a constant, there is a constant chance that we get a simple graph. Conditioned on G being simple, it is uniformly distributed on the set of d-regular graphs.

- **Perfect Matchings in Hypergraphs:** A d-uniform hypergraph is a pair $H = \langle V, F \rangle$ such that $F \subseteq \binom{V}{d}$. A perfect matching in H is a collection M of hyperedges of H such that each vertex belongs to a single edge of M.

  The *adjacency matrix* of a graph G is the $n \times n$ 0-1 matrix A_G defined by $A_G(i,j) = 1$ iff $\{i,j\} \in E(G)$. Just as a perfect matching in G is equivalent to a symmetric permutation matrix contained in A_G, perfect matchings in hypergraphs also have a matrix representation, but it involves d-dimensional matrices.

  The adjacency matrix of a d-uniform order-n hypergraph H is the $[n]^d$ matrix A_H such that $A_H(i_1, ..., i_d) = 1$ iff $\{i_1, ..., i_d\} \in F(H)$. A perfect matching H is equivalent to a 0-1 $[n]^d$ matrix X contained in A_H such that:

  - Each *hyperplane* in X contains a unique one:
  $$\sum_{i_1,\dots,i_{k-1},i_{k+1},\dots,i_d} X(i_1, ..., i_d) = 1 \qquad \text{for all } 1 \le k \le d \text{ and } 1 \le i_k \le n.$$
  - X is *totally symmetric*:
  $$X(i_1, ..., i_d) = X(i_{\sigma 1}, ..., i_{\sigma d}) \qquad \text{for every } \sigma \in \mathbb{S}_d.$$

  The total number of d-uniform perfect matchings on n vertices is easy to compute. Clearly, n must be a multiple of d, and when this happens, the number of perfect matchings is $\binom{n}{d,\dots,d}$. However, the number of perfect matchings in a given hypergraph H is #P hard to compute for $d \ge 2$. For those unversed in complexity theory, this means that if $P \ne NP$, then there is no efficient algorithm that computes it. When $d \ge 3$ there is (probably) not even an efficient algorithm to check whether H has a single perfect matching, in contrast to the graph case, where such algorithms are known.

  A recent breakthrough by Kahn and Vu in the study of these objects was the computation of the threshold for the appearance of a perfect matching in a random hypergraph where every hyperedge appears with probability p. It was shown that, as in the graph case, a perfect matching appears shortly after there are no more isolated vertices.

- **Latin squares:** An order-n Latin square is an $n \times n$ matrix over the *symbols* $[n] := \{1, ..., n\}$ such that each row and each column is a permutation. The following is an example of a Latin square:
  $$L = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \end{pmatrix}.$$

  Latin squares also have an equivalent 3d matrix representation: A Latin square is equivalent to an $n \times n \times n$ 0-1 matrix A with a single one in each *line*, where a line is the set of entries obtained by fixing all but one of the indices and allowing that index to vary over $[n]$. The equivalence is given via $L(i,j) = k \Leftrightarrow A(i,j,k) = 1$. The Latin square L may be viewed as a "topographical map" of A. For example, the 3d representation of the Latin square above is

$$A = \left(\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \right).$$

One useful fact about Latin squares is that it is easy to construct them. For one thing, the multiplication table of any group is a Latin square (since $ab = c$ and $ad = c$ implies $b = d$). For another thing, we can construct the Latin square row by row and we will never get stuck. Choosing the next row in the partial Latin square reduces to choosing a perfect matching in a regular bipartite graph, and this is always possible because of Hall's marriage theorem.

This last insight yields good bounds on the number L_n of order-n Latin squares. Without going into details, permanent bounds can also bound the number of perfect matchings in a bipartite graph, and so by using these bounds to estimate the number of possibilities for each row, we can show that

$$L_n = \left((1 + o(1)) \frac{n}{e^2} \right)^{n^2}.$$

- **1-Factorizations:** A 1-factorization of a d-regular graph G is a partition of its edge set into d 1-factors, or perfect matchings. Equivalently, it is a proper edge-coloring of G using d colors. 1-factorizations of the complete graph K_n are equivalent to symmetric Latin squares with n on the main diagonal, via $L(i,j) = c(\{i,j\})$. We may view the relationship between Latin squares and 1-factorizations as analogous to that of Adjacency matrices of bipartite graphs (which need not be symmetric) and adjacency matrices of general graphs, which are symmetric and have zeros on the main diagonal. It is interesting to note that Latin squares are also equivalent to 1-factorizations of the complete bipartite graph $K_{n,n}$, and this fact yields a nice construction of 1-factorizations.

  Assume that n is a power of 2. We will construct a 1-factorization of K_n recursively as follows. Split $[n]$ into two sets of size $\frac{n}{2}$, color the edges between them using a Latin square and the edges inside each set recursively. This construction implies that $F(n) \geq F(n/2)^2 \cdot L_{n/2}$, which yields the lower bound

  $$F(n) \geq \left((1 + o(1)) \frac{n}{4e^2} \right)^{\frac{n^2}{2}}.$$

  By adding perfect matchings one by one and upper bounding the number of available perfect matchings at each step using standard bounds, we can show that

  $$F(n) \leq \left((1 + o(1)) \frac{n}{e^2} \right)^{\frac{n^2}{2}}.$$

  There is a substantial gap between the lower bound and the upper bound here.

- **Steiner triple systems:** A Steiner triple system (STS) is another analog of a perfect matching for 3-uniform hypergraphs.

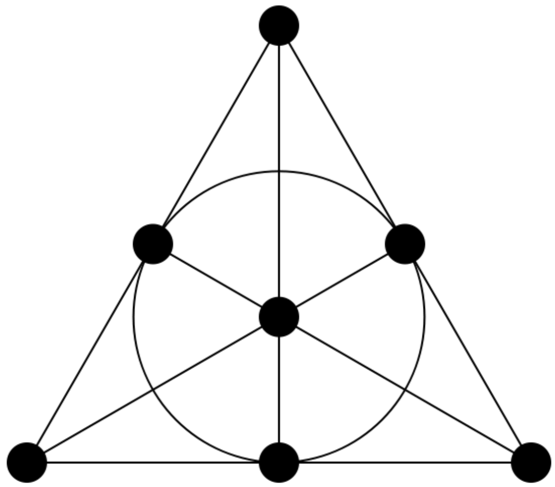

Figure 1: This is an example of an STS on 7 vertices. Each line represents a triple, and you may check that each pair appears in a unique triple.

- A perfect matching is a collection of pairs such that each vertex belongs to exactly one pair.
- An STS is a collection of *triples* such that each *pair* of vertices belongs to exactly one triple.

STSs are triangle decompositions of the edge set of K_n. Latin squares are equivalent to a triangle decomposition of the complete tripartite graph, and indeed it is possible to construct a Steiner triple system recursively using Latin squares. Such a construction yields the lower bound

$$STS(n) \geq \left((1+o(1))\frac{n}{3\sqrt{3}e^2})\right)^{\frac{n^2}{6}}.$$

The entropy method, which we will discuss later in the course, is a powerful tool for estimating the number of combinatorial objects of a certain size. An entropy proof shows that

$$STS(n) \leq \left((1+o(1))\frac{n}{e^2})\right)^{\frac{n^2}{6}}.$$

Once again, there is a substantial gap.

Steiner triple systems can be represented by Latin squares. Given an STS X, define a Latin square L by $L(i,j) = k$ if $\{i,j,k\} \in X$, and set $L(i,i) = i$ for every $i \in [n]$. This Latin square is even more symmetric than the Latin squares that represent 1-factorizations: $L(i,j) = k \Leftrightarrow L(k,j) = i \Leftrightarrow L(k,i) = j \Leftrightarrow$ ... and so on.

- (n, q, r, λ)**-Designs:** The following definition, central to this course, is an overreaching generalization of most of the objects described above.

Definition 0.1. *An (n, q, r, λ)-design is a collection X of q-element subsets of $[n]$ such that every r-element subset of $[n]$ is contained in exactly λ elements of X.*

So a d-regular graph is an $(n, 2, 1, d)$-design, a perfect matching in a d-uniform hypergraph is an $(n, d, 1, 1)$-design, and an STS is an $(n, 3, 2, 1)$-design. This is the concept that Keevash's work deals with. He totally solved the existence and enumeration problems here. Perhaps luckily for the remaining researchers in the field of combinatorial designs, there are many interesting objects that this concept doesn't capture, such as Latin squares, 1-factorizations, and others, some described below.

The aim of this course is to present Keevash's result for the special case of Steiner triple systems, as detailed in his paper "Counting designs". This will show in particular that the upper bound described in the previous section is tight.

- **Sudoku squares:** We are all familiar with Sudoku squares. They are order 9 Latin squares divided into 9 3×3 blocks, with the additional constraint that each block must contain the symbols $\{1, ..., 9\}$. It is possible to define Sudoku squares of size $N \times N$ for any number $N = n^2$, and then it becomes interesting to ask how many order-N Sudoku squares there are.

  Using entropy methods, which really are a very powerful tool, it is possible to show that the number of order-N Sudoku squares is at most $\left((1 + o(1))N/e^3\right)^{N^2}$, but I don't know of any good lower bounds. Perhaps because Sudoku squares are considered to be in the realm of recreational mathematics, there isn't too much serious work on them. It is conceivable, however, that Keevash's methods can be used to give a lower bound here as well.

- **Latin transversals:** Let L be an order-n Latin square. A transversal of L is a collection of n elements of L, exactly one from each row and each column and one of each symbol. Here is an example:

$$L = \begin{pmatrix} \color{red}{1} & 2 & 3 & 4 & 5 \\ 2 & 3 & \color{red}{4} & 5 & 1 \\ 3 & 4 & 5 & 1 & \color{red}{2} \\ 4 & \color{red}{5} & 1 & 2 & 3 \\ 5 & 1 & 2 & \color{red}{3} & 4 \end{pmatrix}.$$

  Transversals of Latin squares are well studied objects, of great importance for many problems of interest in the study of Latin squares. Here the biggest open problem remains the existence problem. It is well known that there are Latin squares of even order without any transversals, but it is conjectured that every Latin square of odd order has a transversal. This is the Ryser conjecture. Here, as far as I can see, Keevash's methods are not sufficient to solve the problem without substantial new ideas. This may be the biggest open problem remaining in the field of combinatorial designs.

  A major open question about Latin transversals was answered recently, when Sean Eberhard, Freddie Manners, and Rudi Mrazović from Oxford showed that the number of transversals in the cyclic Latin square is $(e^{-1/2} + o(1)) \cdot n!^3/n^{n-1}$, an unbelievably precise asymptotic formula. This formula asymptotically matches the maximal possible number of transversals in a Latin square.

- **The n-queens problem:** This is an old problem that Euler, among others, worked on, and as a chess player it is close to my heart. For any large enough n it is possible to place n queens on an $n \times n$ chessboard so that no two attack each other. The question is: In how many ways is it possible to do this?

  This question is related to the question about Latin transversals that was solved by Eberhard, Manners and Mrazović. A transversal of the cyclic Latin square corresponds to a solution of the n-queens problem for "semiqueens" on the "torus". The idea is that instead of the usual square chessboard, we have a toroidal chessboard where falling off the edge of the board sends us back to the opposite edge. A semiqueen is a queen that can move diagonally only in one of the two possible directions, say from the lower left to the upper right.

  Here too, it is difficult to see how Keevash's methods can be adapted without some substantial new ideas.

References

[1] *P. Keevash*, The existence of designs, arXiv preprint arXiv:1401.3665 (2014).

[2] *P. Keevash*, Counting designs, arXiv preprint arXiv:1504.02909 (2015).