

23rd October 2024

Preliminary investigation of algorithms for optimal row-column designs

Balancing Treatment Efficiency and Spatial Distribution in Experimental Designs

Jingning Yao

Mathematical Sciences Institute

supervised by

Dr. Emi Tanaka, A/Prof Katherine Turner



**Australian
National
University**

Master of Mathematical Sciences (Advanced) Thesis

Author:

Jingning Yao

Supervisors:

Dr. Emi Tanaka, A/Prof Katherine T

Project period:

2024-02-17 – 2024-10-24

Mathematical Sciences Institue
Australian National University

Table of contents

List of Figures	ii
List of Tables	iv
Declaration	1
Abstract	2
Notation and terminology	3
1 Introduction	6
2 Background	12
2.1 Linear mixed model	12
2.1.1 Estimation of model parameters	13
2.2 Best linear unbiased estimates and predictions	14
2.2.1 Variance analyse of estimators	16
2.2.2 A-criterion	16
2.3 Neighbor balance and evenness of distribution	19
2.3.1 Concepts of NB and ED	19
2.3.2 Measuring NB and ED	20
3 Methods	22
3.1 Modeling row-column design	22
3.1.1 Random effects matrix	23
3.1.2 Design matrix for treatments	24
3.1.3 Assumptions for A-value calculation	25
3.2 Searching Strategy	26
3.2.1 Existence of the minimum of A-value	27
3.2.2 General structure	27
3.2.3 Permutations and filtering	28
3.2.4 Random search	30
3.2.5 Simulated annealing	31
4 Results	36
4.1 Simulation set-up	36
4.2 Result analyze	38

4.2.1 General behaviour of algorithms	38
4.2.2 Analysis by Algorithm	45
4.2.3 Comparison Between Algorithms	54
4.2.4 Conclusion	62
5 Discussion	63
5.1 Limitations	63
5.1.1 Computational efficiency and convergence rate	63
5.1.2 Balancing multi-objective optimization	63
5.1.3 Limitations and generalizability in practical applications	64
5.2 Future Directions	64
5.2.1 Penalty objective function	64
5.2.2 Algorithm improvment	65
References	67
Appendices	69
A Tools	69

List of Figures

1.1	A 3×3 Latin square design.	7
1.2	Above, the figure presents an example of clustering: multiple replications of treatment 8 are grouped together, while treatments 1 and 19 are always adjacent to each other.	10
3.1	Method been used	28
3.2	Attempting method	28
4.1	Detailed assignment for treatments given by design function in blocks-design package	38
4.2	A v.s. Iteration (Random search)	39
4.3	Detailed assignment for treatments given by random search	40
4.4	A v.s. Iteration (SA with log cooling schedule)	41
4.5	Detailed assignment for treatments given by SA with log cooling schedule	42
4.6	A v.s. Iteration (SA with exp cooling schedule)	43
4.7	Detailed assignment for treatments given by SA with exp cooling schedule	44
4.8	Proportion of Iteration number for random selection	46
4.9	The above plot shows the average runtime by the number of rows with the color indicating the number of columns and faceted by the number of treatments t for Random Selection algorithm.	47
4.10	The above plot shows the average runtime by the number of treatments with the color indicating the number of rows, faceted by the number of columns for Random Selection algorithm.	48
4.11	The above plot shows the difference between A_{SAlog} and A_{op} by the number of rows with the color indicating the number of columns and faceted by the number of treatments for SA log Cooling algorithm. . .	50
4.12	The above plot shows the difference between A_{SAlog} and A_{op} by the number of treatments, with the color indicating the number of rows and faceted by the number of columns for SA with log cooling algorithm. . .	51
4.13	Proportion of Iteration number for SA with exp cooling schedule	52
4.14	The above plot shows the runtime by the number of rows with the color indicating the number of columns and faceted by the number of treatments for SA with exp cooling algorithm.	53

4.15	The above plot shows the runtime by the number of treatments with the color indicating the number of rows and faceted by the number of columns for the filtered ‘SA exp Cooling’ algorithm.	54
4.16	The average relative difference between A_{op} and A_{result} by the number of rows for different algorithms, with the lines representing different algorithms.	55
4.17	The average relative difference between A_{op} and A_{result} by the number of columns for different algorithms, with the lines representing different algorithms.	56
4.18	The average relative difference between A_{op} and A_{result} by the number of treatments for different algorithms, with the lines representing different algorithms.	57
4.19	The average absolute difference between A_{op} and A_{result} by the number of treatments for different algorithms, with the lines representing different algorithms.	58
4.20	the average runtime by the number of rows for different algorithms, with the lines representing different algorithms.	59
4.21	The average runtime by the number of columns for different algorithms, with the lines representing different algorithms.	60
4.22	The average runtime by the number of treatments for different algorithms, with the lines representing different algorithms.	61

List of Tables

1.1	The design elements of Figure 1.1 with Ω and T as defined in Definition 1.1 and F_R and F_C as defined in Definition 1.2. The $n = 9$ -vectors \mathbf{x}_{R_i} for $i = 1, 2, 3$, \mathbf{x}_{C_j} for $j = 1, 2, 3$ and \mathbf{x}_{T_k} for $k = 1, 2, 3$ are the orthonormal basis of V_R , V_C and V_T , respectively.	7
1.1	The design elements of Figure 1.1 with Ω and T as defined in Definition 1.1 and F_R and F_C as defined in Definition 1.2. The $n = 9$ -vectors \mathbf{x}_{R_i} for $i = 1, 2, 3$, \mathbf{x}_{C_j} for $j = 1, 2, 3$ and \mathbf{x}_{T_k} for $k = 1, 2, 3$ are the orthonormal basis of V_R , V_C and V_T , respectively.	8

Declaration

The work in this thesis is my own except where otherwise stated.

Jingning yao

Abstract

Row-Column designs are widely used in experimental design to minimize variance and improve estimation accuracy. However, it sometimes can lead to clustering of treatments, which negatively impacts design efficiency. we present an approach to optimizing row-column designs using the A-criterion, while avoiding clustering simultaneously through the application of random search and simulated annealing algorithm. Our method introduces some statistical criteria to quantify spatial distribution, and by incorporating these metrics into the two mentioned algorithms, we effectively balance the optimization between treatment efficiency and spatial distribution. Additionally, we compare the performance of the different algorithms, demonstrating their relative effectiveness. This work contributes to both the theory and practice of experimental design by offering an improved methodology for achieving optimal and evenly distributed arrangements.

Notation and terminology

Notation	Meaning
\mathbf{y}	Response variable vector
\mathbf{u}	Parameter vector of blocking effects
$\hat{\mathbf{u}}$	Estimated parameter vector of blocking effects by log-likelihood maximization
$\boldsymbol{\tau}$	Parameter vector of treatments effects
$\hat{\boldsymbol{\tau}}$	Estimated parameter vector of treatments effects by log-likelihood maximization
$\boldsymbol{\epsilon}$	Parameter vector of model error
\mathbf{Z}	Blocking factor matrix
\mathbf{X}	Design matrix for treatments
\mathbf{G}	Variance matrix for random effect
\mathbf{R}	variance matrix for model error
\mathcal{A}	The A-value
$\boldsymbol{\Lambda}$	Variance-covariance of estimated treatments effects
∞_n	A vector of length n with all elements equal to 1
\mathbf{D}	Transformation matrix of treatments
\mathcal{D}	Design results
$\mathcal{D}_{r,c}$	The treatment at row r and column c
T	The set of treatments
\mathcal{U}	Adjacency matrix
n_r	Total number of rows
n_c	Total number of columns
n_τ	Total kinds of treatments
C_{NB}	Neighbour balance criteria
$MRS(\mathcal{D})$	The minimum row span of design \mathcal{D}
$MCS(\mathcal{D})$	The minimum column span of design \mathcal{D}
\mathbf{G}^{diag}	Diagonal \mathbf{G} matrix
\mathbf{R}^{diag}	Diagonal \mathbf{R} matrix

Notation	Meaning
I_n	Identity matrix with size $n \times n$
σ_{G_c}	Scaling constant for column effects
σ_{G_r}	Scaling constant for row effects
σ_R	Scaling constant for model error
$A(\mathbf{X})$	A-value function maps design matrix \mathbf{X} to its A-value
\mathbf{X}_{new}	New design matrix generated by permutation
\mathcal{D}_{new}	Design results corresponding to \mathbf{X}_{new}
\mathbf{P}_{ij}	An identity matrix with i -th row and j -th row swapped
C_{NB}	Neighbour balance criteria of \mathcal{D}_{new}
T_{NB}	Tolerance for Neighbour balance criteria
T_{ED}	Tolerance for minimum row span and minimum column span
s	Step length for random search
t_{NB}	Tolerance for Neighbour balance criteria for pairwise permutation
t_{ED}	Tolerance for minimum row span and minimum column span for pairwise permutation
M	Maximum iteration number
$\{\mathbf{X}\}$	Finite design searching space
$\{\mathbf{X}^*\}$	A set of design matrix having optimal A-value
$T(t)$	Temperature at t -th iteration
\mathbf{X}_0	Initial design matrix
n_p	Number of permutations
\mathbf{G}_s	Variance matrix for random effect of simulation
\mathbf{R}_s	variance matrix for model error of simulation
$A_{blocksdesign}$	A-value calculated by design function in blocksdesign package
T_A	Tolerance for A-value in simulation
A_Δ	Difference of A-values between two design matrix
A_{op}	A-value calculated by design function in blocksdesign package in example case
A_{rs}	A-value result given by random search
\mathcal{D}_{rs}	Design result of random search
C_{SAlog}	Neighbour balance criteria of random search
A_{SAlog}	A-value result given by SA with log cooling schedule
\mathcal{D}_{SAlog}	Design result of SA with log cooling schedule

Notation	Meaning
C_{SAlog}	Neighbour balance criteria of SA with log cooling schedule
A_{SAexp}	A-value result given by SA with exp cooling schedule
\mathcal{D}_{SAexp}	Design result of SA with exp cooling schedule
C_{SAexp}	Neighbour balance criteria of SA with exp cooling schedule
Terminology	Meaning
Evenness of distribution (ED)	Multiple replications of a treatment are evenly spread across the design field
Neighbour balance (NB)	Multiple treatments are not consistently placed next to each other in the design
Pairwise permutation	Swapping two treatments in a design matrix to explore new configurations

Chapter 1

Introduction

Experiments play a vital role in scientific research by helping to establish causal relationships between variables. To achieve accurate and reliable results in a comparative experiment, careful experimental design is essential. In this thesis, we formally define an experimental design as given in Definition 1.1.

Definition 1.1 (Design). A comparative experiment is an experimental study involving a set of experimental units, denoted as Ω , and a set of treatments, denoted as \mathcal{T} . The experimental design is defined as a surjective function $D : \Omega \rightarrow \mathcal{T}$. The sizes of Ω and \mathcal{T} are denoted as N and n_t , respectively. We use $V_\Omega := \mathbb{R}^N$ and $V_{\mathcal{T}}$ to denote the vector spaces of Ω and \mathcal{T} , respectively.

In conducting an experiment, a proper design of experiment is crucial for ensuring accurate and reliable results. A row-column design is a particular type of experimental design where the experimental unit is defined by the combination of two orthogonal factors, typically arranged in rows and columns. The row-column experimental structure is common in many biological applications, e.g. 96-well plates and crop field trials. The treatments of interest are then randomly allocated with certain constraints as not to confound the treatments with nuisance factors. See Definition 1.2 for a formal definition of a row-column design. Note that rows and columns do not have to correspond to actual physical dimensions.

Generally, a row-column design can be defined as

Definition 1.2 (Row-column design). A row-column design refers to an experiment where experimental units are defined by the Cartesian product of two sets with sizes n_r and n_c , denoted as \mathcal{R} and \mathcal{C} , respectively, such that $\Omega = \mathcal{R} \times \mathcal{C}$. We denote F_R and F_C as the surjective functions that map Ω to \mathcal{R} and \mathcal{C} , respectively. We use V_R and V_C to denote vector spaces in \mathbb{R}^N for row and column factors, respectively.

In a row-column design, the row and column factors are often considered as nuisance (or blocking) factors. The treatments of interest are then randomly allocated with certain constraints as to reduce confounding the treatment factors with these nuisance factors. An example of row-column design is the *Latin square design* where $n_r = n_c = n_t$ and each treatment appears exactly once in each row and column (see Example 1.1).

Example 1.1 (A 3×3 Latin square design). In a 3×3 Latin square design, we let $n_t = n_c = n_r = 3$ and the resulting random instance of a design is as shown in Figure 1.1 with its corresponding design elements shown in Table 1.1.

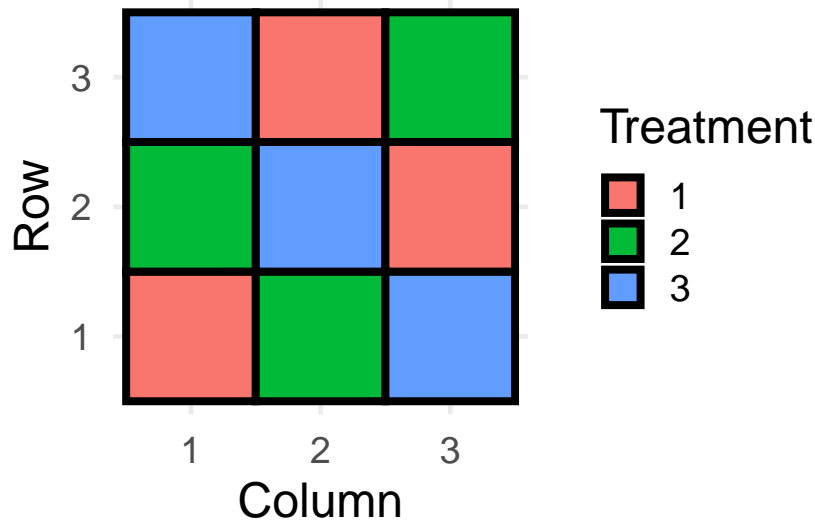


Figure 1.1: A 3×3 Latin square design.

Table 1.1: The design elements of Figure 1.1 with Ω and T as defined in Definition 1.1 and F_R and F_C as defined in Definition 1.2. The $n = 9$ -vectors \mathbf{x}_{R_i} for $i = 1, 2, 3$, \mathbf{x}_{C_j} for $j = 1, 2, 3$ and \mathbf{x}_{T_k} for $k = 1, 2, 3$ are the orthonormal basis of V_R , V_C and V_T , respectively.

$\omega \in$												
Ω	$F_R(\omega)$	$F_C(\omega)$	$T(\omega)$	\mathbf{x}_{R_1}	\mathbf{x}_{R_2}	\mathbf{x}_{R_3}	\mathbf{x}_{C_1}	\mathbf{x}_{C_2}	\mathbf{x}_{C_3}	\mathbf{x}_{T_1}	\mathbf{x}_{T_2}	\mathbf{x}_{T_3}
1	1	1	1	1	0	0	1	0	0	1	0	0
2	1	2	2	1	0	0	0	1	0	0	1	0
3	1	3	3	1	0	0	0	0	1	0	0	1
4	2	1	2	0	1	0	1	0	0	0	1	0
5	2	2	3	0	1	0	0	1	0	0	0	1
6	2	3	1	0	1	0	0	0	1	1	0	0
7	3	1	3	0	0	1	1	0	0	0	0	1
8	3	2	1	0	0	1	0	1	0	1	0	0

Table 1.1: The design elements of Figure 1.1 with Ω and T as defined in Definition 1.1 and F_R and F_C as defined in Definition 1.2. The $n = 9$ -vectors \mathbf{x}_{R_i} for $i = 1, 2, 3$, \mathbf{x}_{C_j} for $j = 1, 2, 3$ and \mathbf{x}_{T_k} for $k = 1, 2, 3$ are the orthonormal basis of V_R , V_C and V_T , respectively.

$\omega \in$												
Ω	$F_R(\omega)$	$F_C(\omega)$	$T(\omega)$	\mathbf{x}_{R_1}	\mathbf{x}_{R_2}	\mathbf{x}_{R_3}	\mathbf{x}_{C_1}	\mathbf{x}_{C_2}	\mathbf{x}_{C_3}	\mathbf{x}_{T_1}	\mathbf{x}_{T_2}	\mathbf{x}_{T_3}
9	3	3	2	0	0	1	0	0	1	0	1	0

Table 1.1 shows how we use a binary method to distinguish between rows, columns, and types of treatments. For row-column designs, we typically use a design matrix to model the arrangement of treatments.

Definition 1.3. A design matrix \mathbf{X} is a matrix that represents the arrangement of experimental units across different treatments in an experimental design. Each row corresponds to an experimental unit, and each column corresponds a treatment level.

The last three columns of the Table 1.1 above can be viewed as the design matrix for Example 1.1, which we will discuss it in detail in the Chapter 3.

In this thesis, the observational unit and experimental unit are considered the same, thus the response of interest for the $\omega \in \Omega$ is a random variable denoted as Y_ω and its observed value (after the experiment) denoted as y_ω . We use $\mathbf{y} = (y_1, \dots, y_n)^\top$ to denote the vector of responses and use this interchangeably to denote a vector of random variable as well. In its simplest form, we assume that the response can be decomposed to the treatment and nuisance factors by assuming a linear model,

$$Y_\omega = \tau_{T(\omega)} + e_\omega, \quad (1.1)$$

where $\tau_{T(\omega)}$ is a constant value with respect to the treatment received for ω (referred to as the *average treatment effects*) and e_ω is a random variable associated with the remaining contribution from ω . Equation 1.1 assumes an additive structure which is fundamental in estimating the average treatment effect – if this assumption does not hold to a large extent, there is little point in conducting the experiment.

Different treatments usually have varying effects on the experimental outcomes, known as treatment effects. The goal of our experimental design is to more accurately measure and estimate the impact of different treatments on the experimental results, i.e., the treatment effects. To offer precise estimation of treatment effects, one way is to seek the optimal value of some statistic criteria, for exam-

ple, A-criteria, minimizing the variance of elementary treatment contrasts. Butler [2013] has show the relation between optimizing design and minimizing the value of A-criteria, and show some possible algorithms to search optimal design in feasible set. These algorithms mainly focus on comparing the arrangements of different treatments, that is, doing permutations, and calculating their A values, optimizing design by iterations.

However, some undesired cluster of replications or some treatment may occur when algorithm are doing permutations along rows and columns. In an experimental design, a treatment usually has multiple replications. Specifically, clustering can refer to multiple replications of the same treatment being grouped together, or different treatments' replications frequently being grouped together in the experimental design see an example in Figure 1.2. Piepho et al. [2018] found that such clustering is considered undesirable by experimenters who worry that irregular environmental gradients might negatively impact multiple replications of the same treatment, potentially leading to biased treatment effect estimates. Williams emphasis that there is a need to design a strategy to avoid clustering and achieve even distribution of treatment replications among the experimental field. Two properties of design are introduced. Evenness of distribution of treatment replications, abbreviated as ED, and neighbour balance, abbreviated as NB. Evenness of distribution represents how multiple replications of a treatment are evenly spread across the design field, rather than being placed close to each other. Neighbour balance, on the other hand, indicates that multiple treatments are not consistently placed next to each other in the design, promoting a more diverse adjacency relationship between treatments. A good ED ensures every replications of a treatment are widely spread in experimental field, and NB helps to avoid replications of the some treatment cluster together repeatedly. Williams introduce a scoring system to analysis ED and NB for a specific design, and introduce a algorithm to optimize ED, NB and some average efficiency factor can be represented by a specific statistic criteria.

We offer an optimization strategy for a design problem, which we can improving ED and NB during optimizing statistic criteria for a design, and avoid unwanted clustering and self-adjacency on the resulting design. In this algorithm, we use A-criteria to evaluate the efficiency of a design. Before the algorithm, we randomly generate a design as an initial design, and calculate the A-criteria as initial value. We update design by selecting a better among its neighbours. The neighbours are pair-wise permutations of a design. Typically, we select a neighbour from all pair-wise permutations of a design for iteration, but this does not ensure ED and NB. To ensure ED and NB during optimization, we add some constraints when generating the pairwise permutations, only maintain or better properties are accepted. By fil-

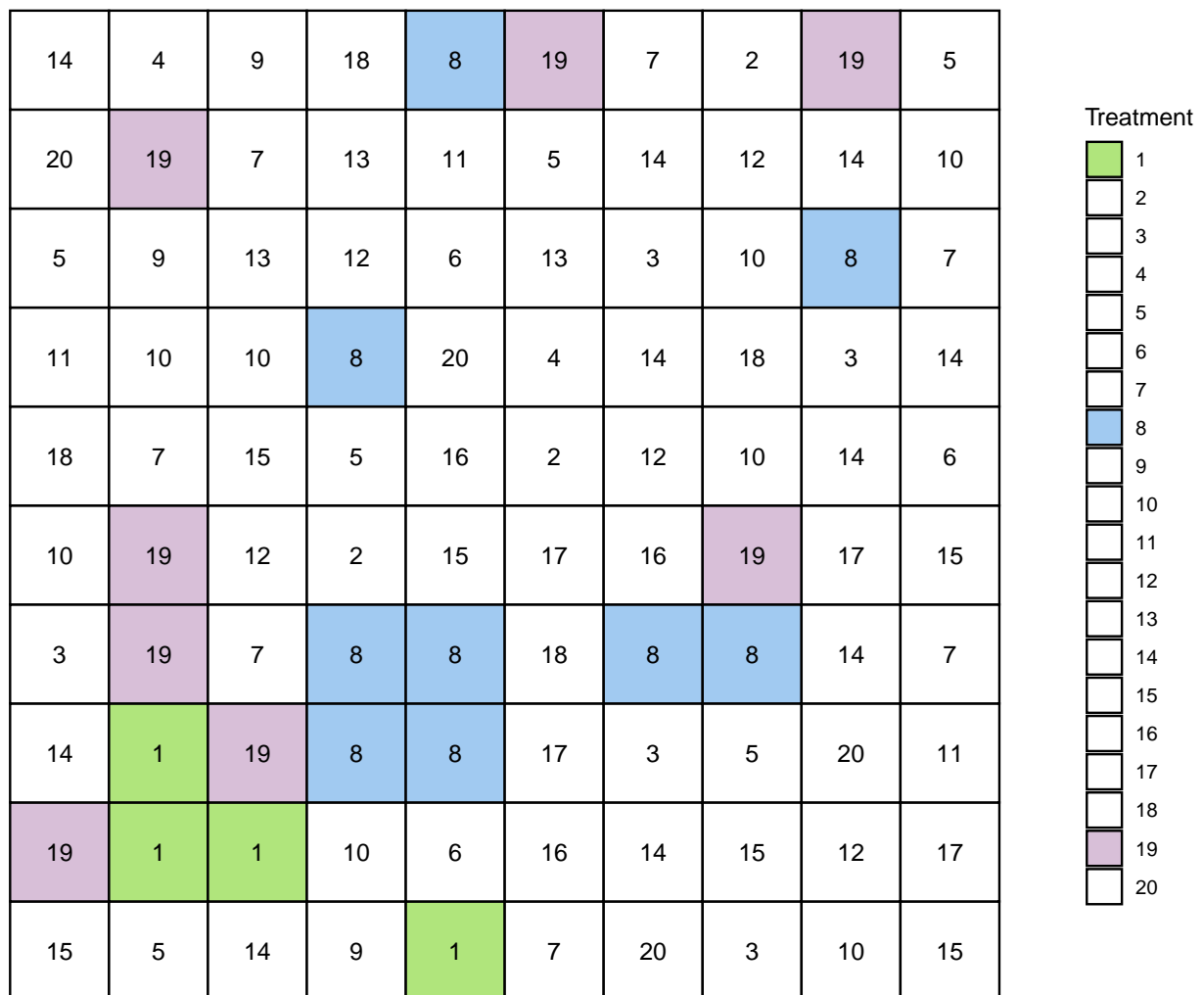


Figure 1.2: Above, the figure presents an example of clustering: multiple replications of treatment 8 are grouped together, while treatments 1 and 19 are always adjacent to each other.

tering design with bad ED and NB, we optimize the statistic criteria such as A-value of the design.

In the Chapter 2, we introduced how to model row-column designs by linear mixed, introduced the methods for calculating various statistics and the meaning behind them, and provided relevant mathematical proofs. In the Chapter 3, we outlined the relevant assumptions made for row-column designs. We introduced three search algorithms—random search and two cooling schedules for simulated annealing—providing detailed descriptions of their steps and discussing their convergence properties. In the Chapter 4, we first provide details of our simulation set-up, presenting specific results with various figures, analyse the performance of different algorithms, and made comparisons among them. In the Chapter 5, we outline our limitations and discussed some potential future directions.

Chapter 2

Background

Linear (mixed) models are the primary method of analysis of experimental data with response that have an approximate Normal distribution. In this chapter, we expand on the model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\tau} + \boldsymbol{\epsilon}$$

to consider linear mixed models for the analysis of experimental data with a more general experimental structure.

2.1 Linear mixed model

Linear mixed model extends linear model by incorporating additionally incorporating random effects into the model that effectively give greater flexibility and capability to incorporate known correlated structures into the model. We now consider a linear mixed model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\tau} + \mathbf{Z}\mathbf{u} + \boldsymbol{\epsilon} \quad (2.1)$$

here \mathbf{y} is $n \times 1$ vector for n observations, $\boldsymbol{\tau}$ is a $t \times 1$ parameter vector of treatment factors, \mathbf{u} is a $q \times 1$ parameter vector of blocking effects, and $\boldsymbol{\epsilon}$ is the $n \times 1$ error vector, \mathbf{X} and \mathbf{Z} are design matrices of dimension $n \times t$ and $n \times q$ for treatment factors and blocking factors, respectively. We here assume blocking factors are random effect, with random error $\boldsymbol{\epsilon}$ we have

$$\begin{bmatrix} \mathbf{u} \\ \boldsymbol{\epsilon} \end{bmatrix} \sim N \left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{G} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} \right) \quad (2.2)$$

where \mathbf{G} is the $q \times q$ variance matrix for \mathbf{u} and \mathbf{R} is $n \times n$ variance matrix for $\boldsymbol{\epsilon}$.

Throughout this thesis, we assume $\mathbf{R} = \sigma^2 \mathbf{I}_n$ and \mathbf{G} is a block diagonal matrix

where each block is a scaled identity matrix. In practice, these structures can be more complex, e.g. Williams and John consider autoregressive and linear variance structures for crossover designs.

In addition, we assume in this thesis that the treatment effects are fixed and blocking factors (i.e. the row and column factors in row-column design) are random effects. It should be noted, however, that the treatment effects can be random effects based on the purpose of the experiment [Robinson, Butler et al.].

2.1.1 Estimation of model parameters

Fixed and random effects

Henderson et al. first derived the set of equations, referred to as *mixed model equations* by Smith, as a precursor to the estimation of fixed effects and predictions of random effects. These equations are derived from maximizing the joint log density of the response and random effects as shown in Lemma 2.1.

Lemma 2.1 (Mixed model equations). *The mixed model equations shown below are derived from maximising the joint log density of \mathbf{y} and \mathbf{u} .*

$$C\hat{\beta} = WR^{-1}\mathbf{y},$$

$$\text{where } C = W^{\top}R^{-1}W + G^*, \quad W = \begin{bmatrix} X & Z \end{bmatrix}, \quad \hat{\beta} = (\hat{\tau}^{\top}, \hat{\mathbf{u}}^{\top})^{\top} \text{ and } G^* = \begin{bmatrix} \mathbf{0}_{p \times p} & \mathbf{0}_{p \times q} \\ \mathbf{0}_{q \times p} & G^{-1} \end{bmatrix}.$$

Proof. Given the model 2.1 with assumption in 2.2, we have $\mathbf{u} \sim N(\mathbf{0}, G)$ and $\mathbf{y}|\mathbf{u} \sim N(X\tau + Z\mathbf{u}, R)$ with probability density functions as

$$f_{\mathbf{u}}(\mathbf{u}) = \frac{1}{\sqrt{(2\pi)^q |G|}} \exp\left(-\frac{1}{2}\mathbf{u}^{\top}G^{-1}\mathbf{u}\right)$$

$$f_{\mathbf{y}|\mathbf{u}}(\mathbf{y}|\mathbf{u}) = \frac{1}{\sqrt{(2\pi)^n |R|}} \exp\left(-\frac{1}{2}(\mathbf{y} - (X\tau + Z\mathbf{u}))^{\top}R^{-1}(\mathbf{y} - (X\tau + Z\mathbf{u}))\right).$$

Ignoring the constants, we have

$$\log f_{\mathbf{u}}(\mathbf{u}) = -\frac{1}{2}(\log |G| + \mathbf{u}^{\top}G^{-1}\mathbf{u})$$

$$\log f_{\mathbf{y}|\mathbf{u}}(\mathbf{y}|\mathbf{u}) = -\frac{1}{2}(\log |R| + (\mathbf{y} - X\tau - Z\mathbf{u})^{\top}R^{-1}(\mathbf{y} - X\tau - Z\mathbf{u})).$$

The joint density of \mathbf{y} and \mathbf{u} is given as

$$f_{y,u}(\mathbf{y}, \mathbf{u}) = f_{y|u}(\mathbf{y}|\mathbf{u})f_u(\mathbf{u}), \quad (2.3)$$

and taking the log of both sides for Equation 2.3, we have the log of joint density as:

$$\begin{aligned} \mathcal{L} &:= \log f_{y,u}(\mathbf{y}, \mathbf{u}) \\ &= \log f_{y|u}(\mathbf{y}|\mathbf{u}) + \log f_u(\mathbf{u}) \\ &= -\frac{1}{2} \left(\log |\mathbf{R}| + \log |\mathbf{G}| + (\mathbf{y} - \mathbf{X}\boldsymbol{\tau} - \mathbf{Z}\mathbf{u})^\top \mathbf{R}^{-1} (\mathbf{y} - \mathbf{X}\boldsymbol{\tau} - \mathbf{Z}\mathbf{u}) + \mathbf{u}^\top \mathbf{G}^{-1} \mathbf{u} \right). \end{aligned}$$

We determine that $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\tau}} = \frac{\partial \mathcal{L}}{\partial \mathbf{u}} = \mathbf{0}$, and write the equation into a matrix form

$$\begin{bmatrix} \mathbf{X}^\top \mathbf{R}^{-1} \mathbf{X} & \mathbf{X}^\top \mathbf{R}^{-1} \mathbf{Z} \\ \mathbf{Z}^\top \mathbf{R}^{-1} \mathbf{X} & \mathbf{Z}^\top \mathbf{R}^{-1} \mathbf{Z} + \mathbf{G}^{-1} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\tau}} \\ \hat{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}^\top \mathbf{R}^{-1} \mathbf{y} \\ \mathbf{Z}^\top \mathbf{R}^{-1} \mathbf{y} \end{bmatrix}. \quad (2.4)$$

$$\Rightarrow (\mathbf{W}^\top \mathbf{R}^{-1} \mathbf{W} + \mathbf{G}^*) \hat{\boldsymbol{\beta}} = \mathbf{W} \mathbf{R}^{-1} \mathbf{y}$$

Substituting the above \mathbf{C} leads to our conclusion □

?@thm-blue provides the specific form for estimation of the parameters. :::
{#thm-blue}

2.2 Best linear unbiased estimates and predictions

The solutions to the mixed models equations in Lemma 2.1 (shown below) give the best linear unbiased estimates (BLUEs) for $\boldsymbol{\tau}$ and best linear unbiased predictions (BLUPs) for \mathbf{u} provided that \mathbf{G} and \mathbf{R} are known.

$$\begin{aligned} \hat{\boldsymbol{\tau}} &= (\mathbf{X}^\top \mathbf{V}^{-1} \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{V}^{-1} \mathbf{y} \\ \hat{\mathbf{u}} &= \mathbf{G} \mathbf{Z}^\top \mathbf{V}^{-1} (\mathbf{y} - \mathbf{X} \hat{\boldsymbol{\tau}}), \end{aligned} \quad (2.5)$$

where $\mathbf{V} = \mathbf{Z} \mathbf{G} \mathbf{Z}^\top + \mathbf{R}$. In this paper, we primarily discuss the treatment effect, so we provide the proof for $\hat{\boldsymbol{\tau}}$.

Proof. By cancelling \mathbf{u} , we have

$$\begin{aligned} \mathbf{X}^\top \mathbf{R}^{-1} \mathbf{X} \boldsymbol{\tau} + \mathbf{X}^\top \mathbf{R}^{-1} \mathbf{Z} (\mathbf{Z}^\top \mathbf{R}^{-1} \mathbf{Z} + \mathbf{G}^{-1}) \mathbf{Z}^\top \mathbf{R}^{-1} \mathbf{y} &= \mathbf{X}^\top \mathbf{R}^{-1} \mathbf{y} \\ \Rightarrow \mathbf{X}^\top [\mathbf{R}^{-1} - \mathbf{R}^{-1} \mathbf{Z} (\mathbf{Z}^\top \mathbf{R}^{-1} \mathbf{Z} + \mathbf{G}^{-1})^{-1} \mathbf{Z}^\top \mathbf{R}^{-1}] \mathbf{X} \boldsymbol{\tau} &= \mathbf{X}^\top [\mathbf{R}^{-1} - \mathbf{R}^{-1} \mathbf{Z} (\mathbf{Z}^\top \mathbf{R}^{-1} \mathbf{Z} + \mathbf{G}^{-1})^{-1} \mathbf{Z}^\top \mathbf{R}^{-1}] \mathbf{y} \end{aligned}$$

$$\Rightarrow X^\top P X \tau = X^\top P y$$

where $P = R^{-1} - R^{-1}Z(Z^\top R^{-1}Z + G^{-1})^{-1}Z^\top R^{-1}$, let $C_{11} = X^\top P X$ then we have the form of estimation for $\hat{\tau}$, which is

$$C_{11}\hat{\tau} = X^\top P y$$

and the estimation of τ is $\hat{\tau} = C_{11}^{-1}X^\top P y$, Now, we only need to prove that $P = V^{-1}$.

$$\begin{aligned} PV &= (R^{-1} - R^{-1}Z(Z^\top R^{-1}Z + G^{-1})^{-1}Z^\top R^{-1})(R + ZGZ^\top) \\ &= I + R^{-1}ZGZ^\top - R^{-1}Z(Z^\top R^{-1}Z + G^{-1})^{-1}Z^\top \\ &\quad - R^{-1}Z(Z^\top R^{-1}Z + G^{-1})^{-1}Z^\top R^{-1}ZGZ^\top \\ &= I + R^{-1}ZGZ^\top - R^{-1}Z(Z^\top R^{-1}Z + G^{-1})^{-1}Z^\top(I + R^{-1}ZGZ^\top) \\ &= I + R^{-1}ZGZ^\top - R^{-1}Z(Z^\top R^{-1}Z + G^{-1})^{-1}(Z^\top + Z^\top R^{-1}ZGZ^\top) \\ &= I + R^{-1}ZGZ^\top - R^{-1}Z(Z^\top R^{-1}Z + G^{-1})^{-1}(I + Z^\top R^{-1}ZG)Z^\top \\ &= I + R^{-1}ZGZ^\top - R^{-1}Z(Z^\top R^{-1}Z + G^{-1})^{-1}(G^{-1} + Z^\top R^{-1}Z)GZ^\top \\ &= I + R^{-1}ZGZ^\top - R^{-1}ZGZ^\top \\ &= I \end{aligned}$$

so we have $\hat{\tau} = C_{11}^{-1}X^\top P y = \hat{\tau} = (X^\top V^{-1}X)^{-1}X^\top V^{-1}y$. □

...

Variance parameters

In practice, the variance parameters for G and R are unknown and must be estimated from data. As a consequence, the solutions in **Thm-1** are referred to as Empirical-BLUEs (E-BLUE) or Empirical-BLUPs (E-BLUPs).

The variance parameters may be estimated using a method of moment approach, however this approach can only be utilized for simple variance structures with balanced numbers of treatments.

The modern prevalent approach to estimating variance parameters use maximum likelihood estimates (MLEs) or residual maximum likelihood (REML) estimates [Patterson and Thompson]. The latter approach takes into account the loss of degrees of freedom to the fixed effects so the estimates of variance are less biased. Interested readers should see Verbyla for the derivation of REML and Smith for computational strategies involving the average information algorithm [Gilmour et al., Johnson and

Thompson].

2.2.1 Variance analyse of estimators

In linear mixed model analysis, when estimating the accuracy of parameters and variance, we typically use the Fisher information matrix to provide a method for variance analysis. For the linear mixed model Equation 2.1, we have the log-likelihood of it is then given as:

$$\log \ell(\boldsymbol{\tau}; \mathbf{y}) = -\frac{n}{2} \log(2\pi) - n \log(\sigma) - \frac{1}{2\sigma^2} (\mathbf{y} - \mathbf{X}\boldsymbol{\tau})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\tau}).$$

The (i, j) -th entry of the Fisher information matrix is defined as

$$I_{ij}(\boldsymbol{\tau}) = -\mathbb{E} \left(\frac{\partial^2}{\partial \tau_i \partial \tau_j} \log \ell(\boldsymbol{\tau}; \mathbf{y}) \right)$$

where τ_i is the i -th entry of $\boldsymbol{\tau}$.

Lemma 2.2. *The Fisher information matrix of Equation 1.1 for $\boldsymbol{\tau}$ is given as*

$$\mathbf{C}_{\text{Fisher}} = -\mathbb{E} \left(\frac{\partial^2}{\partial \boldsymbol{\tau} \partial \boldsymbol{\tau}^\top} \log \ell(\boldsymbol{\tau}; \mathbf{y}) \right)$$

And we have $\boldsymbol{\tau} \sim N(\hat{\boldsymbol{\tau}}, \mathbf{C}_{\text{Fisher}}^{-1})$. We will provide the specific form of this matrix and the related proof in the next section.

2.2.2 A-criterion

Now we have our estimation for the treatment factor, and experimental design aims to further refine our design by focusing on the precision of these estimates. Specifically, we aim to optimize the design so that the treatment effects are estimated with minimal variance, ensuring that the differences between any two treatment levels are as small as possible. To achieve this, we introduce the A-value as a criterion for evaluating the design.

Optimizing the A-value is crucial in row-column designs, for it directly relates to the precision of the treatment effect estimates. The A-value, a measure of design efficiency, quantifies how well the experimental design minimizes the variability when estimating treatment effects. By focusing on minimizing the A-value, we aim to achieve a design that provides the most precise estimates of the difference of the effects between two treatments. A lower A-value means that the design is more

efficient, leading to smaller variances for the difference between treatment effect estimations.

Recall that in this thesis, we consider treatment factors τ are fixed, to elucidate the influence of A-criterion. Our discussion of the A-criterion below is always based on this assumption. \therefore {#def-A-value} Basing on the model formula Equation 2.1, and a estimation of treatment factor $\hat{\tau}$ has n_τ factors. A-criterion measure the average predicted error variance of different treatments. Let $V_{ij} = \text{var}(\hat{\tau}_i - \hat{\tau}_j) = \text{var}(\hat{\tau}_i) + \text{var}(\hat{\tau}_j) - 2\text{cov}(\hat{\tau}_i, \hat{\tau}_j)$, and an A-value \mathcal{A} is

$$\mathcal{A} = \frac{1}{n_\tau(n_\tau - 1)} \sum_i \sum_{j < i} V_{ij} \quad (2.6)$$

\therefore

To discover the relationship between the A-value and the design matrix, I need to find the variance-covariance matrix of $\hat{\tau}$. In fact, it can be proof that

Lemma 2.3. *For treatment effect and its estimator Equation 2.5, we have $(\tau - \hat{\tau}) \sim N(0, \mathbf{C}_{11}^{-1})$.*

Proof. From Equation 2.1 and basing on assumption Equation 2.2 we have

$$\mathbf{y} = \mathbf{X}\boldsymbol{\tau} + \mathbf{Z}\mathbf{u} + \boldsymbol{\epsilon} \sim N(\mathbf{X}\boldsymbol{\tau}, \mathbf{R} + \mathbf{Z}\mathbf{G}\mathbf{Z}^\top), \quad (2.7)$$

so also we have $\hat{\tau} \sim N(\boldsymbol{\tau}, \mathbf{X}^\top \mathbf{R}^{-1}(\mathbf{R} + \mathbf{Z}\mathbf{G}\mathbf{Z}^\top)(\mathbf{X}^\top \mathbf{R}^{-1})^\top)$

We denote

$$\begin{aligned} \mathbf{M} &= \mathbf{X}^\top \mathbf{R}^{-1} \mathbf{X} \\ \mathbf{N} &= \mathbf{X}^\top \mathbf{R}^{-1} \mathbf{Z} \\ \mathbf{J} &= \mathbf{Z}^\top \mathbf{R}^{-1} \mathbf{X} \\ \mathbf{K} &= \mathbf{Z}^\top \mathbf{R}^{-1} \mathbf{Z} + \mathbf{G}^{-1} \end{aligned}$$

In the context of row-column design, the \mathbf{K} matrix is invertible. Schur complement of \mathbf{K} is

$$\mathbf{S} = \mathbf{M} - \mathbf{N}\mathbf{K}^{-1}\mathbf{J}$$

And the inverse matrix of \mathbf{C} can be written as

$$\mathbf{C}^{-1} = \begin{bmatrix} \mathbf{C}^{11} & \mathbf{C}^{12} \\ \mathbf{C}^{21} & \mathbf{C}^{22} \end{bmatrix} = \begin{bmatrix} \mathbf{S}^{-1} & -\mathbf{S}^{-1}\mathbf{N}\mathbf{K}^{-1} \\ -\mathbf{K}^{-1}\mathbf{J}\mathbf{S}^{-1} & \mathbf{K}^{-1} + \mathbf{K}^{-1}\mathbf{J}\mathbf{S}^{-1}\mathbf{N}\mathbf{K}^{-1} \end{bmatrix} \quad (2.8)$$

So from Equation 2.4 we have

$$\begin{bmatrix} \hat{\tau} \\ \hat{u} \end{bmatrix} = \begin{bmatrix} C^{11} & C^{12} \\ C^{21} & C^{22} \end{bmatrix} \begin{bmatrix} X^\top R^{-1} y \\ Z^\top R^{-1} y \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} y$$

Here

$$\begin{aligned} U_1 &= C^{11} X^\top R^{-1} + C^{12} Z^\top R^{-1} \\ U_2 &= C^{21} X^\top R^{-1} + C^{22} Z^\top R^{-1} \end{aligned}$$

From Equation 2.4 and Equation 2.7, we have $\hat{\tau} \sim N(\tau, U_1(R + ZGZ^\top)U_1^\top)$

And we have following results

$$\begin{aligned} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \begin{bmatrix} X & Z \end{bmatrix} &= \begin{bmatrix} C^{11} & C^{12} \\ C^{21} & C^{22} \end{bmatrix} \begin{bmatrix} X^\top R^{-1} \\ Z^\top R^{-1} \end{bmatrix} \begin{bmatrix} X & Z \end{bmatrix} \\ &= \begin{bmatrix} C^{11} & C^{12} \\ C^{21} & C^{22} \end{bmatrix} \begin{bmatrix} X^\top R^{-1} X & X^\top R^{-1} Z \\ Z^\top R^{-1} X & Z^\top R^{-1} Z \end{bmatrix} \\ &= \begin{bmatrix} C^{11} & C^{12} \\ C^{21} & C^{22} \end{bmatrix} \left(\begin{bmatrix} X^\top R^{-1} X & X^\top R^{-1} Z \\ Z^\top R^{-1} X & Z^\top R^{-1} Z + G^{-1} \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & G^{-1} \end{bmatrix} \right) \\ &= I - \begin{bmatrix} 0 & -C^{12}G^{-1} \\ 0 & -C^{22}G^{-1} \end{bmatrix} \\ &= \begin{bmatrix} I & -C^{12}G^{-1} \\ 0 & I - C^{22}G^{-1} \end{bmatrix} \end{aligned}$$

So we have

$$\begin{aligned} U_1 X &= I \\ U_1 Z &= -C^{12}G^{-1} \end{aligned}$$

For the variance of estimation we have

$$\begin{aligned} \text{var}(\hat{\tau}) &= U_1(R + ZGZ^\top)U_1^\top \\ &= U_1 R U_1^\top + U_1 Z G Z^\top U_1^\top \\ &= (C^{11} X^\top R^{-1} + C^{12} Z^\top R^{-1}) R U_1^\top + U_1 Z G Z^\top U_1^\top \\ &= C^{11} X^\top U_1^\top + C^{12} Z^\top U_1^\top + C^{12} G^{-1} G (G^{-1})^\top (C^{12})^\top \\ &= C^{11} - C^{12} (G^{-1})^\top (C^{12})^\top + C^{12} (G^{-1})^\top (C^{12})^\top \\ &= C^{11} \end{aligned}$$

what is the relation between C_{11} and C^{11} ? From Equation 2.8 we have $C^{11} = S^{-1}$

and S is

$$S = \mathbf{X}^\top \mathbf{R}^{-1} \mathbf{X} - \mathbf{X}^\top \mathbf{R}^{-1} \mathbf{Z} (\mathbf{Z}^\top \mathbf{R}^{-1} \mathbf{Z} + \mathbf{G}^{-1})^{-1} \mathbf{Z}^\top \mathbf{R}^{-1} \mathbf{X}$$

And base on the complement of \mathbf{C}_{11} , we rewrite the S

$$\begin{aligned} S &= \mathbf{X}^\top (\mathbf{R}^{-1} - \mathbf{X}^\top \mathbf{R}^{-1} \mathbf{Z} (\mathbf{Z}^\top \mathbf{R}^{-1} \mathbf{Z} + \mathbf{G}^{-1})^{-1} \mathbf{Z}^\top \mathbf{R}^{-1}) \mathbf{X} \\ &= \mathbf{X}^\top \mathbf{P} \mathbf{X} \\ &= \mathbf{C}_{11} \end{aligned}$$

So $\text{var}(\hat{\boldsymbol{\tau}}) = \mathbf{C}^{11} = \mathbf{C}_{11}^{-1}$, which means $(\boldsymbol{\tau} - \hat{\boldsymbol{\tau}}) \sim N(0, \mathbf{C}_{11}^{-1})$. \square

To examine a specific form of $\boldsymbol{\tau}$, in general case, we do linear transform on $\boldsymbol{\tau}$: $\hat{\boldsymbol{\pi}} = \mathbf{D}\hat{\boldsymbol{\tau}}$, where \mathbf{D} is some transform matrix, so we have $\mathbf{D}(\boldsymbol{\tau} - \hat{\boldsymbol{\tau}}) = \boldsymbol{\pi} - \hat{\boldsymbol{\pi}} \sim N(0, \mathbf{D}\mathbf{C}_{11}^{-1}\mathbf{D}^\top)$. We denote $\boldsymbol{\Lambda} = \mathbf{D}\mathbf{C}_{11}^{-1}\mathbf{D}^\top$. If \mathbf{D} is identical matrix \mathbf{I} , then $\boldsymbol{\Lambda} = \mathbf{C}_{11}^{-1}$ and $\hat{\boldsymbol{\pi}} = \hat{\boldsymbol{\tau}}$.

We know that A-criterion is the mean of predicted error variance of the parameter from Equation 2.6 i.e.

$$\mathcal{A} = \frac{1}{n_\tau(n_\tau - 1)} \sum_i \sum_{j < i} V_{ij}$$

Having variance-covariance matrix $\boldsymbol{\Lambda} = \mathbf{C}_{11}^{-1}$, we can rewrite the sum part as $n_\tau \text{tr}(\boldsymbol{\Lambda}) - \mathbb{K}_{n_\tau}^\top \boldsymbol{\Lambda} \mathbb{K}_{n_\tau}$. So we have

$$\mathcal{A} = \frac{1}{n_\tau(n_\tau - 1)} [n_\tau \text{tr}(\boldsymbol{\Lambda}) - \boldsymbol{\infty}_{n_\tau}^\top \boldsymbol{\Lambda} \boldsymbol{\infty}_{n_\tau}] \quad (2.9)$$

same result from Butler et al. [2013]

Derivation above indicate that $\mathcal{A} \propto \text{tr}(\boldsymbol{\Lambda})$, A-criterion as the mean of predicted error variance of the parameter, we prefer it as small as possible to obtain a accurate result from experiment, which means the trace of virance-covirance matrix $\boldsymbol{\Lambda}$ should be as small as possible. And this is our goal on optimal experimental design.

2.3 Neighbor balance and evenness of distribution

2.3.1 Concepts of NB and ED

Piepho et al. [2018] emphasis the the concepts of neighbour balance and even distribution are crucial to mitigating biases and ensuring the reliability of results in row-column design.

Neighbour balance (NB) refers to the principle that, in a row-column experimental design, the frequency with which two treatments are adjacent or near each other should not be excessively high. High adjacency frequency between two treatments can lead to mutual influence, which may cause bias to the experimental results. For example, if the effect of one treatment can spread to neighbouring areas, frequent adjacency could interfere with accurate measurement of each treatment's true effect next to it. Therefore, it is essential to control the adjacency frequency of different treatments to prevent high adjacency for two specific treatments.

Even distribution (ED) aims to ensure that different replications of the same treatment are widely distributed across the experimental field, rather than being clustered in a specific area. This strategy helps to avoid biases caused by specific environmental conditions in certain parts of the experiment field. If replications of one treatment are over concentrated in one area, unique environmental factors in that area might affect the treatment's performance, leading to biased observations. By evenly distributing replications, environmental interference can be minimized, so that we can enhance the reliability of the experimental results.

See Figure 1.2 for instances where bad NB and ED occur.

2.3.2 Measuring NB and ED

Evaluating NB with adjacency matrix

In Piepho et al. [2018], there is a assumption that they are optimizing a binary design, which means each treatment appears only once in each row and column. Under this assumption, their balancing mechanism considers diagonal adjacency, Knight moves, and even more distant points to ensure an optimal balance. In my optimization process, I begin with a randomly selected design matrix. Consequently, my approach considers not only diagonal adjacency but also the adjacent points directly above, below, to the left, and to the right.

We use an adjacency matrix to count the number of times each treatment is adjacent to another. This matrix serves as a crucial tool in my optimization process, enabling precise tracking and adjustment of treatment placements to achieve neighbour balance.

We denote the adjacency matrix as \mathcal{U} , and for treatment i and j in treatment set T , the symbol \mathcal{U}_{ij} represents the count of times treatment i is adjacent to treatment j . Here "adjacent" means treatment j is located next to treatment i

For Given design \mathcal{D} and $\mathcal{D}_{r,c}$ represents the treatment at row r and column c . So \mathcal{U}_{ij}

can be expressed as:

$$\mathcal{U}_{ij} = \sum_{r=1}^{n_r} \sum_{c=1}^{n_c} I_{r,c}(i) F_{r,c}(j)$$

where

$$F_{r,c}(j) = \sum_{m \in \{-1,0,1\}} \sum_{n \in \{-1,1\}} I_{r+m,c+n}(j) + \sum_{m \in \{-1,1\}} I_{r+m,c}(j)$$

n_r and n_c are total number of rows and columns and $I_{r,c}(\cdot)$ is the indicator function, which takes value under following cases

$$I_{r,c}(i) = \begin{cases} 1 & \text{if } \mathcal{D}_{r,c} = i \\ 0 & \text{if } \mathcal{D}_{r,c} \neq i \text{ or } r < 1, r > n_r, c < 1, c > n_c \end{cases}$$

The function $F_{r,c}(j)$ here is actually counting the times that treatment j occurs at places around the position row r and column c .

We measure NB by taking difference of the maximum and minimum of the elements in adjacency matrix \mathcal{U} . Our NB criteria is

$$C_{NB} = \max\{\mathcal{U}_{ij}\} - \min\{\mathcal{U}_{ij}\} \quad i, j \in T$$

Evaluating ED with minimum row and column span

The goal of evaluating the evenness of distribution (ED) is to find the row and column spans for treatments across the entire design matrix. We would like this value as large as possible. This ensures that the treatments $t \in T$ is distributed as evenly as possible within the rows and columns, reducing clustering and promoting a balanced design.

The row span for a given treatment $t \in T$ is defined as the difference between the maximum and minimum row indices where t appears in experiment field.

$$RS(t) = \max\{r : \mathcal{D}_{r,c} = t\} - \min\{r : \mathcal{D}_{r,c} = t\} \quad 1 < r < n_r, \quad 1 < c < n_c$$

And the minimum row span of a design \mathcal{D} is

$$MRS(\mathcal{D}) = \min\{RS(t)\}, \quad t \in T$$

Same for column span

$$CS(t) = \max\{c : \mathcal{D}_{r,c} = t\} - \min\{c : \mathcal{D}_{r,c} = t\} \quad 1 < r < n_r, \quad 1 < c < n_c$$

$$MCS(\mathcal{D}) = \min\{CS(t)\}, \quad t \in T$$

So, for the changes in the design matrix \mathcal{D} during the search process, we tend to accept only those changes where the Minimum Row Span (MRS) and Minimum Column Span (MCS) remain the same or become smaller.

Chapter 3

Methods

3.1 Modeling row-column design

As mentioned in Chapter 2, we use a linear mixed model (LMM) to model the row-column design having two distinct sources of variation, typically referred to as “row” and “column” factors.

Recalling the assumptions, the row and column effects are treated as random effects, which means that they are random factors for spatial or systematic factors across different rows and columns of the experiment field. The treatment effects, on the other hand, are treated as fixed effects because they represent the primary factors of interest that we wish to evaluate in terms of their influence on the response variable.

In Equation 2.1, the treatment effects are modelled as fixed effects, represented by the treatment design matrix X with parameter vector τ , measuring the influence of each treatment on the response variable. The matrix X is constructed such that each row corresponds to an experimental unit, and indicators in each column indicates whether a treatment is applied or not.

The random effects are modelled through the matrix Z and parameter vector u . Matrix Z is designed to capture the row and column structure of the experimental field, in which entries represent the position of each experimental unit located in some specific rows and columns. The parameters in vector u are corresponding row and column effects. They are assumed to follow a normal distribution with mean zero and variance-covariance matrix G .

With y as the vector of observed responses and ϵ as error term, a row-column design can be modelled by Equation 2.1 . where $X\tau$ represents the fixed treatment

effects and Zu captures the random variations basing on rows and columns.

3.1.1 Random effects matrix

The design matrix for the random effects, which is row and column in linear mixed model, follows a binary indicator structure, denoted as Z . For example, we have a 4×4 experiment field, having 4 rows, 4 columns and 16 units. Then random effects matrix should be a 16×8 matrix containing binary indicator as the one presented.

$$Z_{\text{example}} = \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array} \right]$$

Each row corresponds to a specific experimental unit, while the columns represent the row and column factors in the experimental layout. In this matrix, the first set of columns represents the column effects, while the second set of columns represents the row effects. Each entry in this matrix is binary, where a value of 1 indicates that the experimental unit belongs to a specific row or column, and a 0 indicates otherwise. For example, the first row of the matrix has a 1 in both first and fifth columns, meaning that the corresponding unit of it is in the first column and the first row. This structure ensures that each unit is uniquely associated with one row and one column, and we can model the random effects accordingly.

In a more general case, suppose we have a $n_r \times n_c$ row-column experiment field. We should have a random effect matrix with $n_r n_c$ rows and $n_r + n_c$ columns with binary numbers. In this thesis, we assume that the row effects and column effects are independent with each other. However, in more complex experimental design

cases, they may be potentially correlated. The design of the random effects matrix, which separates the row and column effects as independent variables, simplifies the modelling process and the analysis of potential correlations between these effects in more advanced settings. This structure allows for easier identification and analysis of interactions between row and column effects, making the model flexible and adaptable to different levels of complexity in experimental designs.

3.1.2 Design matrix for treatments

The design matrix for the treatment effects is constructed to capture the influence of each treatment on the response variable. In a row-column experimental design, each experimental unit is assigned a specific treatment. The entries in design matrix represents these assignments using binary indicators. Like random effects matrix each row in the matrix corresponds to an experimental unit, while each column represents a different treatment. Here we still use 4×4 experiment field as example, and suppose we have 4 different treatments for each have 4 replications, needing 16 experiment unite. An example design matrix \mathbf{X} for treatments should be a 16×4 matrix with binary indicators as shown below

$$\mathbf{X}_{\text{example}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

For a given experimental unit, that is a given row, the design matrix contains a 1 in the column corresponding to the treatment applied to that unit, and 0 elsewhere. This structure allows for a clear and efficient representation of which treat-

ment is applied to each unit. For example, the first row of the example design matrix represents the first treatment is applied in the unit locating on the first column, first row.

if there are t treatments and N experimental units, the design matrix will have N rows and t columns. Then the design matrix for treatment X with size $N \times t$, should satisfy that for any row n_i

$$\sum_{j=1}^t X_{n_i,j} = 1$$

That is, there is only one treatment can be applied in each experiment unit. And for any treatment t_j with r_j replications, it has

$$\sum_{i=1}^N X_{i,t_j} = r_j$$

All replications of a treatment are applied in experimental field.

3.1.3 Assumptions for A-value calculation

It is important to clarify the key assumptions made in this study before calculating A value for a row-column design. Recalling Equation 2.9, for calculating A value we need the covariance matrix for the random effects, matrix G , covariance matrix for the error term, matrix R , transformation matrix D , random effect matrix Z and treatment design matrix X . We need some basic set-up for these matrices.

Assume that we now have a row-column matrix with n_r rows, n_c columns and $n_r n_c$ plots.

For the covariance matrix for the random effects, matrix G , which captures the variability introduced by the row and column effects. I assume it is a $(n_r + n_c) \times (n_r + n_c)$ diagonal matrix, that is, it has following form,

$$G_{diag} = \begin{bmatrix} \sigma_{G_c}^2 I_{n_c} & 0 \\ 0 & \sigma_{G_r}^2 I_{n_r} \end{bmatrix}$$

I_{n_c} is a $n_c \times n_c$ identity matrix, same to I_{n_r} . And σ_{G_c} and σ_{G_r} is a scale constant. This means that the influences of the rows and columns are independent of each other, meaning there is no correlation between different row and column effects in the design.

Similarly, for the covariance matrix for the error term, matrix R , we assume that it is also diagonal, indicating that the residual errors are uncorrelated across different

experimental units. In this case we have

$$\mathbf{R}_{diag} = \sigma_R^2 \mathbf{I}_{n_r n_c}$$

with identity matrix $\mathbf{I}_{n_r n_c}$ and scale constant σ_R .

Butler [2013] have introduced linear transformations of the treatment parameter vector $\boldsymbol{\tau}$ by using a transformation matrix \mathbf{D} , which allows for the investigation of linear combinations of treatments. However, in this paper, I simplify the approach by setting \mathbf{D} as the identity matrix \mathbf{I}_{RC} . This means that we focus on the individual effects of the treatments rather than their linear combinations.

These assumptions makes the structure of the model becomes more straightforward, allowing us to concentrate on the direct estimation of treatment effects while maintaining independence among the random effects and error terms.

It's important to note that in our design, the random effect matrix \mathbf{Z} remains constant during the optimization process. This means that while the row and column effects are accounted for as random effects, their structure does not change.

With these assumptions in place, the A-value in the context of a row-column design depends only on the treatment design matrix \mathbf{X} . This means that the primary factor influencing the A-value is the distribution and arrangement of treatments within the design, and it directly impacts the variance of the treatment effect estimates. Therefore, optimizing the A-value under these assumptions becomes a problem of optimizing the treatment distribution in the design, ensuring that the treatments are arranged in such a way that the variance of the estimates is minimized.

3.2 Searching Strategy

Before introducing the details of the searching strategy, it is important to establish a solid foundation by proving that the minimum of the A-value exists. The existence of the minimum A-value implies that the A-value has a lower bound, ensuring that the process of iteration optimizing the experimental design is not endless. As we continue to search for smaller A-values, this guarantees that we can eventually stop when the A-value stabilizes or a sufficient number of iterations reached.

This allows us to conclude that we have found an optimal or near-optimal design. Therefore, the existence of this lower bound serves as a critical foundation for our iterative search, giving us confidence that the optimization will converge to a solution.

3.2.1 Existence of the minimum of A-value

To prove that the minimum of the A-value exists, we establish a objective function, that is, the A-value function $A(\mathbf{X})$ maps design matrix for treatment \mathbf{X} to its A-value. It is a function that maps the design space to \mathbb{R} .

$$A : \Omega \rightarrow \mathbb{R}, \quad \mathbf{X} \mapsto A(\mathbf{X})$$

Here Ω is the design space contains all possible design matrix \mathbf{X} .

Now we proof the existence of minimum value of \mathbf{X} , where $\mathbf{X} \in \Omega$.

Proof. Recalling Equation 2.9

$$\mathcal{A} = \frac{1}{n_\tau(n_\tau - 1)} [n_\tau \text{tr}(\mathbf{\Lambda}) - \infty_{n_\tau}^\top \mathbf{\Lambda} \infty_{n_\tau}]$$

This expression is well-defined for all valid design matrices \mathbf{X} . So function $A(\mathbf{X})$ is well-defined.

The A-value represents the average variance of the difference treatment effect estimates, and since variances are always positive, the A-value is naturally bounded from below by zero. So we have

$$A(\mathbf{X}) \geq 0$$

which implies that the A-criterion is bounded below.

In experimental design, the treatment design matrix \mathbf{X} can take on a finite number of possible permutations, especially in practical row-column designs where the number of treatments and experimental units is fixed. In a finite search space, a lower bounded function has its minimum value. \square

3.2.2 General structure

In Piepho et al. [2018], the optimization of NB and ED was typically carried out under the assumption that the A-value was already optimal or fixed. This required identifying a set of solutions that maintained the A-value while improving the balance and distribution properties. In addition to assuming the A-value is fixed, another approach they used is to randomly select a design and then optimizing ED and NB. This process would be repeated multiple times, and the design with the best A-value will be selected.

These approach separates the optimization of ED and NB from the A-value, while we try to merge these two process into one algorithm. We use pairwise permuta-

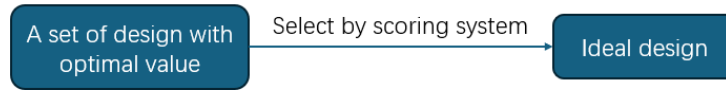


Figure 3.1: Method been used

tion among treatments to change treatment design during iterations. And to avoid design with bad ED and NB, I am using some criteria to filter the permutation, only maintain or better properties are accepted. In this way, a row-column design that satisfies multiple optimization requirements is achieved.

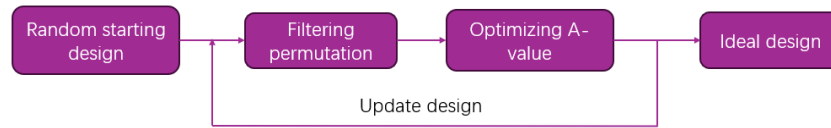


Figure 3.2: Attempting method

Basing on optimal design search methods share a common set of features in exploring the design space given in Butler [2013], my searching method contains following part:

1. A calculation method for an optimal criterion for a given design matrix X
2. An interchange policy to switch the design with in search space Ω
3. An acceptance policy for a new design.
4. A stopping rule to terminate the search.

The criterion calculation part has already been discussed earlier. We will now introduce interchange policy of switching the design.

3.2.3 Permutations and filtering

We use permutations of the treatments to update the design matrix for treatment. We randomly select two different treatments and swap them within the design matrix during the permutation process, without making drastic changes.

Let X be the current design matrix for treatments, where each row corresponds to an experimental unit, and each column represents a treatment. Suppose we randomly select two different treatments, t_i and t_j located in i th row and j th row respectively. X_{new} can be written as

$$X_{new} = P_{ij}X$$

with permutation matrix defined as

$$P_{ij} = \begin{bmatrix} 1 & 0 & \dots & 0 & \dots & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 0 & \dots & 1 & \dots & 0 \\ \vdots & \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & 0 & \dots & 1 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{bmatrix}$$

It is an identity matrix with i -th row and j -th row swapped.

When performing permutations on the design matrix, we apply checks based on the metrics of evenness of distribution (ED) and neighbour balance (NB). The goal is to ensure that only the permutations which improve or at least maintain desirable values for ED and NB are accepted, while others are filtered out.

A random permutation of treatments is generated by swapping two different treatments in the design matrix, as described earlier using a permutation matrix. Once the new design matrix \mathbf{X}_{new} with corresponding design \mathcal{D}_{new} is created, the next step is to evaluate the quality of the permutation by calculating the ED and NB values for the new configuration. As afore-mentioned, we have NB and ED criteria for \mathbf{X}_{new} , that is C'_{NB} , $MRS(\mathcal{D}_{new})$ and $MCS(\mathcal{D}_{new})$. Comparing the newly generated design matrix (offspring) with the original design matrix (parent), We accept the new permutation, if the ED and NB values improves or maintains them without significantly worse. In practice, we often set a tolerance for the ED and NB values. We generally allow ED and NB to become slightly worse, as it's not necessary for them to strictly improve or remain unchanged in every iteration. Our goal is to achieve a balance between ED, NB, and the A-value. For instance, ED might increase slightly while NB decreases a little, as long as the overall balance between the three objectives is maintained. This approach has the added benefit of lowering the acceptance threshold for permutations, which speeds up the algorithm during the random selection process. For instance, we set tolerance for ED and NB as T_{ED} and T_{NB} , which are none negative numbers. Current design matrix \mathbf{X} corresponding design \mathcal{D} has ED and NB value C_{NB} , $MRS(\mathcal{D})$ and $MCS(\mathcal{D})$. New design matrix

\mathbf{X}_{new} mentioned above is accepted when

$$\begin{aligned} (C'_{NB} &\leq C_{NB} + T_{NB}) \\ \wedge (MRS(\mathcal{D}_{new}) &\geq MRS(\mathcal{D}) - T_{ED}) \\ \wedge (MCS(\mathcal{D}_{new}) &\geq MCS(\mathcal{D}) - T_{ED}) \end{aligned} \quad (3.1)$$

is true.

3.2.4 Random search

The random search algorithm begins with a randomly selected design matrix. To ensure that the search is efficient and avoids getting trapped in local optima, we introduce the concept of step length. This parameter determines how many permutations we consider in each iteration. Given the computational constraints, especially when the number of treatments or rows and columns increases, it is impractical to check all possible permutations of a design and evaluate each for acceptance. However, we aim to explore as many permutations as possible to avoid falling into local optima.

At each iteration, we randomly generate a set of permutations. For each generated permutation, we apply the filtering step check by Equation 3.1. If the permutation is not accepted, we randomly select another one and repeat the process. This continues until we have successfully selected a number of permutations equal to the step length.

We denote step length as s . tolerance for ED and NB as t_{ED} and t_{NB} . For a design matrix \mathbf{X} , its ED criteria - minimum row span and minimum column span is, $mrs(\mathbf{X})$ and $mcs(\mathbf{X})$, and NB criteria is denote as $C_{NB}(\mathbf{X})$.

1. **Input:** Original design matrix \mathbf{X} , step length s , tolerance for ED and NB as t_{ED} and t_{NB} .
2. **Initialize:** $k = 1$, ED and NB value for \mathbf{X} as $mrs(\mathbf{X})$, $mcs(\mathbf{X})$ and $C_{NB}(\mathbf{X})$.
3. While $k < s$:
 - Generate a random permutation matrix \mathbf{P} by selecting two different treatments.
 - Apply permutation: $\mathbf{X}_{new} = \mathbf{P}\mathbf{X}$.
 - Calculate new values of ED and NB, $mrs(\mathbf{X}_{new})$, $mcs(\mathbf{X}_{new})$ and $C_{NB}(\mathbf{X}_{new})$.
 - If new values satisfy $mrs(\mathbf{X}_{new}) \geq mrs(\mathbf{X}) - t_{ED}$, $mcs(\mathbf{X}_{new}) \geq mcs(\mathbf{X}) - t_{ED}$ and $C_{NB}(\mathbf{X}_{new}) \leq C_{NB}(\mathbf{X}) + t_{NB}$, accept and remember this \mathbf{X}_{new} .

- Increment k .
 - If $k = s$, output all selected permutations.
4. **Output:** A set of k permutations of original design matrix \mathbf{X}

The Random Search algorithm starts with a randomly selected initial design matrix \mathbf{X}_0 , and we aim to minimize the A-value associated with the design. We set a maximum number of iterations M and a step length s .

The process for each iteration can be described as follows:

We denote a random design matrix \mathbf{X}_0 , and iteration counter k . Maximum number of iteration is M . And denote the A-value for a random design matrix \mathbf{X} is $A(\mathbf{X})$. Current design matrix during the iteration we have is denoted as \mathbf{X}_c

1. **Initialization:** Start with a random design matrix \mathbf{X}_0 , set the iteration counter ' $k = 0$ ', maximum number of iteration as M and step-length s . A-value for the current design is $A_c = A(\mathbf{X}_0)$.
2. **Iteration:** For each iteration k , where $k < M$:
 - Generate s random permutations $\{\mathbf{X}_i\}$, $i = 1, 2, \dots, s$ of the current design matrix \mathbf{X}_c .
 - Calculate the A-value $A(\mathbf{X}_i)$ for each permutation \mathbf{X}_i .
 - Compare the A-values for all s permutations with the current design \mathbf{X}_c .
 - Select the permutation with the smallest A-value, \mathbf{X}_j , among the s candidates.
 - If $A(\mathbf{X}_j) < A(\mathbf{X}_c)$ update the current design matrix: $\mathbf{X}_c = \mathbf{X}_j$; otherwise, retain the current design.
3. **Termination:** Repeat this process until the maximum number of iterations M is reached.

3.2.5 Simulated annealing

Simulated Annealing (SA) is a global optimization algorithm inspired by the annealing process in metallurgy. At higher temperatures, the algorithm allows the acceptance of worse solutions to escape local optima; as the temperature decreases, it converges toward an optimal solution.

Butler [2013] state that the Boltzmann probability

$$P(E) \propto e^{[-E/kt]}$$

offer a pathway to measuring the accept possibility during the algorithm, where E is the energy of the state, for a given temperature t . The constant k is Boltzmann's constant. The energy E corresponds to the value of the objective function, in our case is A-value. During the iterative process, suppose we have a design (state) having A-value (energy) A_1 at time t , and we are shifting our design into a new design with A-value A_2 , resulting in an energy change $A_\Delta = A_2 - A_1$. If A_Δ is negative, that is $A_2 < A_1$, we always accept new design since we have lower A-value. If A_Δ is positive, acceptance follows the Metropolis criterion: a random number, $\delta \in [0, 1]$, is generated, and A_2 is accepted if $\delta \leq \exp(-A_\Delta/kt)$. So we have acceptance rate $P(A_2)$ for a new A-value A_2 .

$$P(A_2) = \begin{cases} 1 & \text{if } A_2 < A_1 \\ \exp(-A_\Delta/kt) & \text{if } A_2 > A_1 \end{cases} \quad (3.2)$$

The basic elements of SA given by Bertsimas and Tsitsiklis [1993] is as followed

1. A finite set S .
2. A real-valued cost function J defined on S . Let $S^* \subset S$ be the set of global minima of the function J , assumed to be a proper subset of S .
3. For each $i \in S$, a set $S(i) \subset S - i$, called the set of neighbours of i .
4. For every i , a collection of positive coefficients q_{ij} , $j \in S(i)$, such that $\sum_{j \in S(i)} q_{ij} = 1$. It is assumed that $j \in S(i)$, if $i \in S(j)$.
5. A nonincreasing function $T : N \rightarrow (0, \infty)$, called the cooling schedule. Here N is the set of positive integers, and $T(t)$ is called the temperature at time t .
6. An initial "state" $x(0) \in S$.

We are applying these elements to a SA that fits our case. Our SA having following elements.

1. A finite searching space $\{\mathbf{X}\}$ consisting all design matrix.
2. An A-value function $A(\mathbf{X})$ defined on $\{\mathbf{X}\}$ and it is real-valued. There is a set of $\{\mathbf{X}^*\}$ that having optimal A-value and $\{\mathbf{X}^*\} \in \{\mathbf{X}\}$
3. For each design matrix \mathbf{X} in searching space, we consider all possible permutations filtered by Equation 3.1 are neighbours of \mathbf{X} , which is a subset of searching space $\{\mathbf{X}\}$.
4. All possible permutations have equal possibility to be chose, and the sum of the possibility is equal to 1.
5. A nonincreasing function $T : N \rightarrow (0, \infty)$, $T(t)$ is called the temperature at t -th iteration.
6. A random initial design \mathbf{X}_0

Suppose we have a current design matrix \mathbf{X}_i and its neighbours filtered by Equation 3.1 contains n_p numbers of design. The next design is determined as follows:

A design matrix \mathbf{X}_j is randomly picked from the neighbours of \mathbf{X}_i . Suppose there is n_p design matrices in the neighbours of \mathbf{X}_i , then the probability of selecting \mathbf{X}_j among all neighbours is $\frac{1}{n_p}$. This is actually the positive coefficients q_{ij} aforementioned in basic elements of SA. We now denote $\mathbf{X}(t)$ to be the design matrix at t -th iteration. Once \mathbf{X}_j is chosen, the next design matrix is determined as follows:

$$P(\mathbf{X}(t+1) = \mathbf{X}_j | \mathbf{X}(t) = \mathbf{X}_i) = \begin{cases} 1 & \text{if } A(\mathbf{X}_j) < A(\mathbf{X}_i) \\ \frac{1}{n_p} \exp\left(\frac{-(A(\mathbf{X}_j) - A(\mathbf{X}_i))}{T(t)}\right) & \text{if } A(\mathbf{X}_j) > A(\mathbf{X}_i) \end{cases}$$

Convergence analyze

In this section, We will discuss the convergence properties of the SA algorithm. During iteration process, temperature cooling schedule plays an important role in convergence. It determines whether the algorithm will reach an optimal or near-optimal solution over time. Basing on work in Sasaki and Hajek [1988], Bertsimas and Tsitsiklis [1993] gives a conclusion on convergence properties, with aforementioned basic element. Define that state i communicates with S^* at height h if there exists a path in S that starts at i and ends at some element of S^* and the largest value of J along the path is $J(i) + h$. Denote d^* be the smallest number such that every $i \in S$ communicates with S^* at height d^* .

The SA algorithm converges if and only if

$$\lim_{t \rightarrow 0} T(t) = 0$$

and

$$\sum_{t=1}^{\infty} \exp\left[-\frac{d^*}{T(t)}\right] = \infty \quad (3.3)$$

To ensure the condition above, the mostly chose cooling schedule is

$$T(t) = \frac{d}{\log t} \quad (3.4)$$

Here d is some constant. It can be initial temperature.

Proof. It is obvious that $\lim_{t \rightarrow 0} T(t) = 0$ in Equation 3.4

Bring Equation 3.4 in to Equation 3.3, we have

$$\sum_{t=1}^{\infty} \exp\left[-\frac{d^* \log(t)}{d}\right] = \sum_{t=1}^{\infty} t^{-\frac{d^*}{d}}$$

It is a harmonic series and it diverge when $\frac{d^*}{d} < 1$, that is, $d^* < d$. So SA of we have a large enough d (initial temperature). \square

Although logarithmic cooling theoretically guarantees convergence to the global optimum, it is computationally demanding and has a very slow convergence rate in practice, making it less feasible for large-scale problems. To address these limitations, exponential cooling is introduced as a more practical alternative. While it does not offer a theoretical guarantee of reaching the global optimum, as noted by Kirkpatrick et al. [1983]. Because of limited time and computational resource, we will try to use exponential cooling in our as given in Aarts and Korst [1989] and well-practised It provides near-optimal solutions within a reasonable time, making it highly effective for large combinatorial optimization problems.

$$T(t) = T_0 \exp(-\alpha * t) \quad (3.5)$$

Here T_0 is the initial temperature, and α is the cooling rate determine how fast the temperature drop.

Algorithm

For accept probability Equation 3.2, we usually start with 0.8 and keep dropping when iteration goes. This probability depends on the magnitude of change in the objective function A-value. To simplify the initialization, we often conduct a few preliminary iterations to observe the range and rate of change in A-values, then use these observations to determine an initial temperature. Several studies have explored methods for autonomously selecting this initial temperature.

1. **Initialization:** Start with a random design matrix \mathbf{X}_0 , set the iteration counter ' $k = 0$ ', maximum number of iteration as M , A-value for the current design is $A_c = A(\mathbf{X}_0)$, initial temperature T_0 and a type of cooling schedule $T(t)$.
2. **Iteration:** For each iteration k , $k < M$:
 - Randomly pick one permutation \mathbf{X}_{new} among all neighbours of current design \mathbf{X}_c .

- Calculate the A-value $A(\mathbf{X}_{new})$ for permutation \mathbf{X}_{new} and compare it with $A(\mathbf{X}_c)$.
 - If $A(\mathbf{X}_{new}) < A(\mathbf{X}_c)$, update $\mathbf{X}_c = \mathbf{X}_{new}$.
 - If $A(\mathbf{X}_{new}) > A(\mathbf{X}_c)$, update $\mathbf{X}_c = \mathbf{X}_{new}$ with probability $\exp[-(A(\mathbf{X}_{new}) - A(\mathbf{X}_c))/T(t)]$.
3. **Termination:** Repeat this process until the maximum number of iterations M is reached.

Chapter 4

Results

4.1 Simulation set-up

To begin the Results section, we'll look at the foundational set-up of my simulation. In order to streamline the computation within limited computational resources, I implemented a series of simplifications.

For \mathbf{G} matrix, I set it to be a diagonal matrix with equal values along the diagonal. It implies that there is no correlation between rows and columns, meaning that the effects for rows and columns are considered independent. And the equal values along the diagonal indicate that the variance of these row and column effects is the same. In this simulation it is $\mathbf{G}_s = 10\mathbf{I}_{n_r+n_c}$

For \mathbf{R} matrix, similarly, it is set as a diagonal matrix with identical values along the diagonal. implying that the geographical locations of different plots are independent of each other and variance of residuals is uniform across plots. In this simulation it is $\mathbf{R}_s = 0.1\mathbf{I}_{n_r \times n_c}$

To compare the differences between algorithms, we used the design function from the R package `blocksdesign` to calculate the optimized A-value $A_{\text{blocksdesign}}$ basing on Edmondson [2020], which we then compared to the results from our iterative process.

For the consistency across all algorithms, we aimed to explore the largest feasible range of row, column, and treatment combinations. We set the maximum number of iterations $M = 2000$ for each algorithm. And we add an early termination criterion: if the iterative algorithm's A-value approaches $A_{\text{blocksdesign}}$ within a specified tolerance T_A , the algorithm would stop iterating and output the A-value and design matrix. In this simulation we set $T_A = 2 \times 10^{-4}$ Therefore we can evalu-

ate the performance of different algorithms across various combinations of row, column, and treatment numbers by comparing the total computational time required, the actual iteration number before reaching either M or an early termination based on $A_{\text{blocksdesign}}$, and the absolute difference between the algorithm's and $A_{\text{blocksdesign}}$. Basing on these evaluations, we can identify how effectively each algorithm performs under different complexity levels and determine the algorithm's efficiency and accuracy across different cases.

In setting up random selection, the final parameter is the step length s . During runtime, the process of selecting permutations for step-length iterations often accounts for a large portion of the total functioning time. Additionally, this selection becomes especially challenging when the row, column, and treatment numbers are low. In such cases, while applying filtering criteria, it can be difficult to find a sufficient number of unique permutations that meet the requirements, as the design itself may not contain enough options. To ensure the algorithm runs smoothly and completes within a reasonable time, we set the step length to $s = 3$.

For the SA algorithm, we need to set the initial temperature T_0 . It actually depends on the acceptance rate for higher A-values at the start of the iteration process. As mentioned in the previous section, we typically aim for an initial acceptance rate of around 0.8. That is we hope $\exp(-A_{\Delta}/T_0) \approx 0.8$. We often lack precise information on A_{Δ} before the algorithm, we set the initial temperature based on an empirical estimate. We set $T_0 = 1$. Some adaptive algorithms for setting the initial temperature are often discussed in the literature.

To examine combinations of row, column, and treatment numbers, we tested a wide range of configurations. Row number ranged from 5 to 25, and column number varied from 5 to 16. For each row and column combination, we tested treatment numbers of 10, 20, 50, 80, and 100. To ensure that the blocksdesign functions operated correctly and to maintain general applicability, we ship a combination, if the product of rows and columns was less than twice the treatment number. It ensures that each treatment could have at least two replication. For each combination of row, column and treatment number, we conduct three replication for more accurate observation.

For the cases when treatment number cannot evenly divide the product of rows and columns, suppose $n_{\text{plot}} = n_r \times n_c$ and treatment number is n_{τ} , we set base replication number as $\lfloor \frac{n_{\text{plot}}}{n_{\tau}} \rfloor$, and randomly pick $(n_{\text{plot}} \bmod n_{\tau})$ treatments assign $\lfloor \frac{n_{\text{plot}}}{n_{\tau}} \rfloor + 1$ replication to these treatments. This approach ensures that the number of each treatment is as close as possible to one another.

4.2 Result analyze

4.2.1 General behaviour of algorithms

Now we observe the overall behaviour of the algorithm, to examine the change of A-value during the process. We here use example with $n_c = n_r = 15$ and $n_\tau = 50$. we now use blocksdesign functions to generate a optimal design \mathcal{D}_{op} with A-value $A_{op} = 0.5027$ under set-up, NB criteria $C_{NB}(\mathcal{D}_{op}) = 3$ and ED criteria $MRS(\mathcal{D}_{op}) = MCS(\mathcal{D}_{op}) = 5$. See detailed assignment as follow

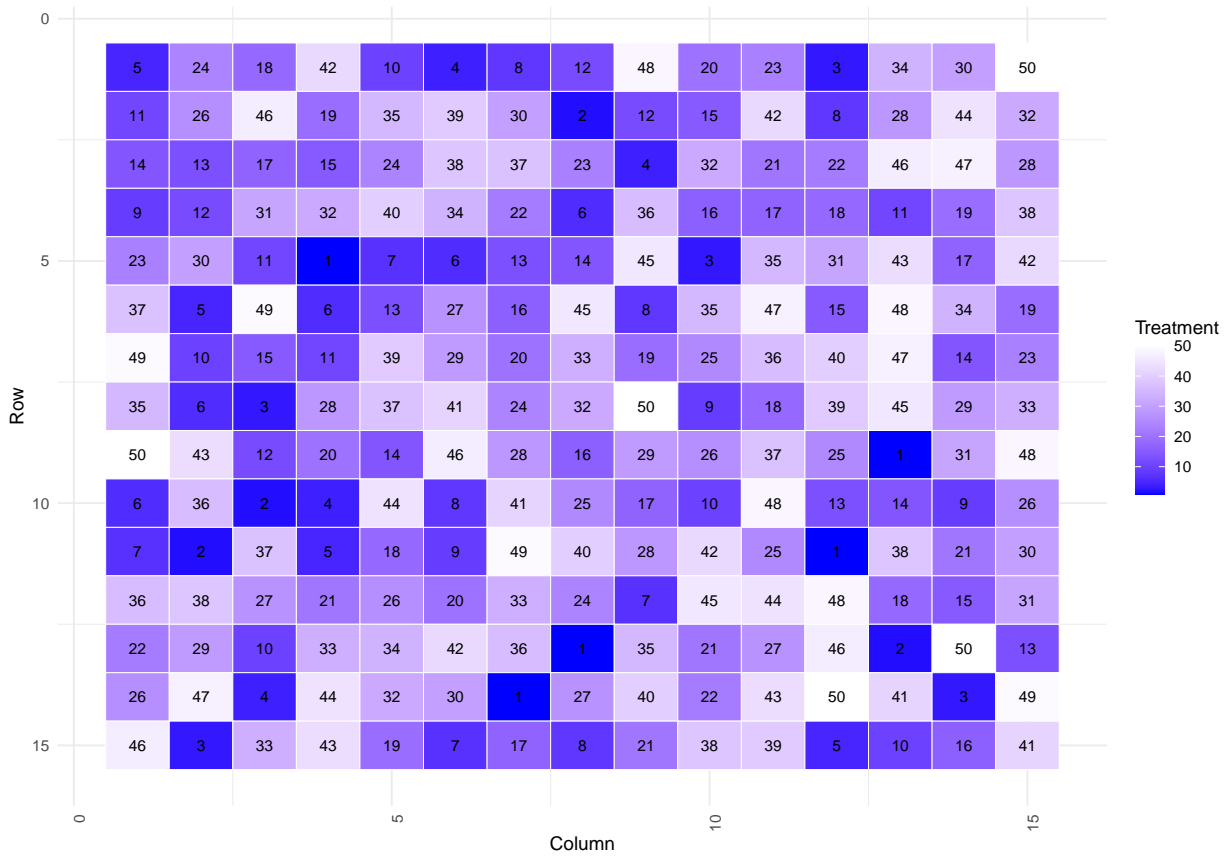


Figure 4.1: Detailed assignment for treatments given by design function in blocks-design package

Random selection

We now look at how random selection behave during the process. The iteration stops at 549-th step with output A-value $A_{rs} = 0.050478$ and NB criteria $C_{NB}(\mathcal{D}_{rs}) = 4$ and ED criteria $MRS(\mathcal{D}_{rs}) = 5$ and $MCS(\mathcal{D}_{rs}) = 7$. The changes in the A-value over iterations and the specific design are shown in Figure 4.2 and Figure 4.3.

From the results, the A-value obtained by Random Search is relatively close to that of the block design, though slightly larger, as is the NB statistic. Recall that we

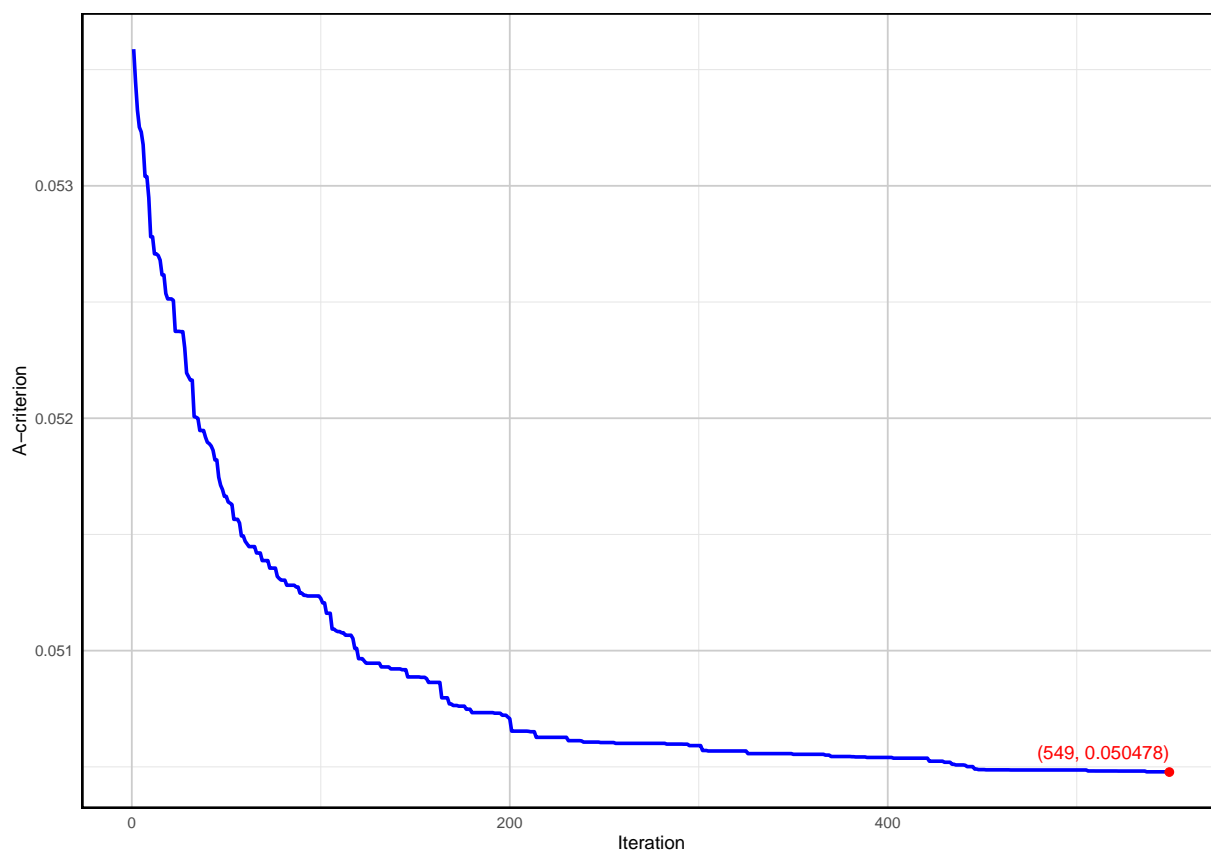


Figure 4.2: A v.s. Iteration (Random search)

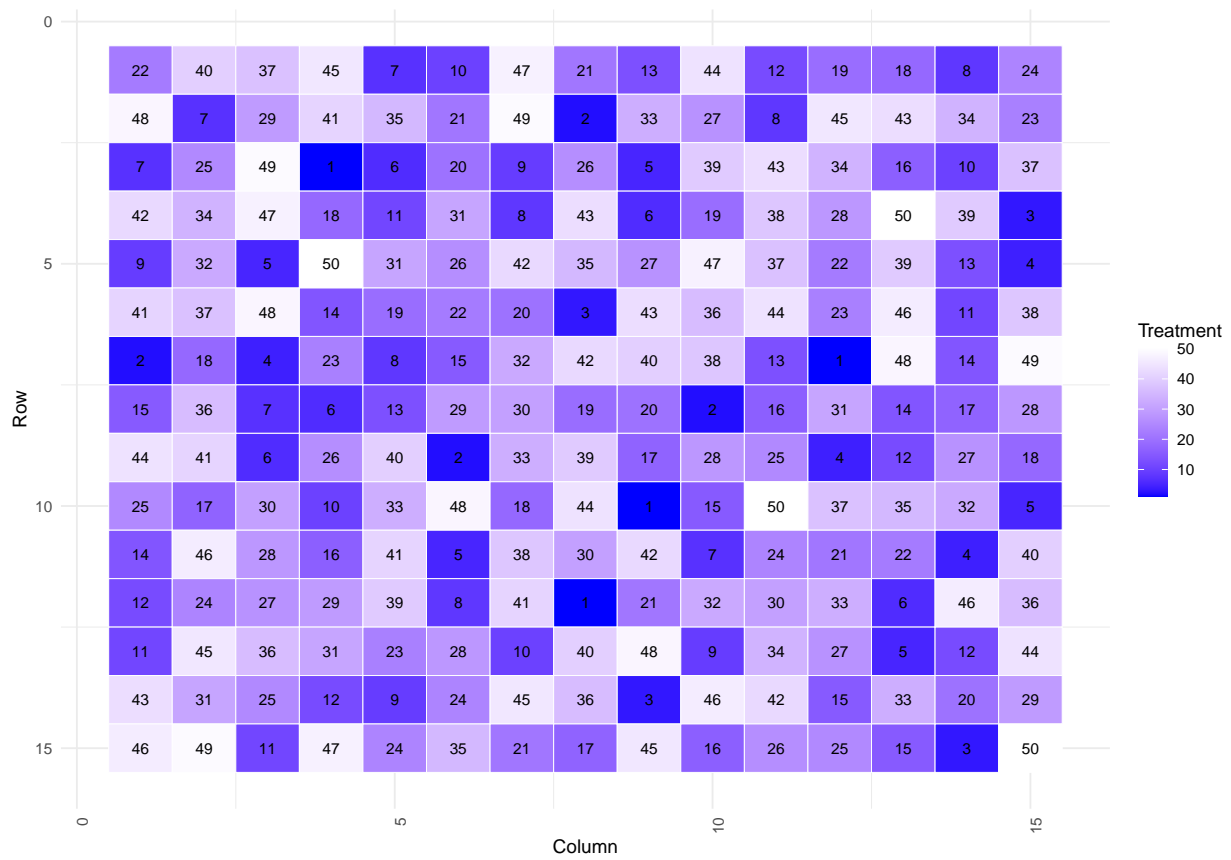


Figure 4.3: Detailed assignment for treatments given by random search

would like A-value and C_{NB} to be as small as possible, and for both MRC and MRC , the larger the better. Therefore, the Random Search outcome has a larger minimum column span, meaning it perform better on evenness of distribution.

SA with log cooling schedule

For SA using the log cooling schedule Equation 3.4, Unlike random selection, the algorithm accepts higher A-values with a certain probability, causing fluctuations in A-values throughout the iterations, resulting in rises and falls rather than a steady decline. Details can be seen in the Figure 4.4 and Figure 4.5.

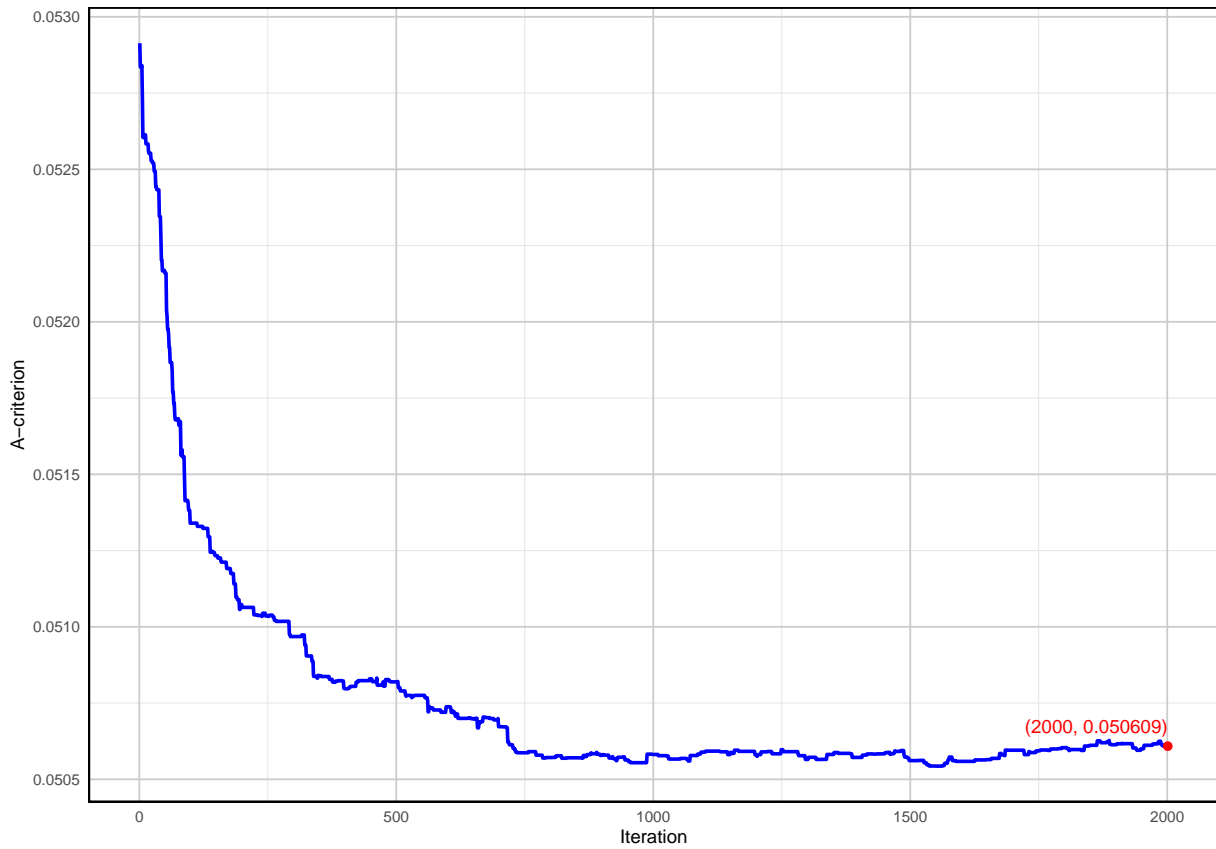


Figure 4.4: A v.s. Iteration (SA with log cooling schedule)

As it indicate in the plot, the process went through all iterations, reaching maximum iteration number M . At the 2000th iteration, the A-value is $A_{SAlog} = 0.050609$ with NB criteria $C_{SAlog}(\mathcal{D}_{SAlog}) = 5$ and ED criteria $MRS(\mathcal{D}_{SAlog}) = 7$ and $MCS(\mathcal{D}_{SAlog}) = 6$.

Compared to random selection, the optimization of the A-value in SA is less effective, since the limitation of a maximum iteration number M . Similar to the NB statistic, where SA shows a less optimal outcome than the block design. However, the two ED statistics given by SA are significantly improved compared to those

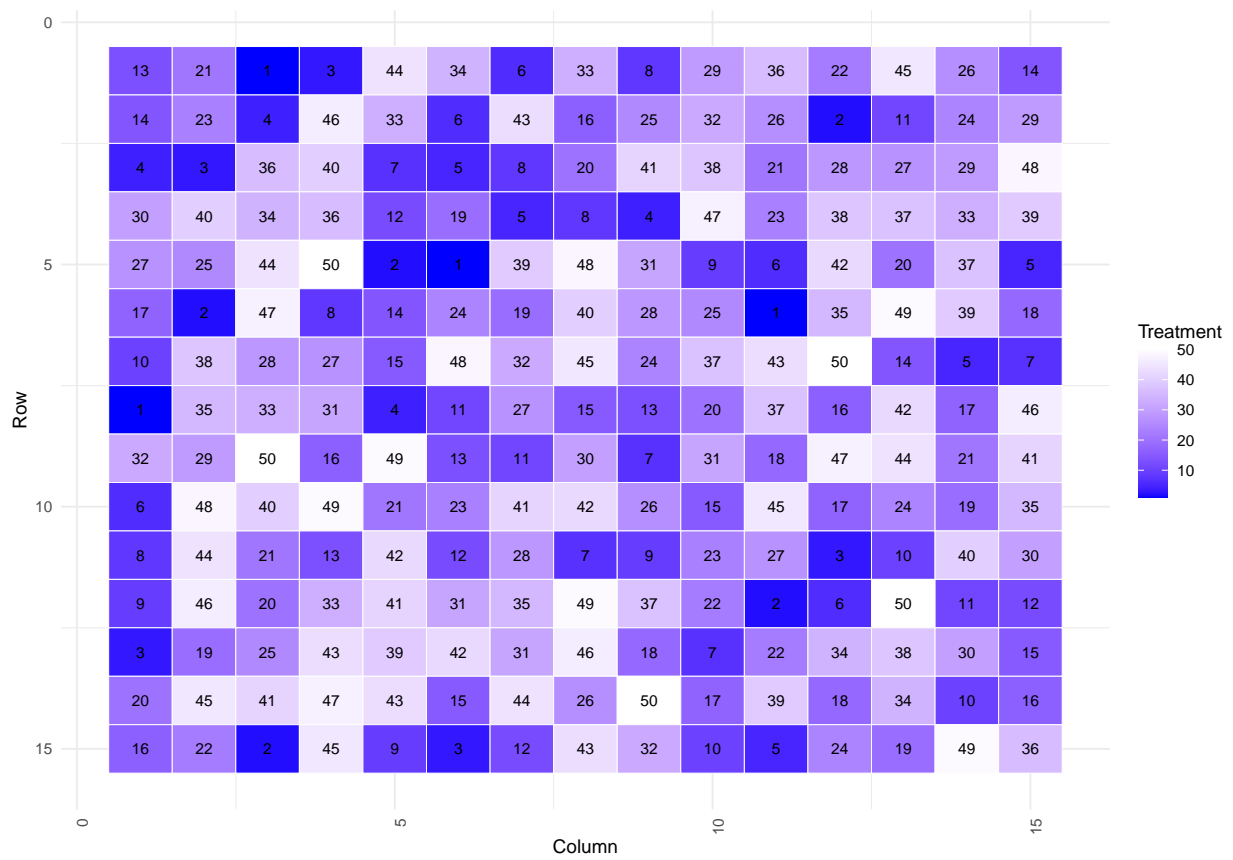


Figure 4.5: Detailed assignment for treatments given by SA with log cooling schedule

provided by the blockdesign function and random selection. Because of the slower decrease in acceptance rates for higher A-values with log cooling schedule, we observe that the algorithm occasionally continues to accept larger A-values even in the latter stages.

SA with exp cooling schedule

This is also SA but with an exponential cooling schedule Equation 3.5. Although exponential cooling schedule does not theoretically guarantee SA convergence to the global optimum, it can produce a relatively near-optimal solution within finite computational resources and time constraints. Because of the rapid temperature decrease in exponential cooling, the algorithm initially shows fluctuations in A-values at the beginning of the iterations. However, as the number of iterations increases, the acceptance rate for larger A-values declines quickly, and algorithm tend to avoid accepting higher A-values. Details are shown in the Figure 4.7 and Figure 4.5

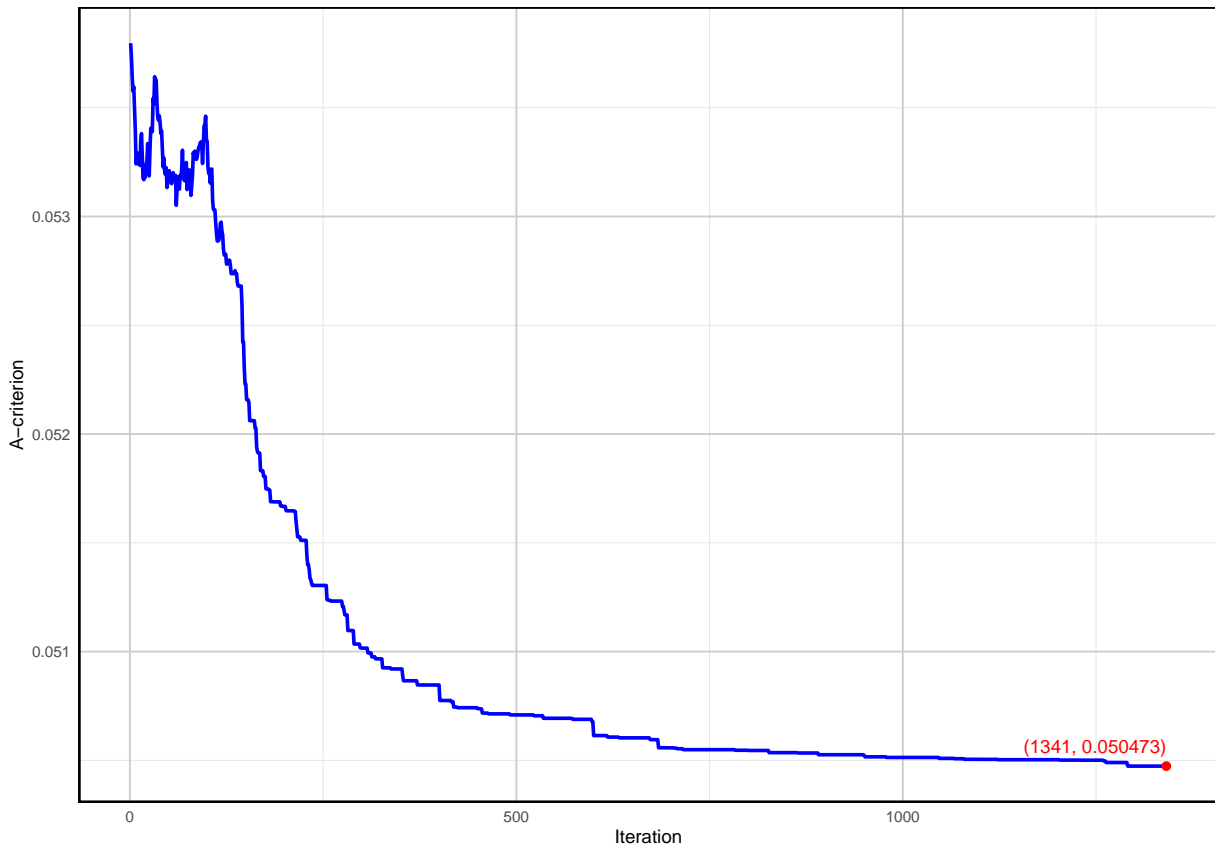


Figure 4.6: A v.s. Iteration (SA with exp cooling schedule)

The iteration process stopped at the 1341-st iteration because the A-value had come sufficiently close to the blockdesign A-value. The results show that the A-value achieved is $A_{SAexp} = 0.050473$, with an NB statistic $C_{SAexp}(\mathcal{D}_{rs}) = 4$, and ED statistics

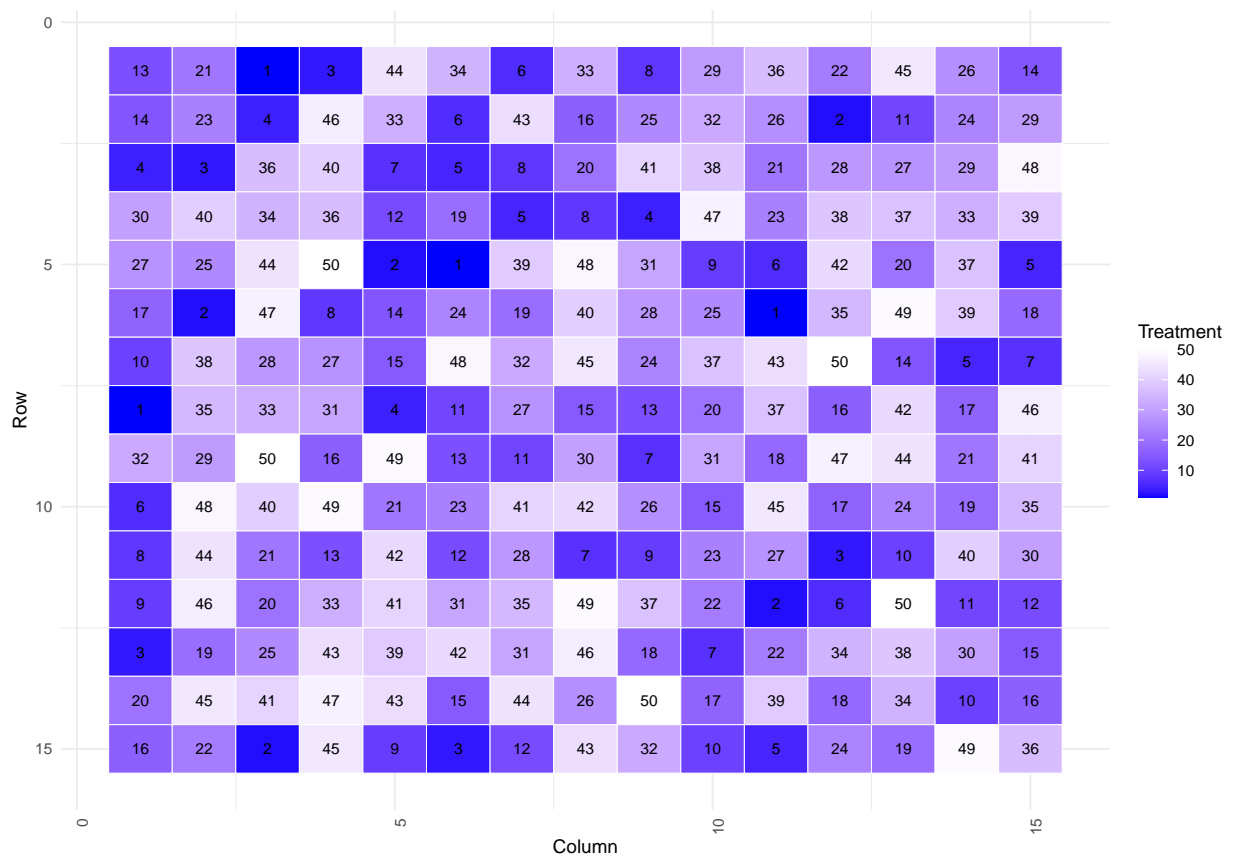


Figure 4.7: Detailed assignment for treatments given by SA with exp cooling schedule

$MRS(\mathcal{D}_{SAexp}) = 5$ and $MCS(\mathcal{D}_{SAexp}) = 6$.

The exponential cooling schedule achieved similar A-values and NB statistics to those of random selection. However, its performance on the ED statistic was less effective compared to both random selection and log cooling. The reason may be the early termination of iterations in exponential cooling, which limited the exploration of the whole design matrix space.

In contrast, random selection evaluates multiple permutations (equal to the step length s) in each iteration, which allows it to consider a broader range of potential solutions. Meanwhile, log cooling slows down the convergence, allow it has sufficient iterations to explore the solution space, although it selects only one neighbour at a time.

4.2.2 Analysis by Algorithm

Now we begin to explore the relationship between function runtime and the number of rows, columns, and treatments. Generally, runtime is proportional to the number of iterations. Here, we have set the maximum number of iterations to 2000. If in the simulation most runs reach the maximum number of iterations, then evaluating the relationship between runtime and rows, columns, and treatment numbers will no longer be meaningful. Therefore, before the analysis, we need to examine the distribution of iteration numbers. If most of the simulations do not reach the maximum number of iterations, we can use runtime as a measure of computational efficiency.

For random selection, many simulations did not reach the maximum number of iterations, which can be observed in Figure 4.8.

So we present the relationship between runtime and the number of rows under different treatment numbers, as shown in the figures below. As mentioned earlier, we have three replications for each combination of rows, columns, and treatment numbers. We calculate the average of the three replications for each combination as the final result.

We can observe that, under different numbers of columns, the relationship between rows and runtime tends to be similar. Here, we will not discuss the origin or significance of this trend but instead focus on the relation between variables and runtime. There is no clear positive or negative relationship between the number of rows and runtime. We see that as the number of rows increases, the runtime does not consistently increase or decrease, which is related to the randomness of random selection. During the iteration process, we might be lucky enough to find a good

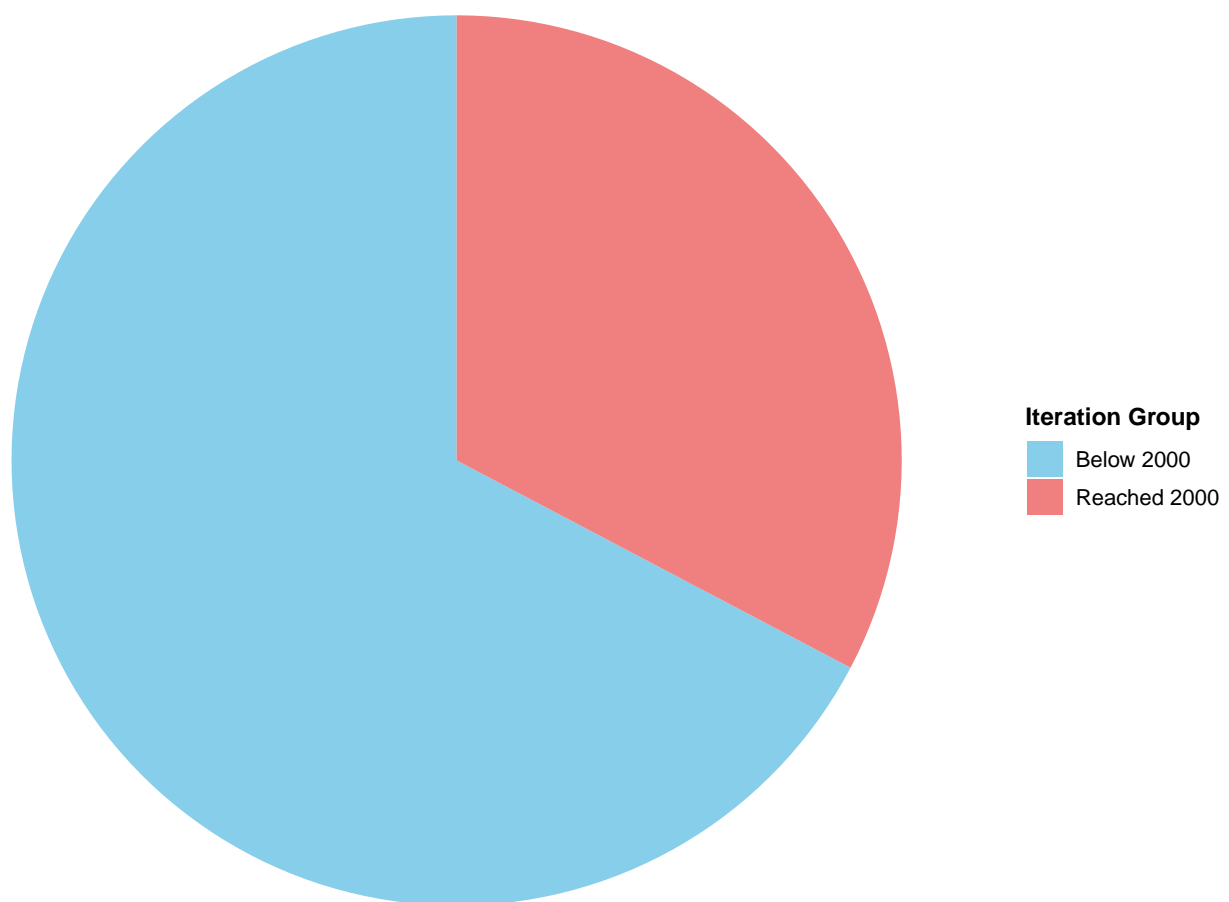


Figure 4.8: Proportion of Iteration number for random selection

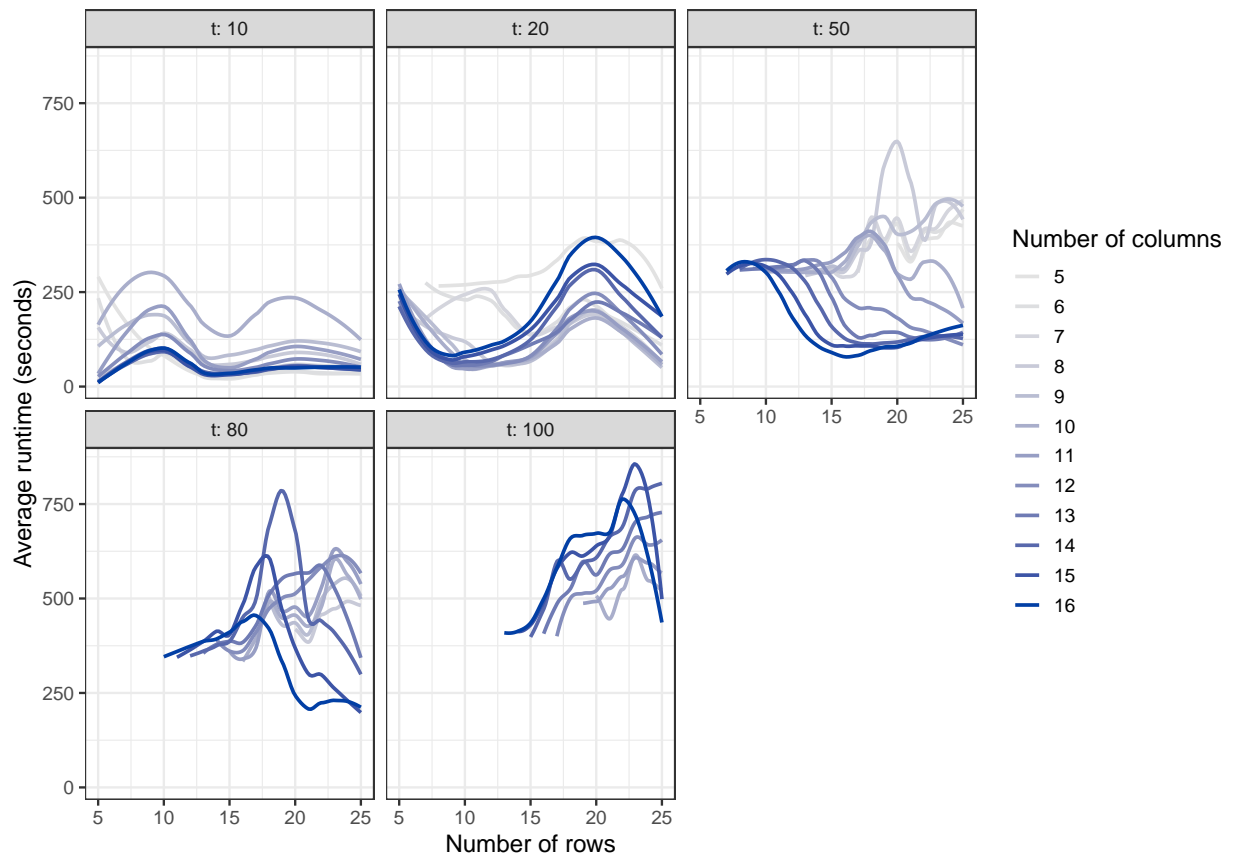


Figure 4.9: The above plot shows the average runtime by the number of rows with the color indicating the number of columns and faceted by the number of treatments t for Random Selection algorithm.

design early and terminate the iteration, or the A-value may gradually decrease over the iterations.

However, by observing the vertical axis of different subplots, we can see that as the number of treatments increases, the range of runtime fluctuations also increases. To demonstrate this, we present the relationship between the number of treatments and runtime under different rows and columns as Figure 4.10 shows.

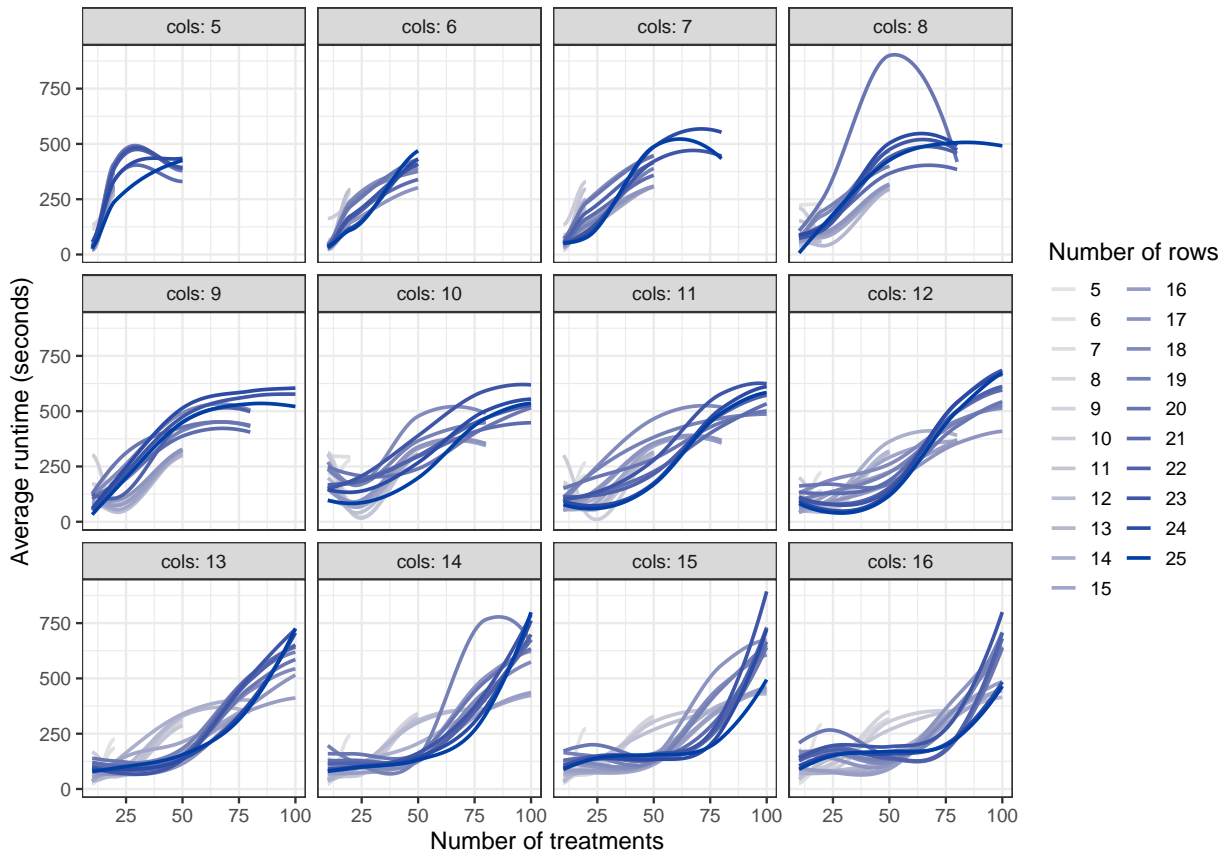
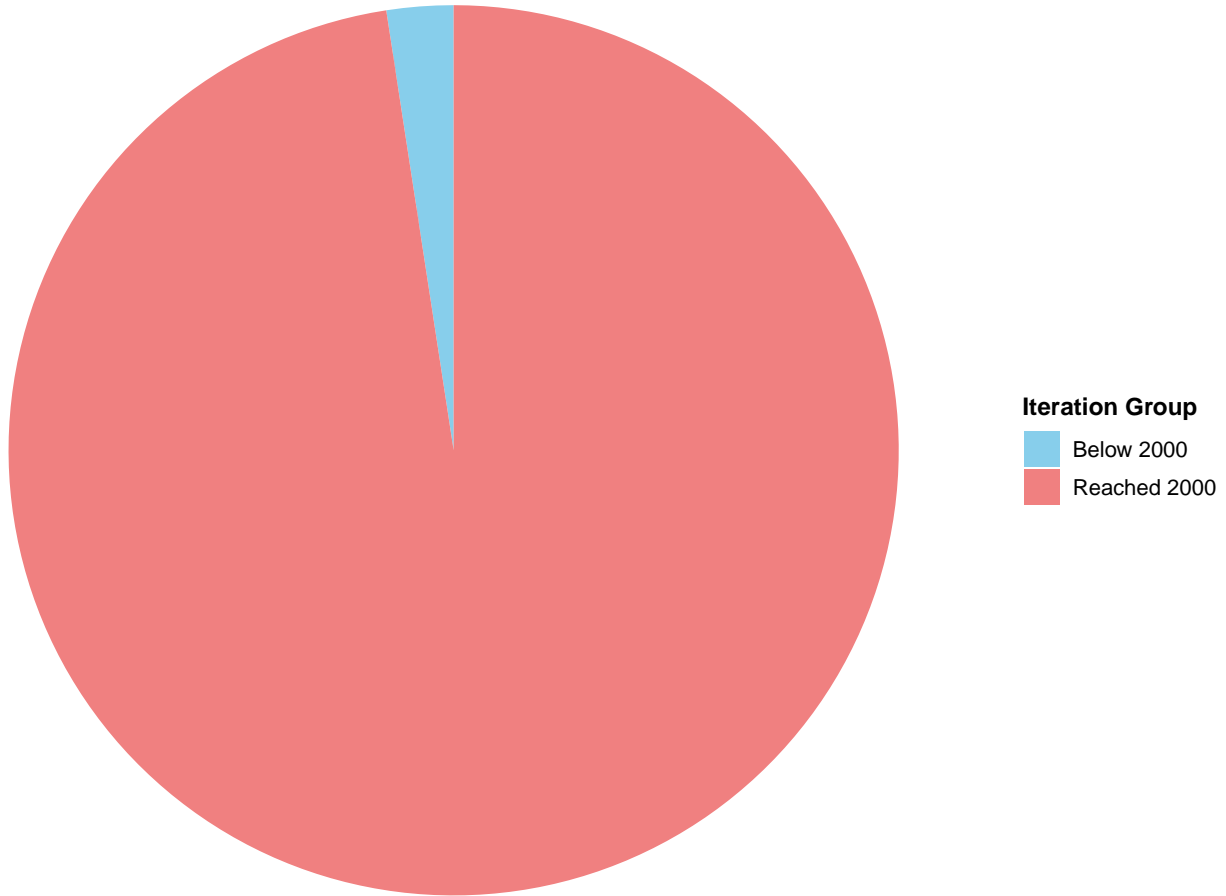


Figure 4.10: The above plot shows the average runtime by the number of treatments with the color indicating the number of rows, faceted by the number of columns for Random Selection algorithm.

By observing the figure, we can see that as the number of treatments increases, runtime shows a positive correlation, indicating that with more treatments, random selection requires more time to find permutations and make comparisons.

This approach leads us to consider the relationship between the number of treatments and runtime. However, for SA with log cooling, due to the slow convergence of log cooling and the limitation of a maximum number of iterations, the algorithm often runs until the maximum number of iterations is reached. This makes it difficult to observe the relationship between the number of treatments and runtime, as runtime is often related to the maximum number of iterations. See Figure 4.13

below for details.



In the figure, we can see that only when the number of rows and columns is small, meaning the design space is limited, SA with log cooling can terminate before reaching the maximum number of iterations. In most other cases, the algorithm stops when it reaches the maximum number of iterations. Therefore, we use the difference $d = (A_{SAlog} - A_{op})/A_{op}$ between the A-value obtained by SA with log cooling at the maximum number of iterations and the A-value calculated by the blockdesign function to evaluate the impact of the number of rows, columns, and treatments on the efficiency of the algorithm.

We first look at the influence of the number of rows on the distance. Similar to the analysis method used for random selection, we provide Figure 4.11.

From the above figure, it is evident that the number of rows and columns is positively correlated with the difference. The greater the number of rows, the larger the difference. Similarly, the greater the number of columns, the higher the lines (towards the red), indicating a larger difference. Based on previous experience, we attempt to examine the relationship between the number of treatments and distance, as shown in the figure below.

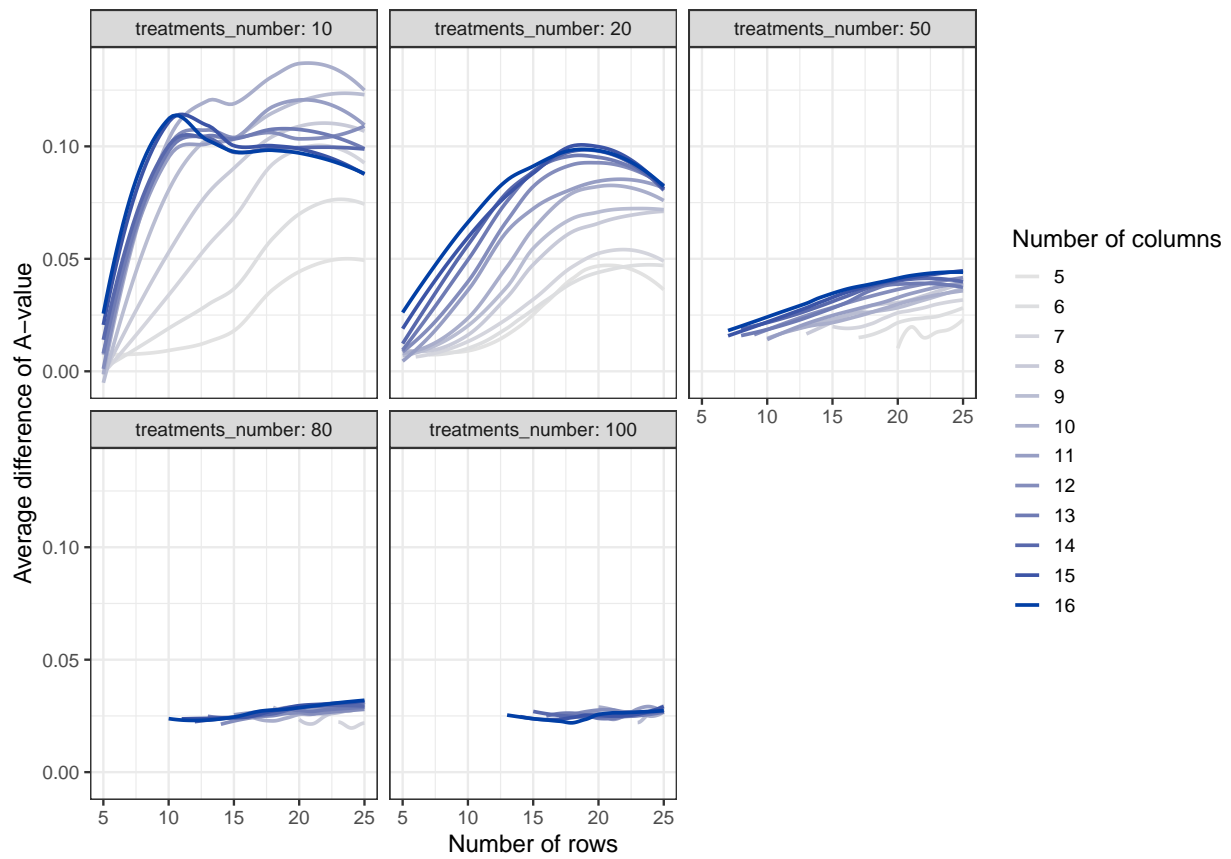


Figure 4.11: The above plot shows the difference between A_{SAlog} and A_{op} by the number of rows with the color indicating the number of columns and faceted by the number of treatments for SA log Cooling algorithm.

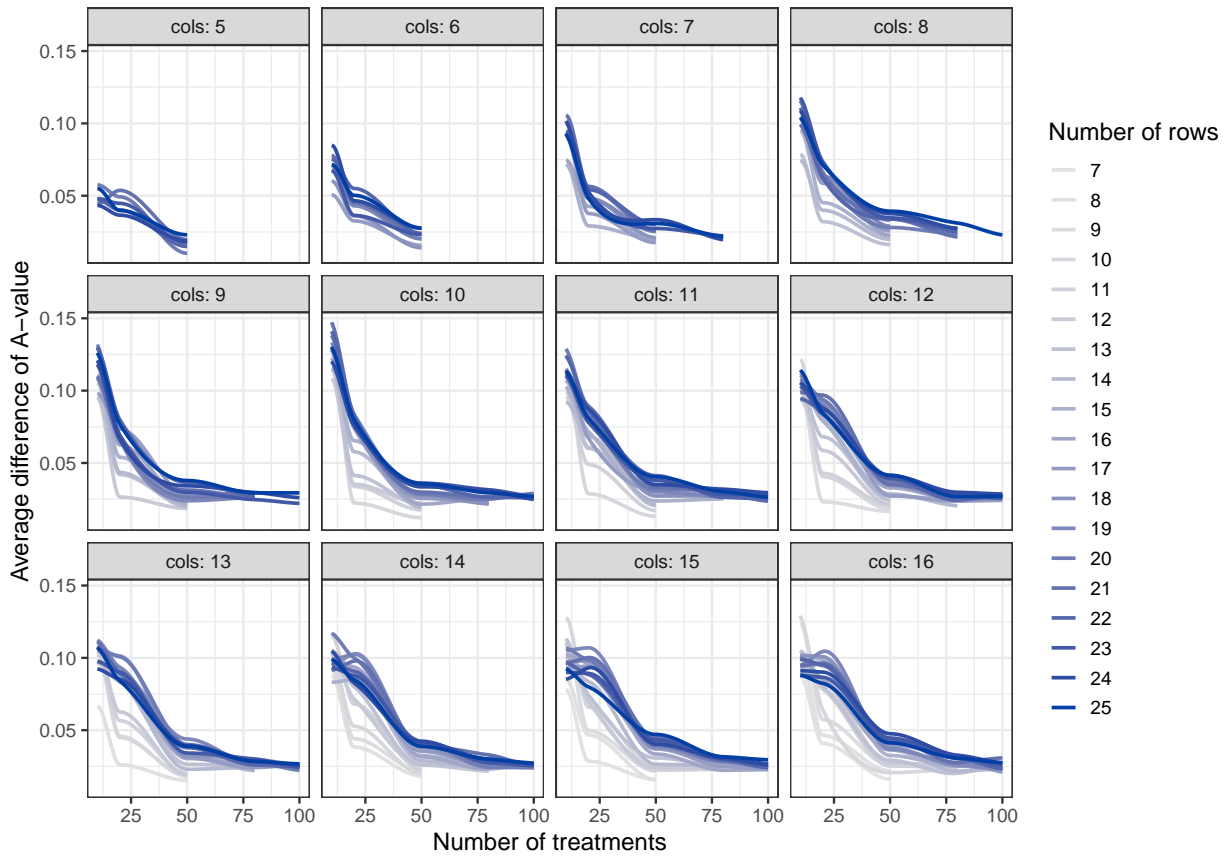


Figure 4.12: The above plot shows the difference between A_{SAlog} and A_{op} by the number of treatments, with the color indicating the number of rows and faceted by the number of columns for SA with log cooling algorithm.

The results show that the treatment number is negatively correlated with the difference. In other words, as the treatment number increases, the resulting value (whether the iteration stops early or reaches the maximum number of iterations) tends to be closer to the optimal value. However, this does not necessarily mean that the absolute distance to the optimal value decreases.

For SA with exponential cooling, we also examine the distribution of the iteration numbers.

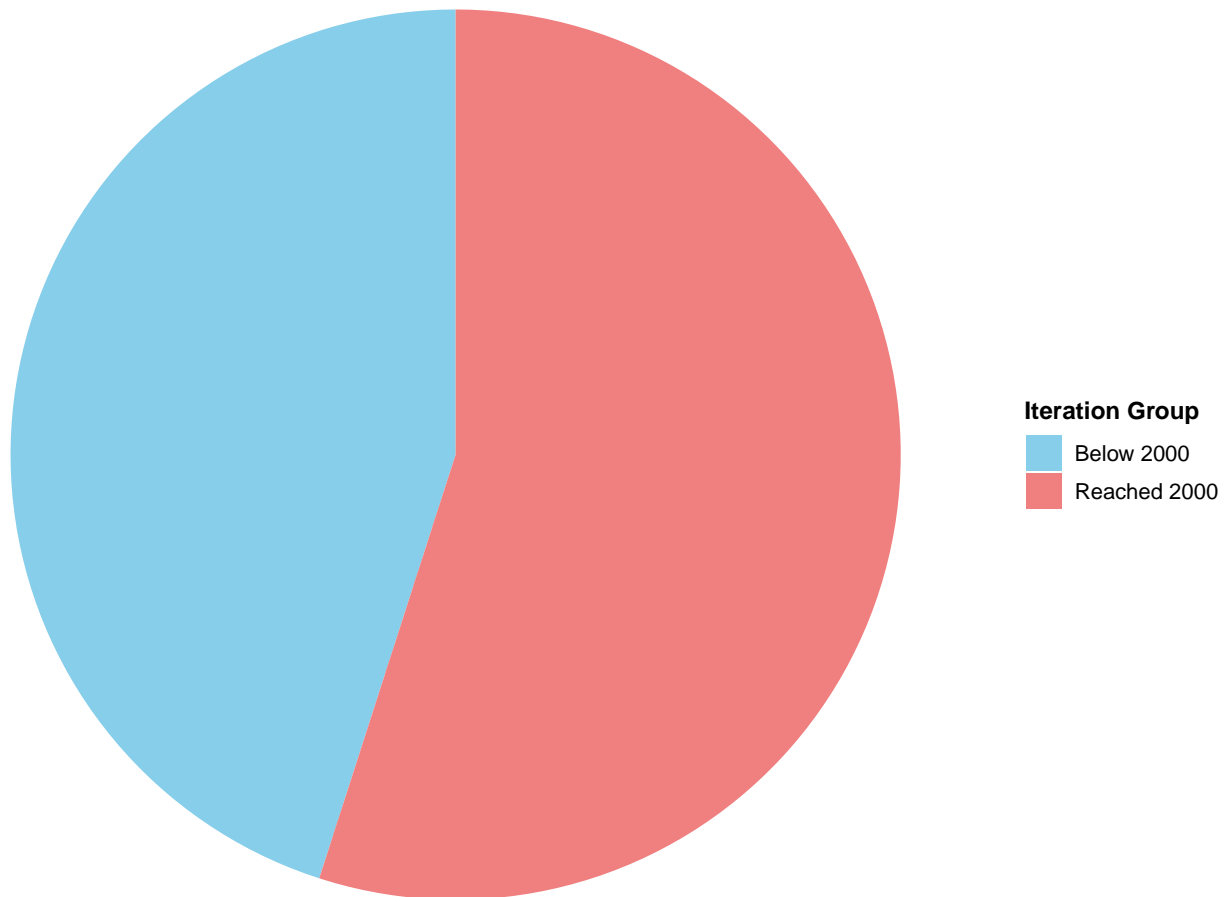


Figure 4.13: Proportion of Iteration number for SA with exp cooling schedule

It can be seen that most of the simulations did not reach the maximum number of iterations, so we can use runtime to compare the effects of rows, columns, and treatment numbers on the algorithm. Details in Figure 4.14

We can see that when the treatment number is small, the influence of rows and columns on runtime is not significant. This could be because the design space is smaller, and for randomness, the algorithm stops after fewer iterations. As the treatment number increases, the impact of rows and columns on runtime becomes apparent: the greater the number of rows and columns, the longer the runtime. This effect can also be observed when examining the influence of the treatment number

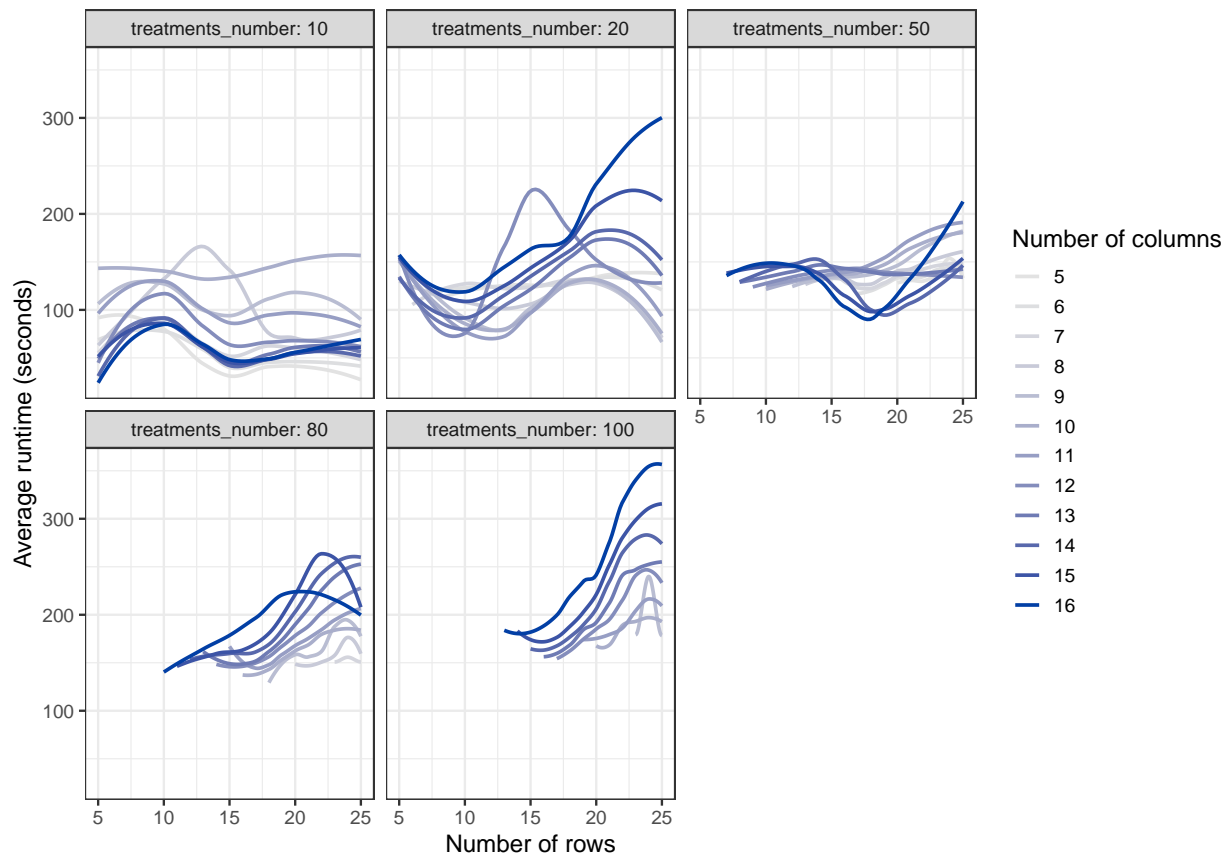


Figure 4.14: The above plot shows the runtime by the number of rows with the color indicating the number of columns and faceted by the number of treatments for SA with exp cooling algorithm.

on runtime.

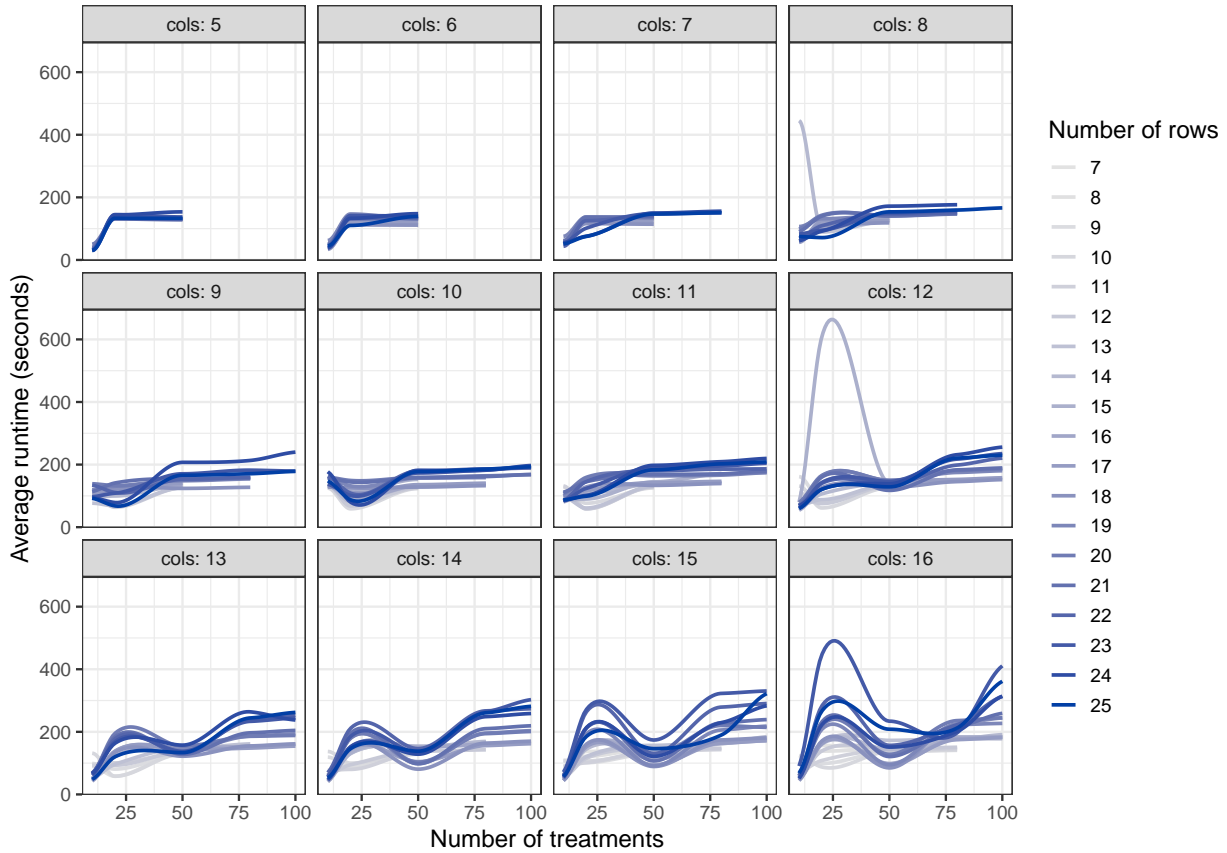


Figure 4.15: The above plot shows the runtime by the number of treatments with the color indicating the number of rows and faceted by the number of columns for the filtered 'SA exp Cooling' algorithm.

Similarly, we can observe that as the treatment number increases, the runtime also increases. Additionally, as the number of rows increases, the vertical axis of the plot also grows, indicating their positive correlation.

4.2.3 Comparison Between Algorithms

Now we compare the effectiveness of different algorithms. To simplify the process, we focus on observing the impact of the number of rows, columns, and treatments on the average runtime and the average difference under equivalent conditions. For example, when comparing the effect of the number of rows on the difference for different algorithms, we take the average of the difference for different column counts and treatment numbers at the same row level to simplify the visualization and makes the comparison more intuitive. Before making comparisons, it is important to note that, for our computational resource limitations, we have limited the maximum iteration number to 2000. Therefore, our comparison cannot determine

which algorithm is better. Therefore, it aims to compare the performance of different algorithms under the same computational constraints. Here, we compare the effects of row number, column number, and treatment number on the difference and runtime across different algorithms.

Comparing with difference

Now, we use the difference to compare the computational efficiency of different algorithms. We present the figures showing the influence of row number, column number, and treatment number on the difference for the three algorithms. Note that for random selection and SA with exponential cooling, the method for calculating the difference is the same as for SA with logarithmic cooling, which means it represents the relative difference, that is $d = (A_{op} - A_{result})/A_{op}$.

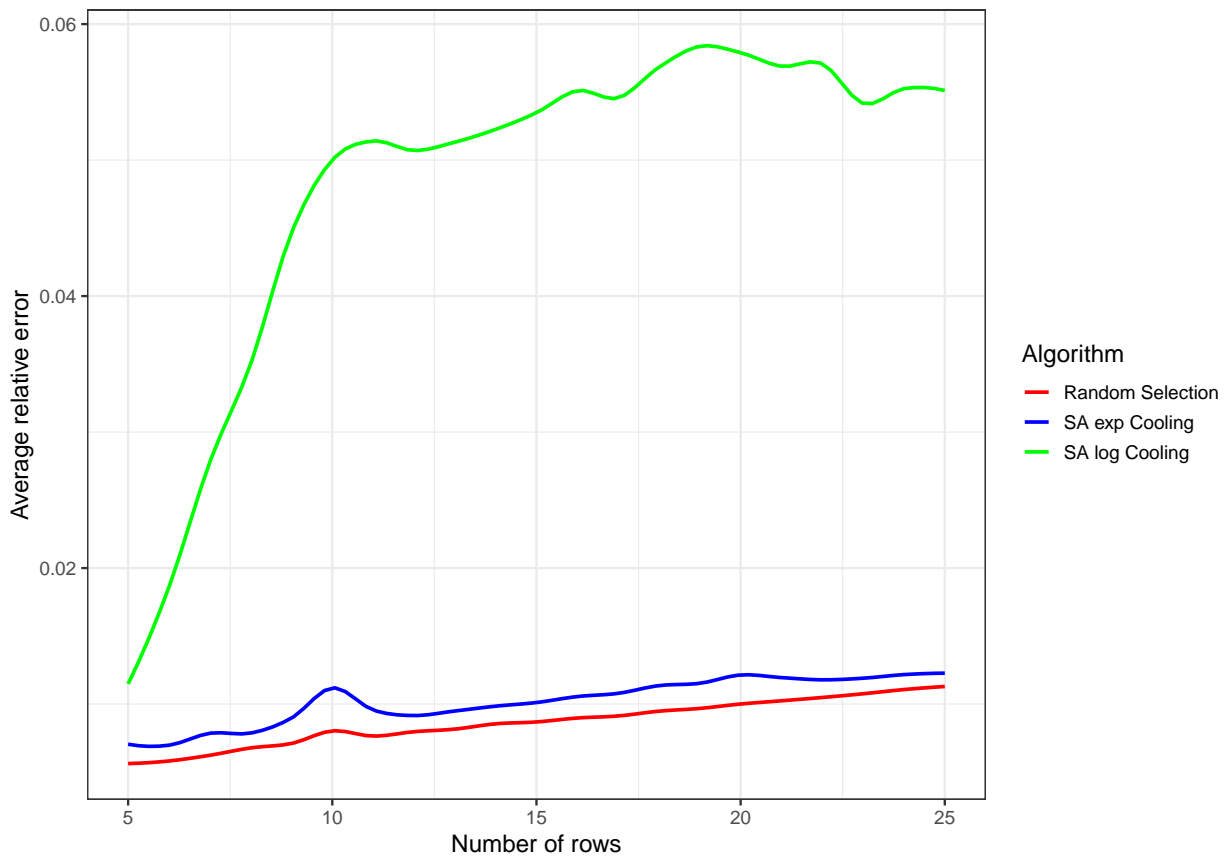


Figure 4.16: The average relative difference between A_{op} and A_{result} by the number of rows for different algorithms, with the lines representing different algorithms.

Overall, it can be observed that for the difference, random selection has the smallest value, SA with exponential cooling is slightly larger, and SA with logarithmic cooling has the largest value. Moreover, SA with logarithmic cooling is more influenced by the three variables.

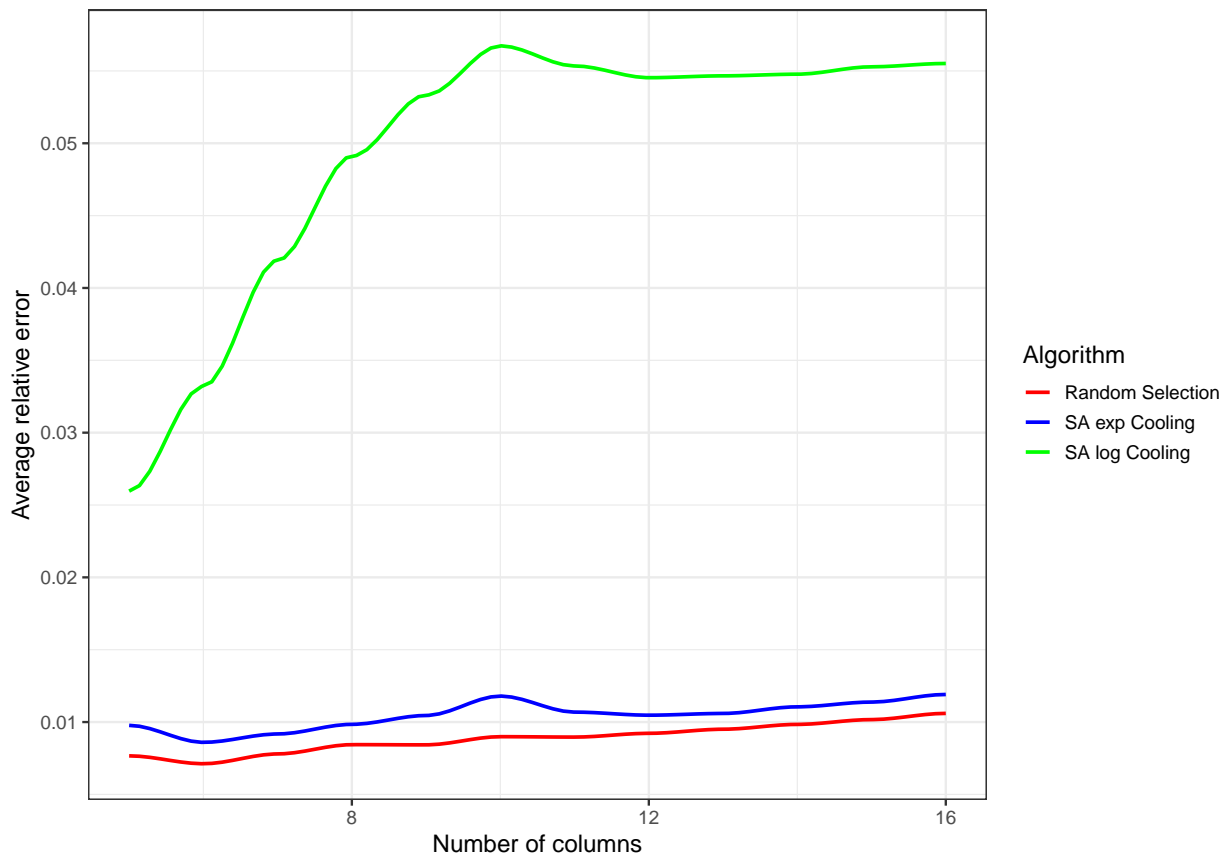


Figure 4.17: The average relative difference between A_{op} and A_{result} by the number of columns for different algorithms, with the lines representing different algorithms.

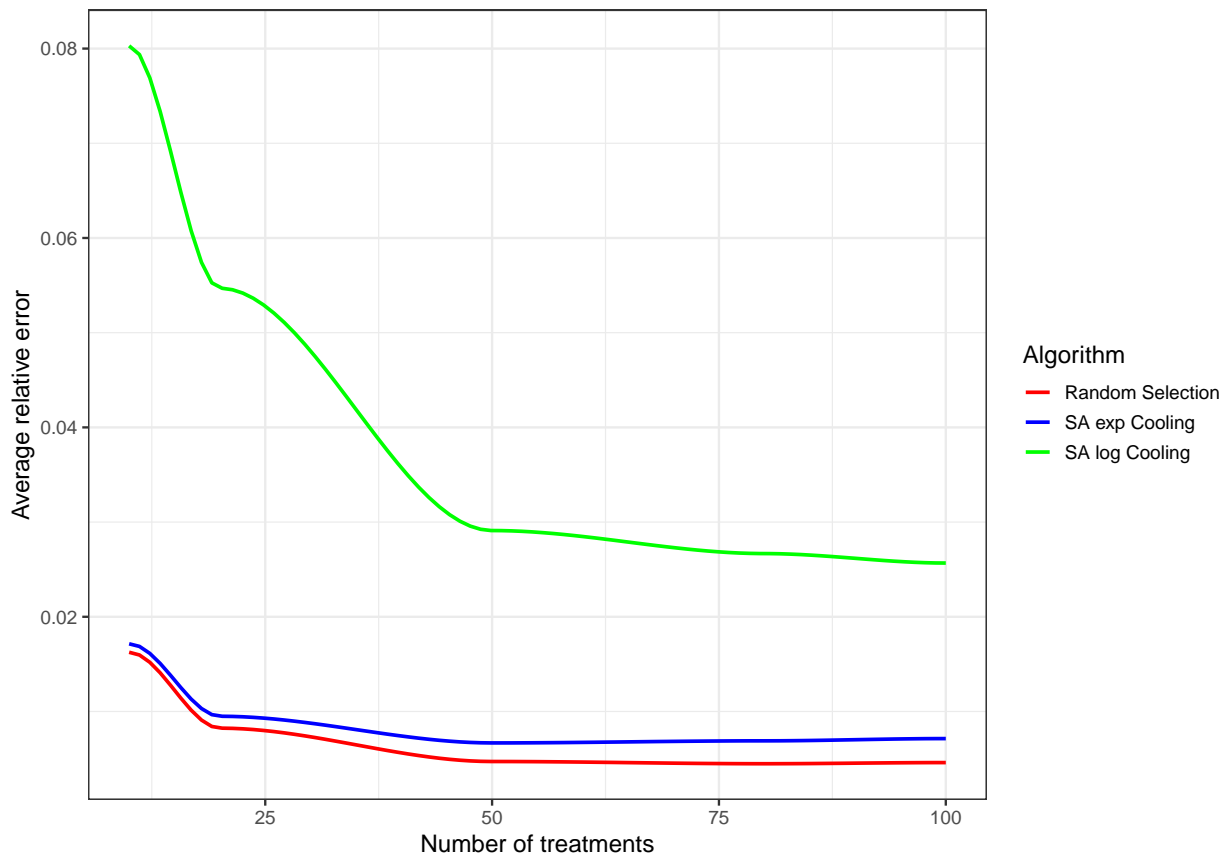


Figure 4.18: The average relative difference between A_{op} and A_{result} by the number of treatments for different algorithms, with the lines representing different algorithms.

In general, for all three algorithms, as the number of rows and columns increases, the relative difference also increases. However, the difference is negatively correlated with the treatment number. As mentioned earlier, a smaller relative difference does not necessarily mean that the absolute distance to the optimal value decreases. We can use the Figure 4.19 below to support this point.

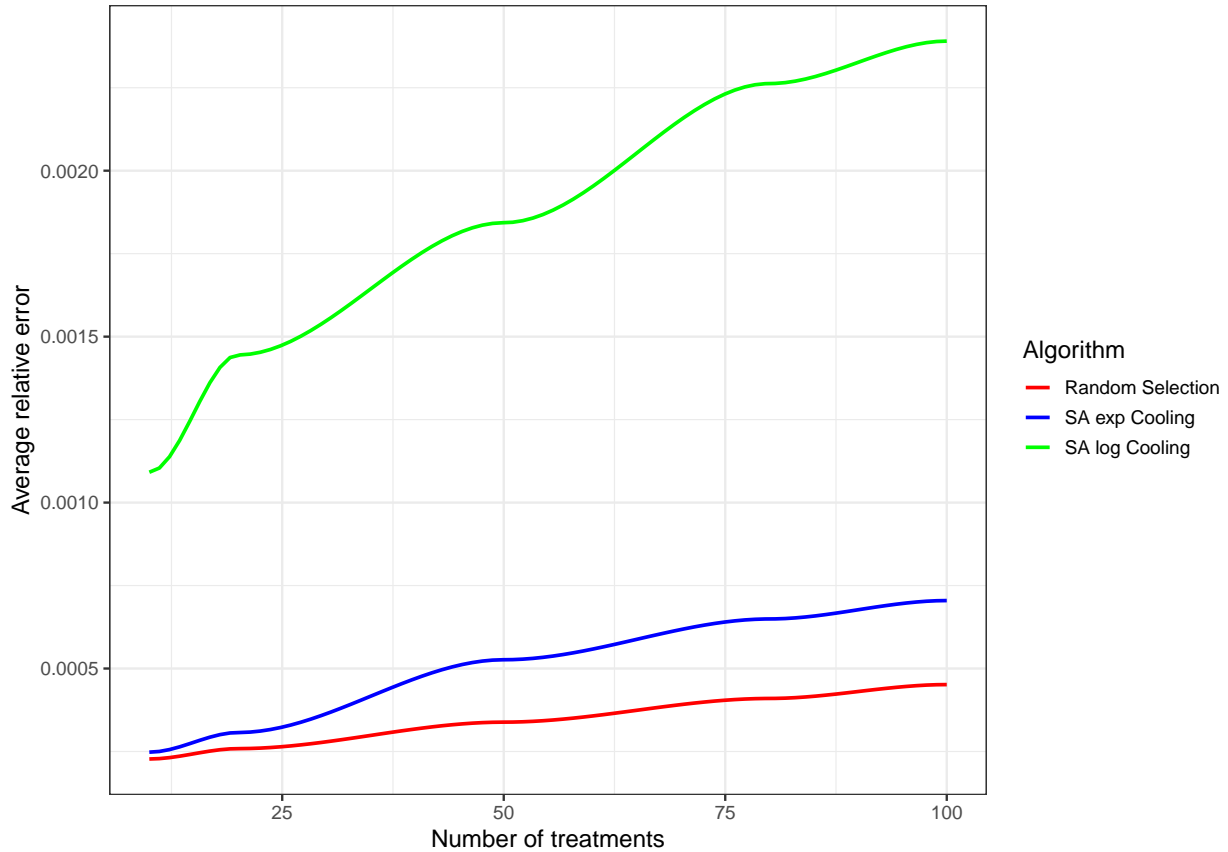


Figure 4.19: The average absolute difference between A_{op} and A_{result} by the number of treatments for different algorithms, with the lines representing different algorithms.

It can be seen here that if we consider the absolute difference, it is actually positively correlated with the treatment number.

Comparing with runtime

Now, let's see what the comparison looks like from the perspective of runtime. We present Figure 4.20, Figure 4.21, and Figure 4.22 to illustrate this.

From the figures, we can see that SA with exponential cooling is the fastest, while SA with log cooling takes a bit longer, and random selection requires the longest time. When the number of rows, columns, and treatments is still small, for the design space is limited and the randomness of the algorithms, the differences are

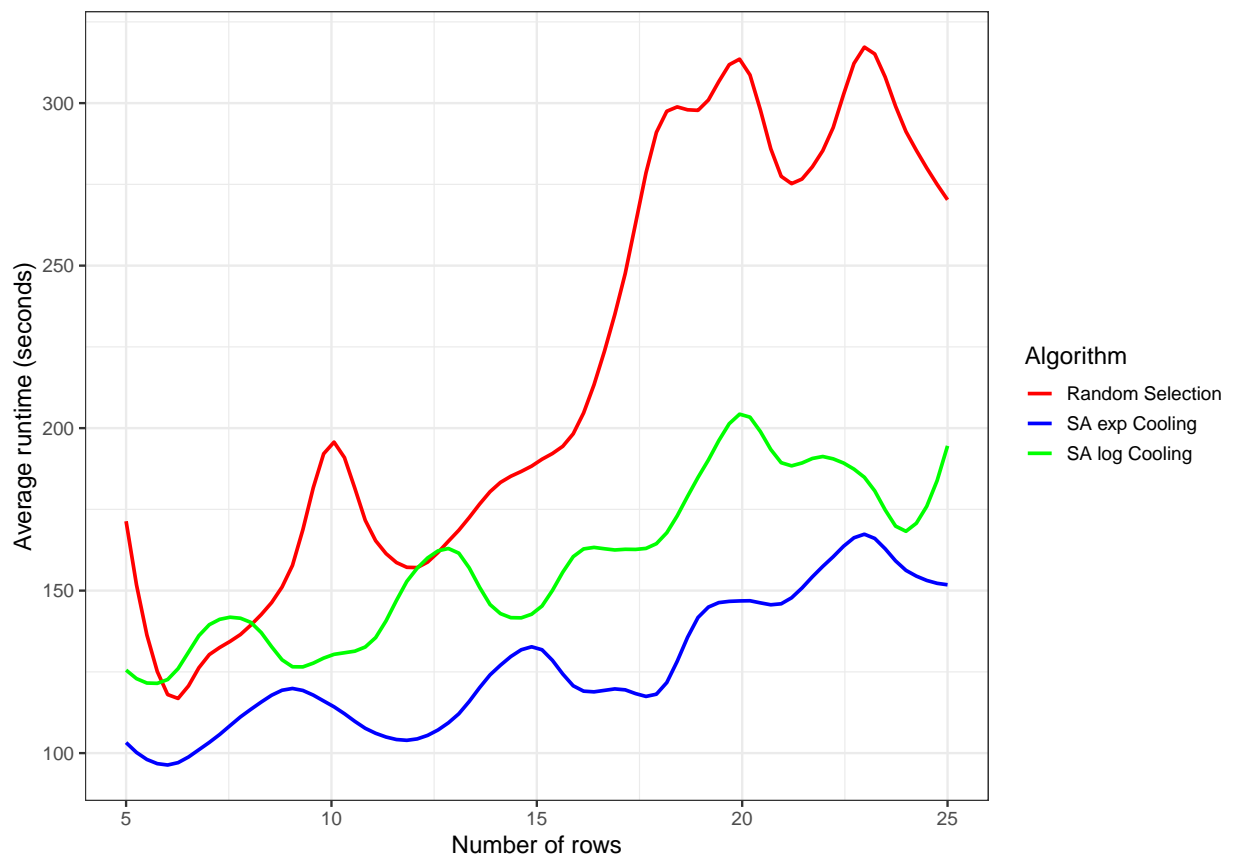


Figure 4.20: the average runtime by the number of rows for different algorithms, with the lines representing different algorithms.

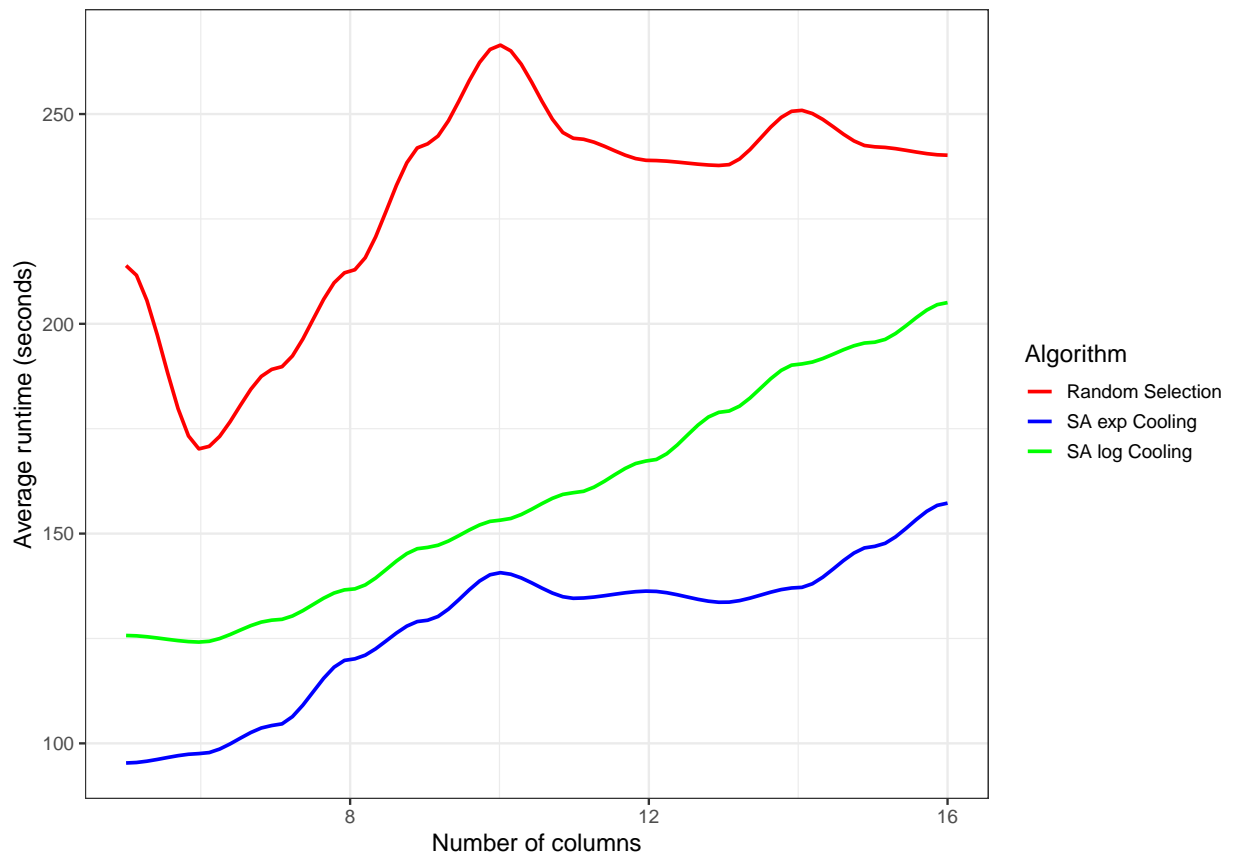


Figure 4.21: The average runtime by the number of columns for different algorithms, with the lines representing different algorithms.

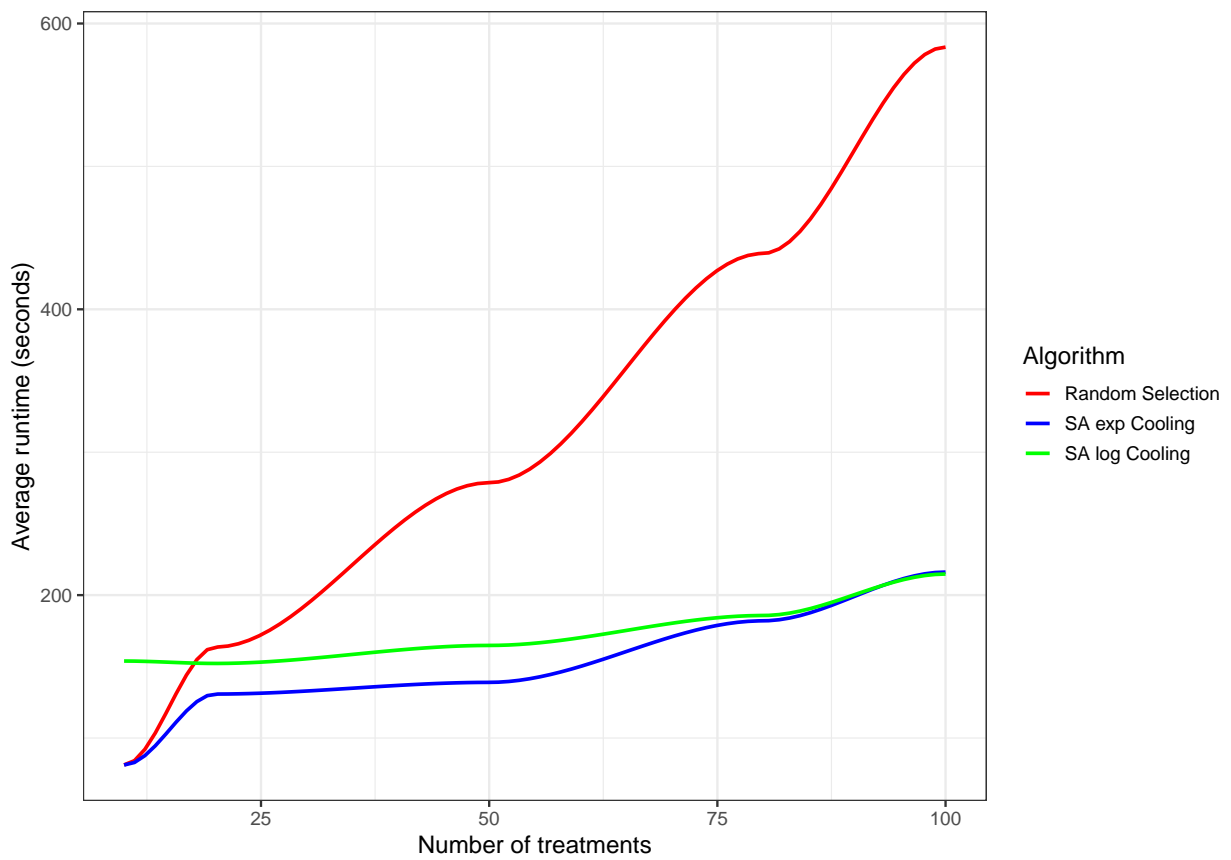


Figure 4.22: The average runtime by the number of treatments for different algorithms, with the lines representing different algorithms.

less noticeable. However, as these numbers increase, the differences in runtime become more significant, with random selection showing the fastest increase in runtime. Overall, the number of rows, columns, and treatments are all positively correlated with runtime, but the treatment number has a more direct impact on runtime.

4.2.4 Conclusion

Now, we summarize the computational performance of the three algorithms under constrained computational resources.

First, for random selection, as seen in the analysis above, this algorithm provides relatively accurate results but requires a longer runtime. When the number of treatments increases, the runtime significantly rises. In our simulations, the step-length was set to 3. Increasing the step-length would allow random selection to search the design space more extensively but would require more time.

Next is Simulated Annealing (SA). Overall, the iteration speed is faster compared to random selection. And increasing the number of rows, columns, and treatments requires more iterations and longer runtime. Although the log cooling schedule theoretically ensures convergence to the global optimum, as shown in the simulations above, it may require more iterations to reach the global optimum. On the other hand, the exponential cooling schedule converges faster, and it often ends iteration process early in the simulations. As mentioned earlier, while it cannot theoretically ensure convergence to the global optimum, it can provide an approximately optimal solution within a relatively short time frame under constrained computational resources and avoid local optima to some extent.

Chapter 5

Discussion

5.1 Limitations

In this section, we focus on examining the limitations and shortcomings of our algorithm, along with discussing the practical constraints observed in the experimental simulation results. Our analysis is organized into the following parts: computational efficiency and convergence Rate, balancing multi-objective optimization, and limitations and generalizability in practical applications.

5.1.1 Computational efficiency and convergence rate

Based on the line plots from the previous section, we observe that the convergence rate of our three algorithms significantly slow down as they approach the optimal solution. This often results in giving a near-optimal solutions, although it performs reasonably well within the limited number of iterations M in our simulation. For iterative methods, the algorithm actually explores the entire design matrix space by evaluating permutations of the matrix. This causes our algorithms, especially random selection, to have long runtimes when a large number of iterations is performed. For certain fixed design assumptions, such as the diagonal G matrix and R matrix we used previously, an algorithm that optimizes based on matrix structure could improve efficiency in such cases.

5.1.2 Balancing multi-objective optimization

We aim to optimize A while avoiding bad NB and ED statistics, which essentially represents a multi-objective optimization problem. In our current approach, we incorporate filtering steps during random search permutation generation and SA

neighbour generation to ensure that NB and ED improve gradually throughout the iterations. Although this approach allows for the stepwise optimization of all three statistics, it fails to bring in the correlations among them and lacks the ability to balance and trade off between the three. Using a multi-objective optimization method would allow us to place these three statistics on an equal priority level. In our current approach, however, the A-value has consistently been the primary variable driving changes in the design matrix.

5.1.3 Limitations and generalizability in practical applications

When comparing the computational performance of different algorithms, we mentioned that such comparisons cannot determine which algorithm is better. For a deeper exploration of the performance of different algorithms, a controlled comparison is needed. For example, setting a common tolerance and allowing all algorithms sufficient time or iteration count to complete, which would enable a fair comparison of runtime or iteration numbers. Our analysis of the algorithms is based on a limited set of constraints, observing and comparing the behaviour of different algorithms under some conditions.

When discussing model errors, we mentioned using the R matrix for modelling. For independent plots, using linear models and the R matrix is very convenient. In Piepho et al. [2018], the *AR1* model is suggested as a way to address cases where plots are correlated. We have not attempted or analysed algorithms using this type of model.

5.2 Future Directions

5.2.1 Penalty objective function

Incorporate a penalty term into the objective function. When adjacent treatments in the design are the same, this penalty term increases the value of the objective function, making the optimization process more likely to avoid having the same treatment in adjacent positions.

For example, we can define a following penalty function. For example we have a design matrix X , the penalty for self-adjacency is

$$f(X) = \sum_{(i,j) \in X} I(X_i = X_j)$$

Here i and j are different plots next to each other in the experiment field, and $I(X_i = X_j)$ is an indicator function equals to 1 if i plot and j plot have the same treatment.

So basing on this penalty function, For a certain design X we change our function into:

$$F(X) = \bar{f}_A^{RC} + \lambda \cdot f(X)$$

here $\lambda < 0$

Or use A-criteria :

$$G(x) = tA + (1 - t) \cdot f(X)$$

here $0 \leq t \leq 1$ and we minimize it

Usually we have such mathematical programming of inequality constrained optimization problem: we minimize objective function $f_0(x)$ with inequality constraints $f_i(x) \leq 0, i \in I = \{1, 2, \dots, m\}$. And we have a well-known penalty function for this problem is

$$F(x, \rho) = f_0(x) + \rho \sum_{i \in I} \max\{f_i(x), 0\}$$

and a corresponding constrained penalty optimization problem is to minimize penalty function $F_2(x, \rho)$, detailed information in Meng et al. [2013]

5.2.2 Algorithm improvment

SA-based multiobjective optimization algorithms

When discussing the A-value, NB statistic, and ED statistic, our goal is actually to find a process that can simultaneously optimize all three, or finding a balance between three statistic criteria. SA is often used in combinatorial optimization problems, as we are using it here to optimize the arrangement of treatment plots. Suman and Kumar [2006] claim that in recent studies, SA has been applied in many multi-objective optimization problems, because of its simplicity and capability of producing a Pareto set of solutions in single run with very little computational cost.

We say a solution is non-dominated if none of its objective values can be improved without worsening at least one other objective. The Pareto set (or Pareto front) is a set of non-dominated solutions in a multi-objective optimization problem. Suman and Kumar [2006] introduce several SA for multi-objective optimization.

SA-based multi-objective optimization algorithms given by Suppapitnarm et al.

[2000] is a promising approach. Instead of Equation 3.2 and using permutation filtering, they give probability step write as this form

$$P = \min(1, \prod_{i=1}^N \exp\{\frac{-\Delta s_i}{T_i}\})$$

Here N is the number of objective functions, and for each objective function i , Δs_i is the difference of objective function between two solution and T_i is the current temperature. Control the optimization rates of different objective functions by setting different cooling schedules for each.

Memory-Based Optimization Algorithms - TABU search

Same with SA, Tabu Search is an optimization algorithm used for solving combinatorial problems. It is a type of local search method that enhances basic hill-climbing algorithms by using memory structures to avoid revisiting previously explored solutions, introduced in Butler et al. [2013]. This helps the search process escape from local optima and encourages exploration of new areas in the solution space. Tabu Search maintains a tabu list, a short-term memory that keeps track of recent moves or solutions that should not be revisited for a certain number of iterations. This is mathematically represented as a set \mathcal{T} where recent solutions X_t are stored for a fixed period k iterations:

$$\mathcal{T} = \{x_t | t \in [t - k, t)\}$$

If a solution $x' \in \mathcal{T}$, x' is considered “tabu” and cannot be revisited. Therefore, besides our two approaches here—random search with step-length and SA that probabilistically accepts worse objective function values—this type of short-term memory-enhanced algorithm (Memory-Enhanced Algorithms) helps ensure efficient computation, minimizing resource consumption while maximizing exploration within the solution space and helping escape local optima.

References

- Emile Aarts and Jan Korst. *Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing*. John Wiley & Sons, Inc., 1989.
- JJ Allaire. *quarto: R Interface to 'Quarto' Markdown Publishing System*, 2023. URL <https://CRAN.R-project.org/package=quarto>. R package version 1.3.
- Dimitris Bertsimas and John Tsitsiklis. Simulated annealing. *Statistical science*, 8 (1):10–15, 1993.
- David Butler. On the optimal design of experiments under the linear mixed model. 2013.
- David Geoffrey Butler, Alison B Smith, and Brian R Cullis. On the Design of Field Experiments with Correlated Treatment Effects. 19(4):547–557. ISSN 1085-7117, 1537-2693. doi: 10.1007/s13253-014-0191-0. URL <http://link.springer.com/10.1007/s13253-014-0191-0>.
- DG Butler, AB Smith, and BR Cullis. On model based design of comparative experiments, 2013.
- Rodney N Edmondson. Multi-level block designs for comparative experiments. *Journal of Agricultural, Biological and Environmental Statistics*, 25(4):500–522, 2020.
- Arthur R Gilmour, Robin Thompson, and Brian R Cullis. Average Information REML: An Efficient Algorithm for Variance Parameter Estimation in Linear Mixed Models. 51(4):1440–1450. doi: 10.2307/2533274.
- Charles Roy Henderson, Oscar Kempthorne, Shayle Roy Searle, and C. M. Krosigk. The Estimation of Environmental and Genetic Trends from Records Subject to Culling. 15(2):192–218.
- D L Johnson and Robin Thompson. Restricted Maximum Likelihood Estimation of Variance Components for Univariate Animal Models Using Sparse Matrix Techniques and Average Information. 78(2):449–456. ISSN 00220302. doi:

- 10.3168/jds.S0022-0302(95)76654-1. URL <http://www.journalofdairyscience.org/article/S0022030295766541/fulltext>.
- Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.
- Zhiqing Meng, Chuangyin Dang, Min Jiang, Xinsheng Xu, and Rui Shen. Exactness and algorithm of an objective penalty function. *Journal of Global Optimization*, 56:691–711, 2013.
- H. Desmond Patterson and Robin Thompson. Recovery of inter-block information when block sizes are unequal. 58(3):545–554. ISSN 0006-3444, 1464-3510. doi: 10.1093/biomet/58.3.545. URL <https://academic.oup.com/biomet/article-lookup/doi/10.1093/biomet/58.3.545>.
- Hans-Peter Piepho, Volker Michel, and Emlyn Williams. Neighbor balance and evenness of distribution of treatment replications in row-column designs. *Biometrical Journal*, 60(6):1172–1189, 2018.
- G K Robinson. That BLUP is a Good Thing : The Estimation of Random Effects. 6: 15–32. URL <http://www.jstor.org/stable/2245695>.
- Galen H Sasaki and Bruce Hajek. The time complexity of maximum matching by simulated annealing. *Journal of the ACM (JACM)*, 35(2):387–403, 1988.
- Alison B Smith. Multiplicative mixed models for the analysis of multi-environment trial data.
- Balram Suman and Prabhat Kumar. A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the operational research society*, 57(10):1143–1160, 2006.
- A Suppakitnarm, Keith A Seffen, Geoff T Parks, and PJ Clarkson. A simulated annealing algorithm for multiobjective optimization. *Engineering optimization*, 33(1):59–85, 2000.
- Arnas P Verbyla. A conditional derivation of residual maximum likelihood. 32:227–230. ISSN 0004-9581. doi: 10.1111/j.1467-842X.1990.tb01015.x. URL <https://onlinelibrary.wiley.com/doi/10.1111/j.1467-842X.1990.tb01015.x>.
- E.R. Williams and J.A. John. Construction of crossover designs with correlated errors. 49(1):61–68. ISSN 13691473. doi: 10.1111/j.1467-842X.2006.00463.x. URL <https://onlinelibrary.wiley.com/doi/10.1111/j.1467-842X.2006.00463.x>.

Appendix A

Tools

This thesis was written using Quarto 1.4.550 [Allaire, 2023] and the following system and R packages:

```
- Session info -----
setting  value
version  R version 4.4.1 (2024-06-14 ucrt)
os       Windows 11 x64 (build 22631)
system   x86_64, mingw32
ui       RTerm
language (EN)
collate   English_United States.utf8
ctype    English_United States.utf8
tz        Australia/Sydney
date      2024-10-23
pandoc    3.1.11 @ C:/Program Files/RStudio/resources/app/bin/quarto/bin/tools/ (

- Packages -----
package      * version date (UTC) lib source
cli          3.6.3   2024-06-21 [1] CRAN (R 4.4.1)
digest       0.6.36  2024-06-23 [1] CRAN (R 4.4.1)
evaluate     0.24.0  2024-06-10 [1] CRAN (R 4.4.1)
fastmap      1.2.0   2024-05-15 [1] CRAN (R 4.4.1)
htmltools    0.5.8.1 2024-04-04 [1] CRAN (R 4.4.1)
jsonlite     1.8.8   2023-12-04 [1] CRAN (R 4.4.1)
knitr        1.47    2024-05-29 [1] CRAN (R 4.4.1)
later        1.3.2   2023-12-06 [1] CRAN (R 4.4.1)
```

processx	3.8.4	2024-03-16	[1]	CRAN	(R 4.4.1)
ps	1.7.6	2024-01-18	[1]	CRAN	(R 4.4.1)
quarto	1.4.4	2024-07-20	[1]	CRAN	(R 4.4.1)
Rcpp	1.0.12	2024-01-09	[1]	CRAN	(R 4.4.1)
rlang	1.1.4	2024-06-04	[1]	CRAN	(R 4.4.1)
rmarkdown	2.27	2024-05-17	[1]	CRAN	(R 4.4.1)
rstudioapi	0.16.0	2024-03-24	[1]	CRAN	(R 4.4.1)
sessioninfo	1.2.2	2021-12-06	[1]	CRAN	(R 4.4.1)
tinytex	0.51	2024-05-06	[1]	CRAN	(R 4.4.1)
xfun	0.45	2024-06-16	[1]	CRAN	(R 4.4.1)
yaml	2.3.8	2023-12-11	[1]	CRAN	(R 4.4.1)

[1] C:/Users/yaoya/AppData/Local/R/win-library/4.4

[2] C:/Program Files/R/R-4.4.1/library
