Optimal Design of Experiments on Connected Units

How to use experiments to measure networks better,
and how to use networks to make experiments better.

Ben M Parker

September 2017

UNIVERSITY OF Southampton

Background

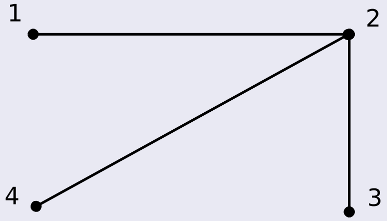
Mathematical research in Design of Experiments is concerned with studying the mathematical and statistical properties of an experiment, usually in being able to gather the most information from the experiment with fixed resources.

Background

- Often we assume when designing experiments that subjects (experimental units) are independent.
- Particularly, if we apply a treatment to one subject, we generally assume that the treatment does not affect other subjects. (SUTVA-Stable Unit Treatment Value assumption)
- In my research, I investigate how the structure of relationships between subjects affects the design of experiments on these subjects.
- In a JRSS paper [3] we published a first paper on design for networks, which we review in this talk as background, and then expand to current work.

Introduction

- We consider a graph, as a collection of nodes N and edges E.
- The nodes represent subjects on which we apply some treatment. We have $|N| = n$ subjects.
- The edges represent some relationship between the subjects.
- We allow the relationship between our subjects to be specified by the adjacency matrix A where $A_{ij} = 1$ if i and j are related and $A_{ij} = 0$ otherwise. By convention, $A_{ii} = 0$.
- We assume initially that links are non-directional, such that $A_{ij} = A_{ji}$, i.e. A is symmetrical.



$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Examples of problems we might be interested in

- The nodes may be subjects in a marketing experiment.
- The links may be a social relationship, such as being “friends” on Facebook, or “working together” in a statistics group.
- The treatments may be different advertising campaigns, such as “Buy Coke” or “Buy Pepsi” (or “Smoking Kills!”.)
- The responses may be the amount of a products bought by a subject, or how much the subject likes the product.

Idea behind model:

If a subject is exposed to a treatment, their friend may see this treatment and their response may be altered (positively or negatively). We assume that if a relationship exists between two individuals, the response of the first subject is dependent on the treatment applied to the second.

Optimal design without any network effect

Model:

$$Y_i = \mu + \tau_{t(i)} + \epsilon_i$$

where

- ϵ_i are i.i.d with mean 0, variance σ^2;
- $t(i)$ is the treatment applied to subject i;
- τ_j are treatment effects for $j = 1, \ldots, m$;
- we assume w.l.o.g that $\tau_m = 0$ for uniqueness.

Optimality Criteria

When we design an experiment, we need to consider what we want to know: for example, we may be more concerned with precise measurement of the effect of treatment 1 than we are with that of treatment 2.
We assume in this work that we wish to minimise the average variance of all estimates of pairwise differences of treatment effects:

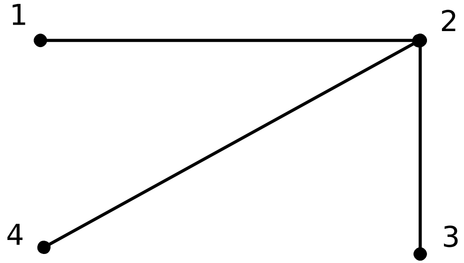
$$\frac{2}{m(m-1)} \sum_{j=1}^{m} \sum_{l>j}^{m} \mathrm{Var}(\widehat{\tau_j - \tau_l}).$$

This is A-optimality for estimating the difference in treatment effects.

For $m = 2$ treatments, it is clear that the balanced design with an equal number of subjects given each treatment is optimal.

The idea

We assume that if we apply a treatment j to a subject, there will be a network effect of γ_j to all neighbours of that subject.



For example, if we apply treatment 1 to subject 2, there will be a network effect of γ_1 on subject 1 as well as the standard subject effect on subject 1 from its own treatment.

Optimal design with network effect

We introduce the

Linear network effect model:

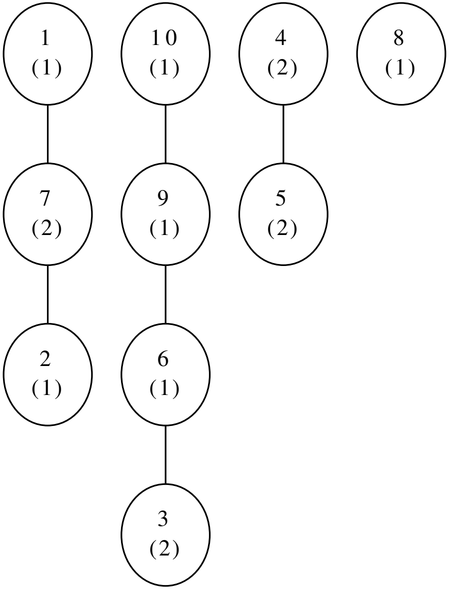
$$Y_i = \mu + \tau_{t(i)} + \sum_{k=1} A_{ik}\gamma_{t(k)} + \epsilon_i$$

where

- as before $t(k)$ is the treatment given to subject k;
- and now γ_j is the corresponding network effect, which is the change in the behaviour on a subject due to giving a connected subject a particular treatment.

We may also now be interested in the network effects themselves, and can consider A-optimality for the network effects.

Example 1



By exhaustive search, we find optimal designs for the $m = 2$ treatment case are :

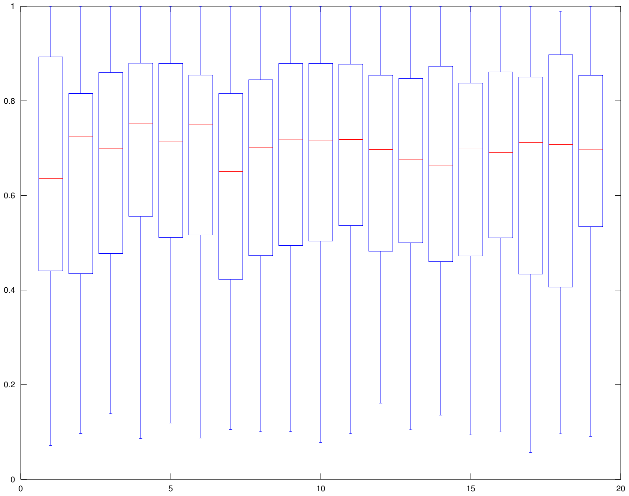
- $\{1, 1, 2, 2, 2, 1, 2, 1, 1, 1\}$ for estimating the difference in the subject effects (i.e. we give treatment 1 to subjects 1,2,6,8,9, and 10 and treatment 2 to the other subjects).
- $\{2, 2, 1, 1, 1, 1, 2, 2, 2, 1\}$ for optimality in estimating the difference in the network effects γ_i.

Note that

- The optimal design for estimating the difference in treatment effects is not balanced; i.e treatments 1 and 2 are not applied to an equal number of subjects. This is an unusual property in optimal design.

Neglecting the network effect

Boxplot of efficiencies of balanced designs for 20 random networks:



Bias of design

In example 1, if we wrongly assumed there was no network effect when in fact it existed, then for the balanced design $\{1, 1, 1, 1, 1, 2, 2, 2, 2, 2\}$ which is optimal in the case there were no network effects, the bias is as

$$\begin{pmatrix} 0 & 0 & 0.6 & 0.8 \\ 0 & 0 & -0.2 & -0.2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} \mu \\ \tau_1 \\ \gamma_1 \\ \gamma_2 \end{pmatrix}$$

If the network effects are large compared to the subject effects, ignoring network effects might lead to the subject effects being estimated badly with very large bias.

Important message!

By not taking into account a network effect in our design, our experiment has higher variance than necessary, and/or biased estimators.

Example 2: A Social Network

- We wish to design an experiment to determine the effectiveness of a new advertisement.
- Some people are sent an amusing video advert for a type of soft drink (treatment 1), and some are shown another advert (treatment 2).

- The amount of soft drink each of the 20 people buy in the following week would then be found by a survey.
- Interested in how effective the advert is, but also how effective the advert is at being conveyed to friends of those who saw the advert (this is known as viral marketing), so 20 people are chosen where the connectivity structure of their online social network is known.

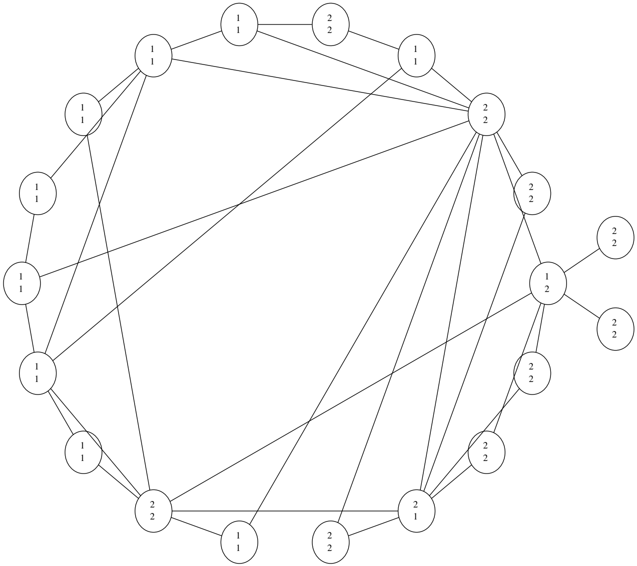


Figure: Optimal design for (top) estimating the subject effects of the advert and (bottom) estimating the network effects of the advert

Finding the Optimal Design

All properties of the design can be found by calculating the information matrix. For this example, it turns out to be:

$$I = \begin{pmatrix} n & n_1 & n_2 & \dots & n_{m-1} & \sum_i n_{i1} & \sum_i n_{i2} & \dots & \sum_i n_{im} \\ n_1 & n_1 & 0 & \dots & 0 & n_{11} & n_{12} & \dots & n_{1m} \\ n_2 & 0 & n_2 & \dots & 0 & n_{21} & n_{22} & \dots & n_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ n_{m-1} & 0 & 0 & \dots & n_{m-1} & n_{m-1,1} & n_{m-1,2} & \dots & n_{m-1,m} \\ \sum_i n_{i1} & n_{11} & n_{12} & \dots & n_{1,m-1} & n_{11}^{(2)} & n_{12}^{(2)} & \dots & n_{1m}^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \\ \sum_i n_{im} & n_{m1} & n_{m2} & \dots & n_{m,m-1} & n_{m1}^{(2)} & n_{m2}^{(2)} & \dots & n_{mm}^{(2)} \end{pmatrix}.$$

- n_i is the number of nodes given treatment i.
- n_{ij} is $\#$ of ways of drawing a path of length 1 from a node given treatment i to one given treatment j.
- n_{ij} is $\#$ of ways of drawing such a path of length 2....

General Principles for Design Algorithm

- $\mathcal{X}$ is the design space, the set of all possible assignments of treatments to experimental units for our experiment
- To assess how good a design $x \in \mathcal{X}$ is we calculate the value of some optimality criterion $f(x)$
- Typically this will be some function of the Fisher information matrix, $I(x)$
  - e.g A-optimality involves taking the average variance of all unknown parameters, which corresponds to $f(x) = \text{tr}(I^{-1}(x))$.
  - D-optimality which minimises the confidence region for a combination of parameters, can be expressed as $f(x) = \det(I^{-1}(x))$
- Finding I, then I^{-1}, and sometimes $f()$ is computationally hard

The curse of dimensionality

- Typically the design space $\mathcal{X}$ to search may be large.
- For example, if we have n experimental units which each take values in some set $\mathcal{F}$ then the size of the design space will be $|\mathcal{F}|^n$, and we suffer from the curse of dimensionality.
- Even if F is the binary set $\{0, 1\}$ such that each experimental unit may be assigned one of two treatments, then the size of the design space may be 2^n.
- If F is a continuous set (e.g. $\mathcal{R}$ or, a set of equivalent size, the interval $[0, 1]$, the design space is (infinitely) bigger.
- In this work we consider unstrucured treaments, so $F = \{(1, 2, \ldots, m)^n\}$

Reducing the design space

Ideally, we evaluate the optimality criterion $f(x)$ at all possible designs $x \in \mathcal{X}$ to find the optimal design $x^* = \arg\max_{x \in \mathcal{X}} f(x)$, but for large design spaces and/or criteria that are difficult to calculate, computational restrictions may mean we can not do this in a reasonable computation time.

In some cases, analytic results allow designs to be found exactly without a search algorithm.

However, in general we seek some algorithm that enables us to find the optimal design (or a design which is near optimal).

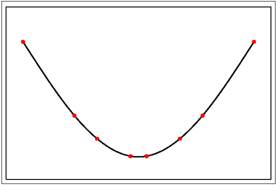
In overview, these come in two categories

- reducing the complexity of the calculation of $f(.)$
- evaluating a subset of $\mathcal{X}$ such that the overall number of calculations of $f(.)$ is smaller.

Reducing the complexity of $f(x)$

Methods includes:

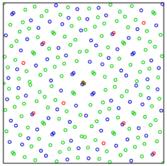
- emulation, used extensively within computer designs, where instead of evaluating a complicated function $f()$, we evaluate an emulator $g()$, a function which is simpler to evaluate but we believe has the same properties such that $f(x) \approx g(x)$ for all $x \in \mathcal{X}$.
- Updating formula can also be used in some problems, where we know that $f(x+z) = f(x) + h(z)$ for some easy-to-calculate function h.

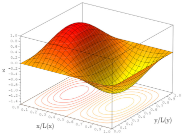


Searching a subset of $\mathcal{X}$

Methods include

- Space-filling designs, where a representative sample of designs in $\mathcal{X}$ are evaluated.
- many optimisation algorithms, which use the t previously evaluated designs $x_0, x_1, \ldots x_{t-1}$ and their optimality criterion $f(x_0), f(x_1), \ldots, f(x_{t-1})$, to choose which design x_t should be evaluated next.
  - Fedorov Exchange algorithms, coordinate exchange algorithms, simulated annealing, particle swarm optimisation, many stochastic search algorithms such as Nelder-Mead, are the subject of research.
  - Some of these algorithms are deterministic, in that the choice of x_t is mandated, others are stochastic in that x_t is chosen randomly.
  - To avoid local maximum designs, the initial x_0 is often chosen randomly, even if the rest of the algorithm is deterministic.

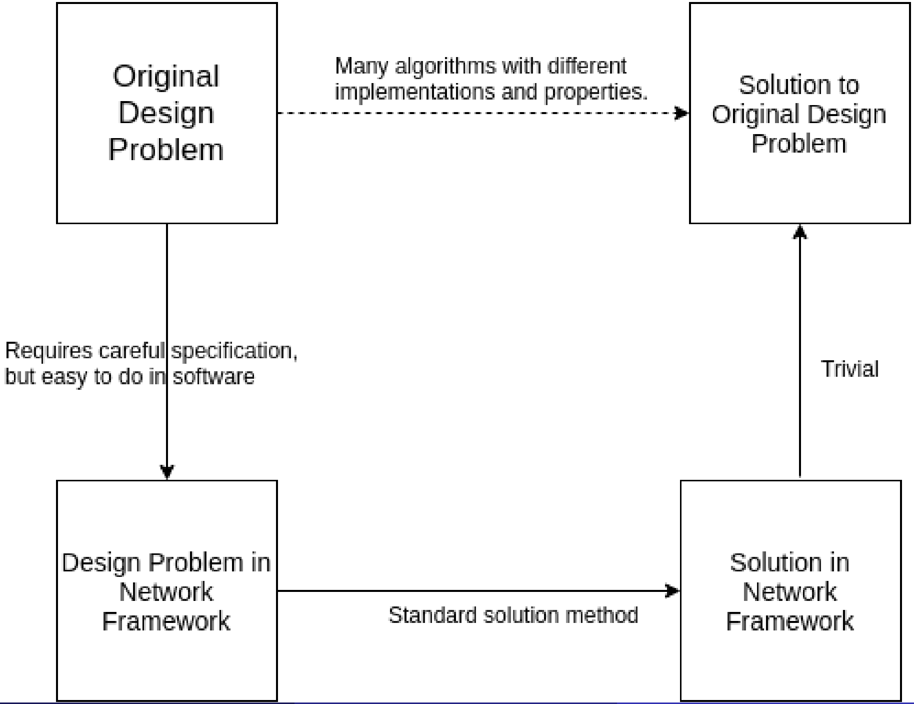




A network approach for experimental design

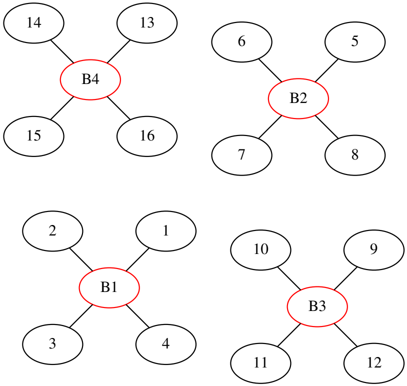
Much of the literature in experimental design revolves around trying to find better algorithms for particular problems. In this work, we do not seek to find new algorithms, but to suggest how mapping the problem to a network domain allows us to vastly reduce the number of designs evaluated in order to allow us to find better designs. Existing algorithms are still used within this context.
We claim that there are many standard problems in design that can be abstracted by representation as a network, for example

1. Block Design
2. Row-column design
3. Crossover design
4. etc, etc,

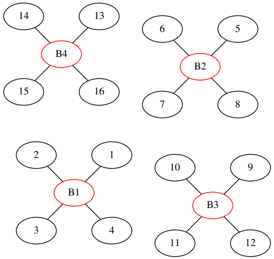

Original Design Problem
Many algorithms with different implementations and properties.
Solution to Original Design Problem
Requires careful specification, but easy to do in software
Trivial
Design Problem in Network Framework
Standard solution method
Solution in Network Framework

Simple Blocked Experiment

Block	Experimental Units			
1	1	2	3	4
2	5	6	7	8
3	9	10	11	12
4	13	14	15	16



Block	Exp. Units			
1	1	2	3	4
2	5	6	7	8
3	9	10	11	12
4	13	14	15	16



	Original Problem	Network Problem
No of Treatments	2	6
Exp. Units	$\{1, 2, \ldots 16\}$	$\{1, 2, \ldots, 16, B1, B2, B3, B4\}$
Wish to estimate	$\tau_1 - \tau_2$	$\tau_1 - \tau_2$
Opt. criterion	A	A_s
Restrictions	Can apply either treatment to any unit.	Can apply treatments 1,2 to units $\{1, 2, \ldots, 16\}$, and treatments 3,4,5,6 to units B1-B4.

To find the optimal design, we can model for our blocked experiment as

$$Y_i = \mu + \tau_{t(i)} + \sum_{k=\{1,\ldots,16,B1,B2,B3,B4\}} A_{ik}\gamma_{t(k)} + \epsilon_i, \quad i = 1, \ldots, 16$$

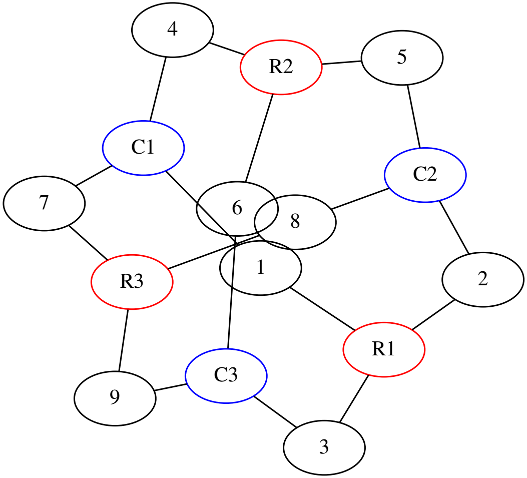
where $A_{ij} = 1$ whenever there is a link (e.g. $A_{14,B4} = 1$), and $A_{ij} = 0$ otherwise.
By writing $b_{j(i)} = \sum_{k=\{1,\ldots,16,B1,B2,B3,B4\}} A_{ik}\gamma_{t(k)}$, we can see immediately that this is equivalent to

$$Y_i = \mu + \tau_{t(i)} + b_{j(i)} + \epsilon_i,$$

a more familiar representation of a blocked experiment, where $b_j(i)$ is block effect of experimental unit i being in block j. As $A_{ij} = 1$ if and only if node i is linked to node j, and this only happens when experimental unit i is in block j for $j = \{B1, B2, B3, B4\}$, we can replace $\gamma_{t(B1)} = \gamma_{m+1} = b_1$, $\gamma_{t(B2)} = \gamma_{m+2} = b_2$, and so on.

Row Column Design

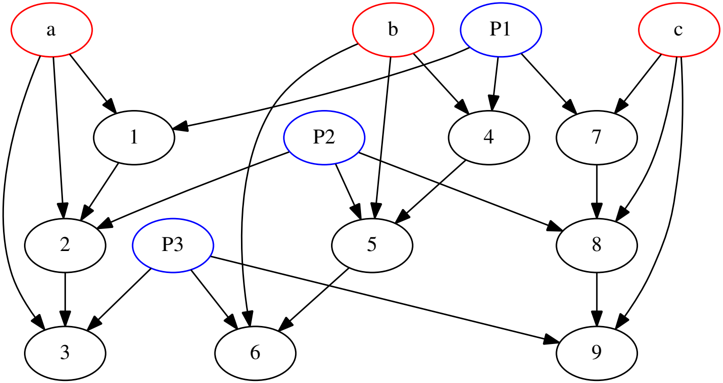
	C1	C2	C3
R1	1	2	3
R2	4	5	6
R3	7	8	9



;

Crossover Design

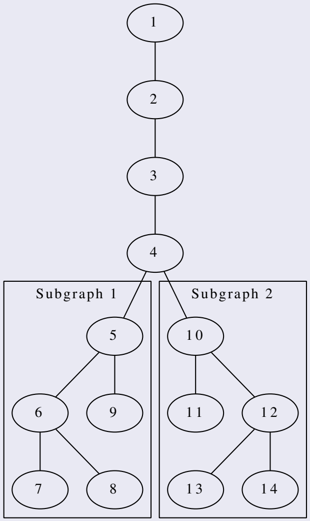
		Period		
		P1	P2	P3
Subject	a	1	2	3
	b	4	5	6
	c	7	8	9



Symmetry of labels

- For our criteria we are only interested in differences between treatments
- The treatment effects themselves are irrelevant, and treatments are equivalent up to relabelling.
- For example, 1,2,2,3,1 and 1,3,3,2,1 are equivalent.
- We can thus reorder any design so that we only evaluate designs where the first occurrence of label j must come before the first occurrence of label $j+1$.

Symmetry of graphs



Subgraphs 1 and 2 are exchangeable; i.e for subdesign A on subgraph 1, and subdesign B on subgraph 2 (call this [A1,B2]), we need not also consider [A2,B1] as by symmetry this design has the same criterion value.
We can reduce our design space greatly if we can identify subgraphs where the designs are exchangeable.
This is equivalent to finding an automorphism for our network, a relabelling or permutation of the set N such that the edges E are preserved.
This is the Graph Automorphism Problem.
Recently it has been shown (but not yet peer reviewed!) that this is a hard problem, but is not (quite) NP-hard.

Graph Automorphism Problem practically

In practice fast algorithms exist already that can effectively and quickly find automorphisms in most cases, for example the VF2 algorithm[1], which in general will find isomorphisms between two graphs G_1 and G_2.
This algorithm is based on a tree search, where a set of nodes is maintained where a partial match between subgraphs of G_1 and G_2 occurs, and then nodes connected to these subgraphs are considered to see if they can extend the matching subgraphs.
This algorithm is fast, and implemented by the popular `igraph` package (`http://igraph.org/`) available in many programming languages, including R.

General Algorithm for using networks for design

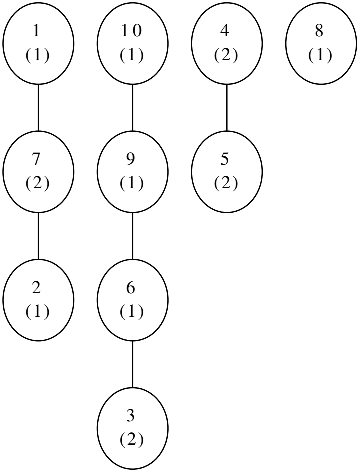
1. Rewrite original problem in network form.
2. Find the set of mappings `isos` of network.
3. Set $k = 1$, $num_{\text{eval}} = 0$ and pick initial candidate design x_1.
    1. Check whether $x_k = \min_{\text{lex}} \texttt{isos}(x_k)$, i.e. if x_k is the minimum design in the lexicographical order of all isomorphisms.
    2. If so,
        - evaluate the optimality criterion $d_k = f(x_k)$
        - increment num_{eval} by 1.
    3. Set $k = k + 1$. Pick next candidate design x_k according to algorithm of choice `next` based on $x_1, \ldots, x_{k-1}$ and $d_1, \ldots, d_{k-1}$, so that $x_k = \text{next}[(x_1, \ldots, x_{k-1}), (d_1, \ldots, d_{k-1})]$.
    4. If we have reached a stopping criterion, i.e if $\text{stop}[(x_1, \ldots, x_k), (d_1, \ldots, d_k), num_{\text{eval}}] = 1$ for some stopping criterion `stop`, then stop, otherwise Repeat from 1.

- Lexicographical order means a dictionary order such that

$$111 < 112 < 121 < 122 < 211 < 212 < 221 < 222$$

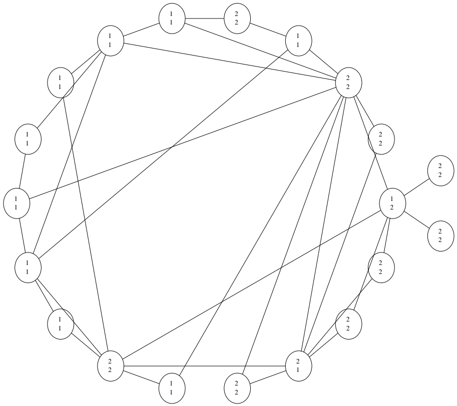
- Typical choices of the `next` algorithm might be the Fedorov exchange algorithm, an exhaustive search, etc.
- Typical choices of the `stop` criterion might be
  - Stop if $k = k_1$ or $num_{\text{eval}} = k_2$ representing some fixed budget on the number of designs examined or function evaluations made,
  - or that $\max(d_{k-t}, \dots, d_k) = d_{k-t}$, i.e. we have not seen an improvement in the last t steps.

Example 1



We calculate the same optimal design, but evaluate the information matrix 236 times as opposed to 507 times, and use a processing time of 0.02 seconds as opposed to 0.04 seconds.

Example 2



524287 without isomorphisms, 221183 with.
58.58s without, opposed to 31.56 with.

Example 3: Non-rectangular field trials

Most field trial experiments focus on rectangular fields. Let us assume we have an irregularly shaped field divided into plots as shown below 1.

		1	2	3
		4		5
6	7	8	9	10
11	12	13	14	15

Table: An irregularly shaped collection of plots in a field.

We perform a trial on 3 different fertilisers, to estimate with minimum average variance the difference between fertilizer (subject) effects.
We assume that any fertiliser applied might leach to any of the 8 possible plots that touch the treated plot, including those that only touch at corners.

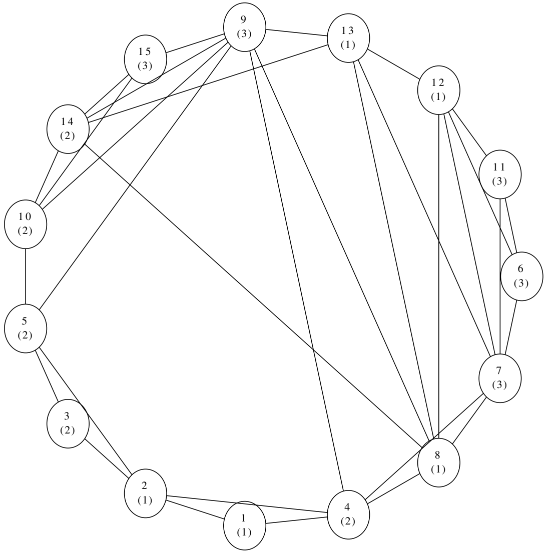


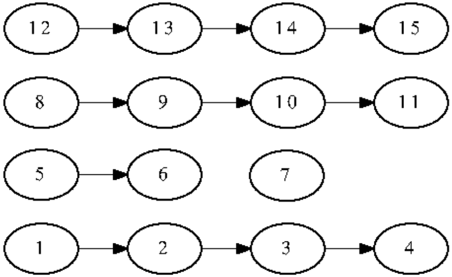
Figure: The top number is the plot number, and the bottom number the optimal treatment allocation. The optimal function value of the third criterion is 0.4055.

Example 4: A crossover trial with planned dropouts

In a four period crossover trial, assume that one participant will not be able to take the treatment in the third period of the trial.

		Period			
		1	2	3	4
Subject	a	1	2	3	4
	b	5	6		7
	c	8	9	10	11
	d	12	13	14	15

- We use each subject/period combination as a node in our network
- Modify our methodology such that links are uni-directional;
- Perform exhaustive search as before to minimise variance in our subject effects.



		Period			
		1	2	3	4
Subject	a	1	1	2	1
	b	1	3		2
	c	2	2	3	1
	d	3	3	3	2

The optimal design has optimality criterion 0.4128. Note that it is quite different in form from usual cross-over designs.

Description	n	No. automorphisms	Evaluations without automorphisms	Evaluations with automorphisms
Small social network (1)	10	8	507	236
Small social network	10	1	511	511
Larger social network (2)	20	8	524,287	221,183
Block design with neighbour effects (3)	12	384	535,008	18,766
Non-rectangular field trial (3)	15	2	2,368,741	1,581,572
Crossover trial with dropouts(4)	15	6	2,262,800	904,555

Description	n	No. automorphisms	Time without automorphisms	Time with automorphisms
Small social network (1)	10	8	0.04	0.02
Small social network	10	1	0.04	0.04
Larger social network (2)	20	8	58.58	31.56
Block design with neighbour effects (3)	12	384	108.52	33.68
Non-rectangular field trial (3)	15	2	279.6	197.58
Crossover trial with dropouts(4)	15	6	283.86	134.26

Computational Justification: Complexity without isomorphisms

Recall that $|\mathcal{X}| = n^m$. For most optimality criteria, when calculating $f(x)$, we calculate the Fisher information matrix and invert it to find a variance-covariance matrix, then take some calculation of this final matrix.

- The calculation of the Fisher information matrix from F^TF where X is an $n \times (2m)$ matrix is $O(n^2m)$
- The Fisher information matrix is of size $2m \times 2m$. Inverting the matrix depends on the algorithm used, but is $O((2m)^k)$ where $2 < k \leq 3$.
- Calculating the optimality criterion from the $2m \times 2m$ variance-covariance matrix depends on what criterion is used; the trace (A-optimality) is $O(m)$, taking the determinant (D-optimality) will typically be $O(m^k)$ where $2 < k \leq 3$.

For our designs using A-optimality, we have many more experimental units than treatments ($n >> m$), so the limiting step is the first one and therefore $f(x)$ has computational complexity of $O(n^2m)$.

Complexity of new framework

The complexity of the overhead in the new framework algorithm involves

1. the initial time to find z isomorphisms;
2. the computational cost of checking whether each design is lowest lexicographically amongst all possible isomorphic designs.
   - We apply an isomorphism to reorder each design, each $O(n)$.
   - We do this z times (once for each isomorphism), and then sort the resulting list of isomorphic designs to find the lexicographically smallest, which is $O(z \log z)$ for a good sorting algorithm;
   - Overall cost of step 2 is $O(nz^2 \log z)$.

Thus the computational complexity of checking that a design is valid, and evaluating $f(x)$ if so, is $O(nz^2 \log z + n^2 m/z)$ where the second term is because we calculate $f(x)$ for one design in every z.
The effectiveness of our new method compared to the old depends on the ratio

$$\frac{z^2 \log z + nm/z}{nm}$$

Thus z must be small compared to nm but not too small.

Evaluating blocked designs

We evaluate several blocked experimental structures:

1. 3 blocks of size 3 ($n = 9$), with 3 treatments. The optimal designs are randomised complete block designs;
2. 4 blocks of size 3 ($n = 12$), with i) 3 and ii) 4 treatments. The optimal designs are i) randomised complete block designs and ii) balanced incomplete block designs;
3. A row-column structure with 3 rows and 3 columns, each row-column intersection containing a single experimental unit, with 3 treatments.

   The optimal designs are Latin Squares of size 3:

1	2	3
2	3	1
3	1	2

4. A row-column structure with 4 rows and 4 columns, each row-column intersection containing a single experimental unit, with i)3 and ii)4 treatments. The optimal designs for ii) are Latin Squares of size 4.

Results for blocked designs

Ex.	Description	n	m	Number of isomorphisms	Evaluations without isomorphisms	Evaluations with isomorphisms
1	3x3 Blocks	9	3	1,296	94	2,925
2i	4x3 Blocks	12	3	82,944	379	86,126
2ii	4x3 Blocks	12	4	82,944	1,808	605,960
3	3x3 Row Column	9	3	72	241	2,807
4i	4x4 Row Column	16	3	1,152	34,873	7,123,656
4ii	4x4 Row Column	16	4	1,152	1,610,909	170,863,644

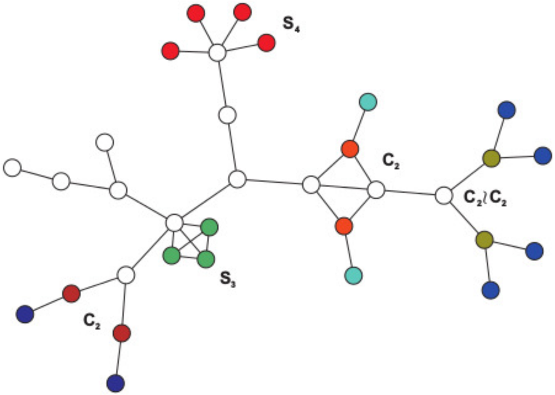
Results for blocked designs

Ex.	Description	n	m	Number of isomorphisms	Time without isomorphisms	Time with isomorphisms
1	3x3 Blocks	9	3	1,296	2.52	1.54
2i	4x3 Blocks	12	3	82,944	55.44	310.02
2ii	4x3 Blocks	12	4	82,944	378.82	1,051.54
3	3x3 Row Column	9	3	72	1.9	0.48
4i	4x4 Row Column	16	3	1,152	6,051.12	493.32
4ii	4x4 Row Column	16	4	1,152	141,456.6	14,123.94

We do not claim that these designs would be sensibly found via this method, as the solutions are known analytically, but we seek to demonstrate the benefits of isomorphisms via improvements in calculations.

Current work - exploiting symmetries

(This last part from upcoming thesis from V. Koutra.)



Example from "Symmetry in Complex Networks", MacArthur,Sánchez-García, Anderson,2008[2].

Conjecture

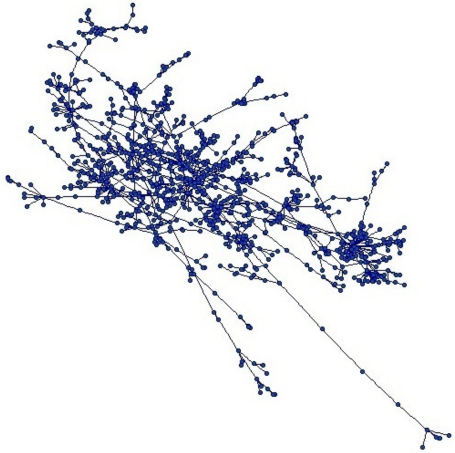
If we

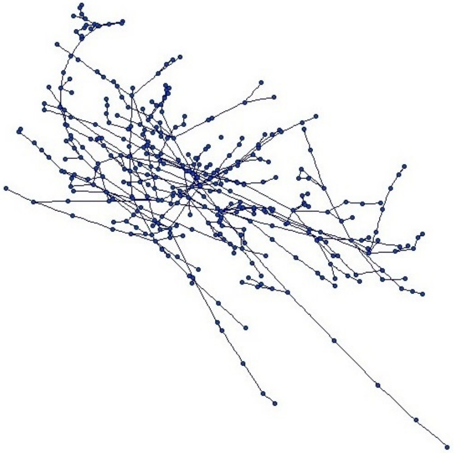
- decompose the network into a sets of nodes which exhibit symmetries and a skeleton (remainder).
- find an optimal design $d*$ for subjects in the skeleton ignoring other nodes
- find an optimal design d for the whole network, with $d*$ fixed on the skeleton.

then d is an optimal design.

The importance of this is that it allows designs on social networks to be found much more quickly.

A social network consisting of 1025 vertices and 1043 edges and is illustrated. The edges indicate the ties among PhD students and their advisors in theoretical computer science (Johnson, 1984).

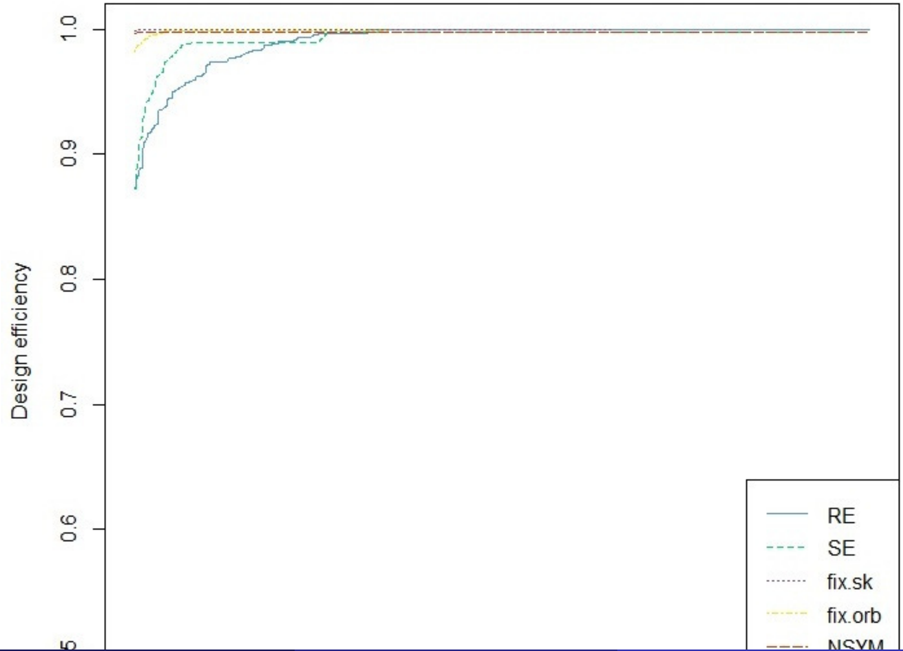




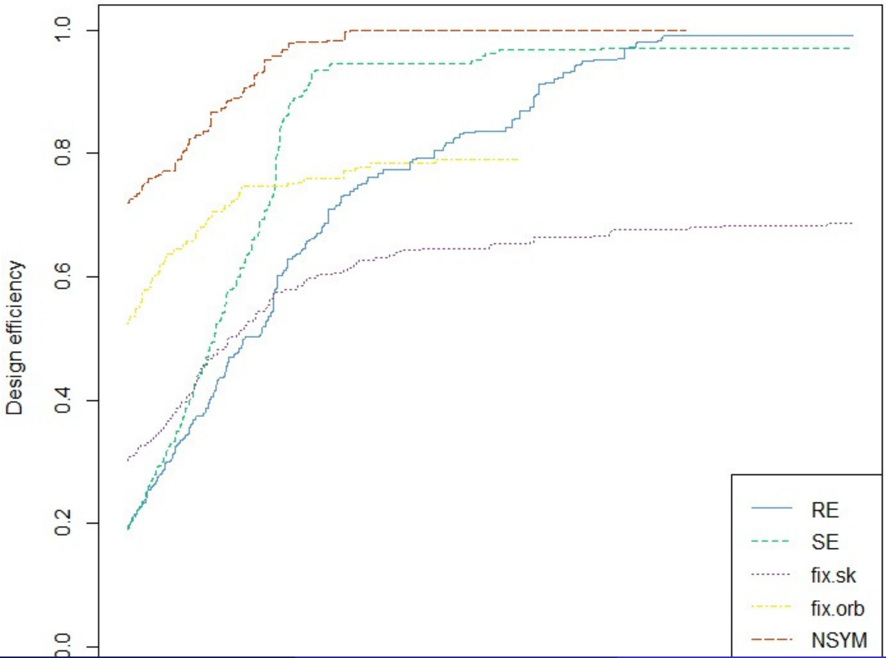
NSYM- an algorithm for faster design?

1. Obtain the graph skeleton
2. Find an (near-) optimal design on the graph skeleton using the systematic exchange algorithm.
3.
    1. Make a different allocation of the treatments under ϕ_1 and ϕ_2.
        - ϕ_1: Have a balanced allocation of the treatments within each vertex orbit, and ultimately achieve an overall balance along all orbits;
        - ϕ_2: Have the same treatment (which will be randomly selected among the different treatments) allocated to all the units within the same vertex orbit.
    2. The design is constructed by fixing the allocation on the skeleton (Step I) as well as on the vertex orbits (Step II). The resulting design is expected to be near-optimal.

ϕ_1



ϕ_2



Conclusions

- Recapped work on linear network models
- Presented a method we can use networks to find designs for experiments that have no overt network form.
- Condsidered a general procedure for finding designs which help us to ignore isomorphisms and remove failed effort.
- Presented some algorithms that seem to be effective in finding near-optimal designs quicker for networked experiments: both experiments on real networks, and experiments where we induce a networked structure.
- A unifying method applicable on many common structures where we perform DOE.
- Software in early phase available to do this readily in R.

References

Luigi Pietro Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento.
An improved algorithm for matching large graphs.
In *3rd IAPR-TC15 workshop on graph-based representations in pattern recognition*, pages 149–159, 2001.

Ben D MacArthur, Rubén J Sánchez-García, and James W Anderson.
Symmetry in complex networks.
Discrete Applied Mathematics, 156(18):3525–3531, 2008.

Ben M. Parker, Steven G. Gilmour, and John Schormans.
Optimal design of experiments on connected units with application to social networks.
Journal of the Royal Statistical Society: Series C (Applied Statistics), 66(3):455–480, April 2016.