UNIVERSITY OF CALIFORNIA

Los Angeles

Advancements in the Design and Analysis of Order-of-Addition Experiments

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Statistics

by

Zachary Shane Stokes

2021

ABSTRACT OF THE DISSERTATION

Advancements in the Design and Analysis of
Order-of-Addition Experiments

by

Zachary Shane Stokes
Doctor of Philosophy in Statistics
University of California, Los Angeles, 2021
Professor Hongquan Xu, Co-Chair
Professor Weng Kee Wong, Co-Chair

The ordering effect of sequential treatment addition is present in many problem areas ranging from online search ranking and job scheduling to election standardization and the design of clinical trials. The straightforward approach to studying this ordering effect on a response is to test each permutation of the treatment components. When the run size becomes too costly, a random sample of permutations can instead be used for estimation and inference. Noting the inefficiencies present in both of these approaches, well-designed order-of-addition experiments have been proposed to reduce the computational complexity of studying the ordering effect without sacrificing statistical power; however, this problem is under-studied in the statistical literature.

In this work we make several state-of-the-art advancements to the order-of-addition literature. By leveraging the structure of order-of-addition data, we pose flexible, position-based models that work well in practice. These new models inspire and necessitate the creation of parsimonious designs that guarantee stable parameter estimates. We further adapt these designs for a component screening problem in which the pool of treatment components is larger than the number of available positions in the sequence.

We test our methods against several prominent order-of-addition problems, including drug chemotherapy and job scheduling. Our models and model-based optimal designs outperform the accepted alternatives in most cases, and in other situations our methods act as valuable tools for encouraging robust experimentation. To encourage the use of quality order-of-addition designs in wider practice, we implement and compare popular, nature-inspired metaheuristic algorithms for solving difficult design problems without closed-form solutions. We find that certain variants and hybridized algorithms can produce quality designs for obtaining stable parameter estimates in order-of-addition models or for studying black-box deterministic functions on the space of permutations.

The dissertation of Zachary Shane Stokes is approved.

Yingnian Wu

Frederic R Paik Schoenberg

Weng Kee Wong, Committee Co-Chair

Hongquan Xu, Committee Co-Chair

University of California, Los Angeles

2021

To my parents . . .

for their unending love and support

and for teaching me that despite the odds

I can achieve anything by working hard and being kind.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This work would not have been possible without the guidance and encouragement of my advisors, Professor Hongquan Xu and Professor Weng Kee Wong. Their dedication to my research and continued education has made an invaluable impact on my academic career. At every critical juncture of my graduate studies, they have shared their time, wisdom and experience to ensure that I can make the most informed decision, while also allowing me the flexibility to choose my own path. With their support and willingness to share their expertise in the design of experiments, I have become a confident statistician, technical writer, presenter, experimenter and researcher.

I am also very grateful to have Professor Rick Schoenberg and Professor Yingnian Wu on my committee. Their classes provided critical training and exposed me to new and interesting applications of my research. Their helpful feedback on my dissertation and useful insights from their own projects allowed me to see my research from new perspectives.

I also owe a great deal of gratitude to the many other people who have taught, encouraged and supported me throughout my education. From my first day at UCLA I was determined to build my skills and reputation as a statitiscal consultant. With the guidance and support of Mahtash Esfandiari and Vivian Lew, I have accomplished this goal beyond what I ever thought possible. Being part of the Statistical Consulting Center has given me knowledge and opened me up to valuable experiences that I will continue to draw from in my personal and professional life as I mature as a consultant. I am grateful to have had the opportunity to work alongside them.

The Department of Statistics staff has been incredibly supportive during my time in the department. Specifically, I would like to thank Glenda Jones, Chie Ryu and Laurie Leyden for putting up with all of my wild requests and strange situations. Whether I was sorting out jobs in other departments or trying to secure housing, they were always willing to take the extra steps to advocate for me and to help me be successful. Their flexibility and steadfastness allowed me to take advantage of the many opportunities that UCLA has to offer.

One of the most impactful opportunities I have had during my time at UCLA was to work in the center for Inflammatory Bowel Diseases in the Department of Medicine. Through my collaboration with this group, I not only grew as an applied statistician, researcher and entrepreneur, but also made many friends. I would like to thank Dr. Daan Hommes and Dr. Aria Zand for continuing to support my career and for challenging me to become a well-rounded data scientist.

My success at UCLA would not have been possible without the mentorship and encouragement of the many statistics teachers and professors I have had along the way. I owe much appreciation to Mr. Jeff Lalaian, who nurtured my early interest in statistics and gave me the tools for critical thinking that I still use today. His passion for the subject shines through his instruction, and his desire to see students learn and succeed is undeniable. I am also grateful to Professor Abhyuday Mandal for being my mentor at the University of Georgia and for inspiring me to turn my interest in the design of experiments into a graduate career path. Throughout my education, his guidance has advanced my research and professional growth.

Finally, and most importantly, the love and encouragement of my family and friends has been a driving force behind my success. I am very fortunate to have a strong support network that is always there for me when I face challenging situations, have to make difficult decisions, or just need to center myself.

My time at UCLA has been greatly improved thanks to the friendship and support of my fellow researchers. Specifically, Lin Wang, Ye Tian, Alan Vázquez, Yuhao Yin, Jake Elmstedt, Onyambu Onyancha, Arjun Akula, and Erik Nijkamp have challenged and encouraged me to become a better statistician and have offered crucial advice about my career plans and research. I am also incredibly lucky to have a group of lifelong friends who have been by my side the last decade. Jeremy, Mary Cate, George, Kevin, Chip, Haley, Jeff and Katy, I could not be more thankful for the friendship we have. I cherish the memories we have formed and look forward to building more together soon.

Jessica, words can hardly express the gratitude I have for your unconditional love and endless support, patience and belief in me these last four years. The happiness and strength that you bring me has inspired me to do my best every day, even when things get tough. Despite what we have had to overcome to get here, there are wonderful opportunities and adventures ahead of us and I am so excited to share them with you. To the Peters family, thank you for opening your home to me. Having a west coast family has been so important in keeping me grounded and focused during this uncertain time of my life.

Ultimately, I would not be where I am today without the deep love and personal sacrifices of my parents. It is only fitting that I dedicate this work to them for believing in me every step of the way, even when it was hard for me to believe in myself. Dad, thank you for teaching me to push myself and for showing me that the most important things in life come from dedication and perserverance. Mom, it would take another dissertation to express my love for you. You will always be my biggest inspiration, and the kindness you have instilled in me is one of my greatest strengths. Tyler and Colton, thank you for supporting and challenging me in ways that only brothers can. You have both encouraged me to keep going and to always find humor and peace, even in difficult situations.

Chapter 3 is a version of Stokes and Xu (2020) to appear in Statistica Sinica. Chapter 5 borrows ideas from Stokes et al. (2020) published in Chemometrics and Intelligent Laboratory Sciences.

VITA

2016	B.S. Statistics, B.S. Mathematics, The University of Georgia
2017–2018	Statistical Analyst, Center for Inflammatory Bowel Diseases UCLA Department of Medicine
2017–2021	Statistical Consultant, UCLA Statistical Consulting Center
2018–2021	Teaching Assistant, UCLA Department of Statistics
2019	C.Phil. Statistics, UCLA

PUBLICATIONS AND PRESENTATIONS

Stokes, Z. and Xu, H. (2021). Designs for Order-of-Addition Screening Experiments. Preprint.

Zand, A., **Stokes, Z.**, van Deen, W., and Hommes, D. (2021). Learning Predictive Algorithms for Long-term Negative Health Outcomes in Inflammatory Bowel Diseases (IBD) Patients. Preprint.

Stokes, Z. and Xu, H. (2020). A Position-Based Approach for Design and Analysis of Order-of-Addition Experiments, *Statistica Sinica.* To Appear.

Zand, A., Kim, B., van Deen, W., **Stokes, Z.**, Platt, A. and Hommes, D. (2020). The Effects of Inflammatory Bowel Disease on Caregivers: Burden & Productivity Decrease, *BMC Health Services Research* **20**(1) 556.

Stokes, Z., Mandal, A. and Wong, W. K. (2020). Using Differential Evolution to Design Optimal Experiments, *Chemometrics and Intelligent Laboratory Systems* **199**, 103955.

Zand, A., Sharma, A., **Stokes, Z.**, Reynolds, C., Montilla, A., Sauk, J. and Hommes, D. (2020). An Exploration Into the Use of a Chatbot for Patients With Inflammatory Bowel Diseases: Retrospective Cohort Study, *Journal of Medical Internet Research* **22**(5), e15589.

Zand, A., Nguyen, A., **Stokes, Z.**, van Deen, W., Lightner, A., Platt, A., Jacobs, R., Reardon, S., Kane, E., Sack, J. and Hommes, D. (2019). Patient Experiences and Outcomes of a Telehealth Clinical Care Pathway for Postoperative Inflammatory Bowel Disease Patients," *Telemedicine and Telecare* **26**(7), 889-897.

Zand, A., Nguyen, A., **Stokes, Z.**, Reynolds, C., Dimitrova, M., Khong, H., Khandadash, A., Dvorsky, M., van Deen, W., Sauk, J., Esrailian, E., and Hommes, D. (2019). The Development of a Screening tool to Identify and Classify Non-adherence in Inflammatory Bowel Disease, *Chron's & Colitis 360* **1**(3), otz035.

Stokes, Z. and Xu, H. *Design and Analysis of Order-of-Addition Experiments with Application to Drug Sequencing.* DAE 2019, University of Tennessee, October 18-21, 2019.

Stokes, Z. *Advanced Statistical Models with Applications to Healthcare Research.* Invited speaker, Ministry of Public Health (MINSAP), Havana, Cuba, July 2-6, 2018.

Stokes, Z., Mandal, A., and Wong, W. K. *Differential Evolution for Optimal Design Creation.* DAE 2017, UCLA, October 12-14, 2017.

CHAPTER 1

Introduction

In 1935, R.A. Fisher performed the famous "Lady Tasting Tea" experiment in which four cups of tea were made by first introducing milk and then adding tea, while four others were made with tea first, milk second. The goal of this experiment was to test the lady's ability to label the mixture present in each of the eight cups when tasting them in a random order (Fisher, 1935). It is rumored that the lady was able to correctly identify all eight cups of tea, leading to the conclusion that it is highly unlikely that she was blindly guessing but rather has some ability to determine the order.

As interesting as this result is, the experiment is regarded as a historic moment in the field of statistics because of its place as one of the supporting pillars of randomization analysis of experiments (Basu, 1980). However, this trial also represents the first modern statistical attempt to understand the relationship between the sequential order of a set of components (milk and tea) and a measured outcome (correct detection of recipe).

In the decades since the lady proved her tea-tasting ability, the prevalence and complexity of similar experiments has grown tremendously, reaching a diverse set of disciplines and finding increased everyday importance on critical applied problems in the realm of data science. Order-of-addition problems have been found when studying physical and simulated phenomenon in many areas, such as chemistry and biology (Olsen et al., 1994; Ryberg, 2008), food science (Jourdain et al., 2009), political science (Grant, 2017), operations research (Garey et al., 1976), and mechanics and engineering (Voutchkov et al., 2005). As one specific example, we encounter order-of-addition experiments in both past and present drug combination projects. Combination chemotherapy has become commonly used in cancer treatment, viral infection eradication and super bacteria inhibition (Ding et al., 2013; Jaynes et al., 2013; Ding et al., 2015; Silva et al., 2016; Xiao et al., 2019). A major limitation in the current techniques for drug combination experimentation is that drugs are simultaneously added and drug sequence is not considered. However, drug sequence often plays a major role in deciding endpoint efficacy, since the early addition of certain drugs could prepare the biological system to better accept or defend the later drugs. Pre-clinical and clinical studies indicate that drug sequence is of great importance to improve the effect of the treatment (MacBeath and Yaffe, 2012; Wang et al., 2020). These and other critical applications are the driving force behind this dissertation.

1.1 Motivation and Objectives

Given the great impact that order-of-addition problems have across a diverse set of research areas and industries, and its foundation in classical statistics, it is astounding that robust statistical methodology for the design or analysis of order-of-addition experiments has only recently been developed. The key problem faced by practitioners looking to solve order-of-addition problems is the combinatorial explosion that takes place as the size of the component pool grows. Many of the recently developed models and designs, while providing a solid foundation for the study of this problem, are lacking in flexibility and as such perform poorly on large problems. Likewise, the existing methods fail to tackle several important cases that are practically relevant to many applied problems. Our aim is to fill a sizable portion of this gap while also highlighting the flexibility of our approaches and the broad implications of this research for many relevant problems in sequential drug administration and job scheduling.

1.2 Outline of the Dissertation

The outline of this dissertation is as follows. In Chapter 2 we introduce many relevant ideas to the modern study of order-of-addition experiments. Chapter 3 then presents several key developments in the design and analysis of order-of-addition experiments, including a family of flexible models and a robust design construction. Building on these new ideas, Chapter 4 studies the more nuanced problem of order-of-addition screening designs, in which the number of available positions cannot accommodate the number of components. As such, the endpoint analysis must isolate the most important components. We construct screening designs to aid in this process. In order to cover other significant problems in the order-of-addition family, we demonstrate in Chapter 5 the efficacy of nature-inspired metaheuristic algorithms to locate optimal designs, including a thorough comparison of popular methods and their modern variants. Chapter 6 summarizes our results and outlines other significant research areas that are lacking in the order-of-addition literature and could be explored using this dissertation as a foundation. Throughout this work we tie our results into many relevant applied problems that can benefit from the developed methods.

1.3 Individual Contributions

The major individual contributions to the field of order-of-addition experiments are described in Chapters 3 and 4. These contributions include new models and corresponding optimal designs for analyzing data from general order-of-addition experiments, along with the development of optimal designs for the more specialized component screening problem. Chapter 5 presents a novel study of the use of popular metaheuristic algorithms to solve pressing order-of-addition design problems in the areas of optimality and space-filling properties. These new methods are applied in the context of sequential drug administration and job scheduling.

CHAPTER 2

Current State of Order-of-Addition Research

The current state of order-of-addition research is characterized by a search for parsimonious models and efficient designs. There are two major classes of models, which differ according to the key assumption that is made about how the component positions interact with the response. This chapter first introduces pivotal concepts in the design of experiments that are used throughout our work and the notation used to frame order-of-addition problems. Next, the existing model classes and the corresponding designs that have been constructed are described. Finally, brief consideration is given to the many statistical and computational research areas that study problems closely related to the analysis of order-of-addition experiments.

2.1 Overview of Relevant Design Methodology

Before describing the state-of-the-art research of order-of-addition problems, we first briefly review the necessary background for constructing model-based designs and the applicable criteria that will be used throughout this work. Wu and Hamada (2009) provides further detail on many aspects of the design and analysis of controlled experiments.

2.1.1 Design Framework

We consider a design with n runs, where n is usually set by the experiment's budget and other practical considerations. For p experimental variables, each run is a point in p-dimensional space $\boldsymbol{x}_i^{\mathrm{T}} = (\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, \ldots, \boldsymbol{x}_{ip})$ that details the treatment to be administered and yields the corresponding scalar response y_i. We can combine the n runs into a design matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)^{\mathrm{T}}$. Depending on the nature of the experimental variables and assumptions made about the underlying data-generating process, there are many criteria that can be used to compare the quality of competing designs.

2.1.2 Factorial Designs

When each experimental variable is a discretized factor with a fixed number of levels, one criterion that we can use to assess the goodness of a design is the generalized wordlength pattern (GWLP). The GWLP $(W_1, \ldots, W_m)$ measures the aliasing of factorial effects, where $W_i \geq 0$ measures the overall aliasing of i-factor interactions on the general mean under the standard ANOVA model. An important property of the GWLP is that it characterizes the orthogonality or strength of a design. Xu and Wu (2001) showed that a design is an orthogonal array of strength t if and only if $W_1 = \cdots = W_t = 0$. Applying this result, we have $W_1 = 0$ if and only if the design is level balanced, that is, each level appears the same number of times in each column. Among level balanced designs, designs with small W_2 are preferred. The generalized minimum aberration criterion (Xu and Wu, 2001) favors designs which sequentially minimize $W_1, W_2, \ldots$. Generalized minimum aberration designs are model robust in the sense that they minimize contamination of higher-order effects on the estimation of lower-order effects. There are many variations and special cases of minimum aberration criteria. For examples, see Fries and Hunter (1980); Tang and Deng (1999); Xu (2003); Xu et al. (2009).

2.1.3 Optimal Approximate Designs

Instead of a factorial approach, we can also construct designs given a statistical model and an objective. Specifically, the assumed statistical model takes the form

$$y_i = f(\boldsymbol{x}_i, \boldsymbol{\beta}) + \epsilon_i, \quad i = 1, \ldots, n, \quad \boldsymbol{x} \in \mathcal{X}, \tag{2.1.1}$$

where y_i is the response at the vector of explanatory variables $\boldsymbol{x}_i$. The errors ϵ_i are identically and independently distributed with zero mean and some constant variance. $f(\boldsymbol{x}_i, \boldsymbol{\beta})$ is a known continuous function assumed to capture the relationship through the vector of unknown model parameters $\boldsymbol{\beta}$. The p-dimensional region available for experimentation is $\mathcal{X}$. If the model $f(\boldsymbol{x}, \boldsymbol{\beta})$ can be written as $\boldsymbol{f}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\beta}$, we refer to it as linear and the model matrix is given by $(\boldsymbol{f}(\boldsymbol{x}_1), \boldsymbol{f}(\boldsymbol{x}_2), \ldots, \boldsymbol{f}(\boldsymbol{x}_n))^{\mathrm{T}}$. Otherwise we refer to it as non-linear.

Given the study objective and the fixed total number of observations n available for the study, our goal is to optimally select a set of $|\boldsymbol{\beta}| \leq k \leq n$ support points $\boldsymbol{x}_i$'s, $i = 1, 2, \ldots, k$, from $\mathcal{X}$ to observe responses. We require the design space to be a compact set so that the optimum exists. Each support point is associated with a weight w_i such that $\sum_{i=1}^{k} w_i = 1$. If there are n distinct support points, each with weight $1/n$, we call the collection of points an exact design. In general, finding theoretically optimal exact designs for a fixed n is difficult.

We can instead look for optimal approximate designs, those in which the number of support points is less than n and the weights are often unequal. For the given model and criterion, the optimal approximate design can be implemented by rounding each nw_i to the nearest integer to ensure the resulting replicate counts sum to n. More background on approximate designs can be found in Kiefer (1959) and Silvey (1980). We denote a k-point approximate design with weight w_i at $x_i, i = 1, \ldots, k$ by $\boldsymbol{\xi}$. If $\boldsymbol{\nabla} f(\boldsymbol{x}, \boldsymbol{\beta})$ is the partial derivative of the mean function with respect to the model parameters $\boldsymbol{\beta}$, then the normalized information matrix for the design $\boldsymbol{\xi}$ is given by

$$\boldsymbol{M}(\boldsymbol{\xi}, \boldsymbol{\beta}) = \sum_{i=1}^{k} w_i \Big(\boldsymbol{\nabla} f(\boldsymbol{x}_i, \boldsymbol{\beta})\Big)\Big(\boldsymbol{\nabla} f(\boldsymbol{x}_i, \boldsymbol{\beta})\Big)^{\mathrm{T}}. \tag{2.1.2}$$

The covariance matrix of the maximum likelihood estimates for the parameters $\boldsymbol{\beta}$ is inversely proportional to $\boldsymbol{M}(\boldsymbol{\xi}, \boldsymbol{\beta})$, so making the information matrix large in some sense is desirable. There are two widely-used criteria to meet this goal: D-optimality and A-optimality. A D-optimal design maximizes $|\boldsymbol{M}(\boldsymbol{\xi})|$ while an A-optimal design minimizes $tr(\boldsymbol{M}^{-1}(\boldsymbol{\xi}))$. The D-optimality criterion seeks to minimize the volume of the confidence ellipsoid around the parameter estimates, and the A-optimality criterion minimizes the sum of the variances of the parameter estimates. When the model is nonlinear, the information matrix is a function of the unknown parameters and locally optimal designs or Bayesian optimal designs can be found (Atkinson, 1996).

Given a reference design and one of these two criteria, we can compare the quality of any proposed design to this reference one. For convenience, we define the D- and A-efficiency of $\boldsymbol{\xi}_1$ under model (2.1.1) relative to $\boldsymbol{\xi}_2$ respectively as

$$D(\boldsymbol{\xi}_1) = \{|\boldsymbol{M}(\boldsymbol{\xi}_1)|/|\boldsymbol{M}(\boldsymbol{\xi}_2)|\}^{1/p}, \ A(\boldsymbol{\xi}_1) = \{tr(\boldsymbol{M}^{-1}(\boldsymbol{\xi}_2))/tr(\boldsymbol{M}^{-1}(\boldsymbol{\xi}_1))\}, \tag{2.1.3}$$

where p is the number of columns in the model matrix. If the above ratios are 0.5, then the design $\boldsymbol{\xi}_1$ requires twice as many observations to perform as well as $\boldsymbol{\xi}_2$ with respect to the given criterion. If the reference design $\boldsymbol{\xi}_2$ is optimal under the chosen criterion then we sometimes refer to these values as the D- and A-efficiency of $\boldsymbol{\xi}_1$ without explicitly referring to the reference design.

We can use the general equivalence theorem for checking the optimality of a design over all designs defined on $\mathcal{X}$. This result is based on directional derivative considerations of the convex design criterion (Kiefer, 1961; Fedorov, 1972). Each convex optimality criterion has a unique equivalence theorem. For example, under a linear model over compact space $\mathcal{X}$ with the two criteria discussed above, a design $\boldsymbol{\xi}^*$ is optimal if and only if

$$D: \quad \boldsymbol{f}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{M}(\boldsymbol{\xi}^*)^{-1}\boldsymbol{f}(\boldsymbol{x}) - p \leq 0 \ \ \forall \boldsymbol{x} \in \mathcal{X}, \tag{2.1.4}$$

$$A: \quad \boldsymbol{f}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{M}(\boldsymbol{\xi}^*)^{-2}\boldsymbol{f}(\boldsymbol{x}) - tr(\boldsymbol{M}(\boldsymbol{\xi}^*)^{-1}) \leq 0 \ \ \forall \boldsymbol{x} \in \mathcal{X}, \tag{2.1.5}$$

with equality obtained at the design points $\boldsymbol{x} \in \boldsymbol{\xi}^*$. We refer to each of (2.1.4) and (2.1.5) as checking conditions and call the function on the left hand side the design's sensitivity function.

2.1.4 Space-Filling Designs

Sometimes we are interested in understanding the relationship between the inputs to a deterministic computer simulation and its output. While the original simulation may take a significant amount of time to generate each output, we can train a surrogate model that acts as a cheaper approximation of the full simulation (Fang et al., 2006; Santner et al., 2013). Since there is no noise in the observations, we can focus on designs that minimize the bias. Johnson et al. (1990) found that designs which in some way maximally fill the design space are effective for this purpose. The authors propose the maximin and minimax distance criteria for finding such space-filling designs. A design S^* with fixed size n is a maximin or minimax optimal design respectively if

$$\min_{s,s' \in S^*} d(s,s') = \max_{S} \min_{s,s' \in S} d(s,s'), \tag{2.1.6}$$

$$\max_{t \in T} d(t,S^*) = \min_{S} \max_{t \in T} d(t,S), \tag{2.1.7}$$

where T is a collection of candidate points in the design space, S is any set of points with cardinality n, d is a distance function and $d(t,S) = \min_{s \in S} d(s,t)$. These designs have been shown to work especially well for cases in which our data is modeled as a Gaussian Process, a spatial process that generates multivariate normal observations where the covariance is a function of the distances between the observations. For a thorough introduction to space-filling designs with Gaussian Processes and the computer experiments that use them, see Gramacy (2021).

2.2 Order-of-Addition Framework

With the primary criteria that we will use to benchmark the proposed designs established, we can introduce the major ideas behind order-of-addition experiments. In many cases, an experimental process consists of performing a set of steps in a fixed order in which the execution of each step depends on a set of experimental variables. At the conclusion of each run of the process, the response or responses of interest are recorded. However, in some cases the variables of interest are the steps of the process themselves. For example, Voutchkov et al. (2005) studied the relationship between welding patterns of a crucial aircraft engine mount and the resulting structural properties. While there are many variables that could be studied as part of the welding process (time, temperature, etc.), the experimenters were interested in understanding the impact of the order of 6 critical weld paths.

In this example of an order-of-addition experiment, the goal is to predict the structural integrity of the finished product from the weld path sequence; however, order-of-addition experiments may have other goals. For example, we may wish to make inferences about the effect of the order of candidate names on a ballot on the outcome of an election (Grant, 2017) or to optimize the completion rate of a sequence of onboarding screens in a mobile application to encourage a positive user experience. In order to design and analyze these and other order-of-addition experiments we need a standard framework.

We call each material to be added or step to be performed in the experiment a component. Each set of ordered components is called a sequence or ordering. If the experiment involves m components, denoted by $0, 1, \ldots, m-1$, and the budget allows for n runs, then we can collect a set of n orderings in an $n \times m$ component matrix $\boldsymbol{A} = (a_{ij})$, where each row is a permutation of the components $0, 1, \ldots, m-1$. There are a total of $m!$ possible permutations. Let $\mathbf{F}_m$ be the component matrix of the full design with $m!$ distinct rows and m columns, where each row is a permutation of m components. To illustrate this notation, Table 2.1 gives the component matrix of the full design for four components $\mathbf{F}_4$.

Table 2.1: Component matrix for the full design $\mathbf{F}_4$.

Run	a_1	a_2	a_3	a_4	Run	a_1	a_2	a_3	a_4
1	0	1	2	3	13	2	0	1	3
2	0	1	3	2	14	2	0	3	1
3	0	2	1	3	15	2	1	0	3
4	0	2	3	1	16	2	1	3	0
5	0	3	1	2	17	2	3	0	1
6	0	3	2	1	18	2	3	1	0
7	1	0	2	3	19	3	0	1	2
8	1	0	3	2	20	3	0	2	1
9	1	2	0	3	21	3	1	0	2
10	1	2	3	0	22	3	1	2	0
11	1	3	0	2	23	3	2	0	1
12	1	3	2	0	24	3	2	1	0

Performing all possible permutations (*i.e.* following the full design $\mathbf{F}_m$) quickly becomes unfeasible even for experiments with five or more components. For example, in the welding problem there are six possible welds, each following one of two directions, leading to a full design with over 40,000 runs. Furthermore, each individual run of this experiment requires 32 hours to perform. To save time and cost, it is necessary to instead choose a subset of the runs to perform. Natural questions then arises of which subset to choose and how to model the response.

2.3 Relative Position Models

There have only been a few studies to develop models for studying the ordering effect. At the heart of the models that have been proposed is a key assumption: is the ordering effect driven by the relative position or the absolute position of each component in the sequence? In other words, can we influence the response by simply moving a single component to be later in the sequence, or does understanding this relationship require that we consider which components were displaced when the component was moved? The choice of which of these assumptions to make about the underlying data generating process plays a significant role in how we model the problem. However, it is often the case that our model assumption is incorrect, so it is also important to consider that the other assumption may be the correct one, or that the true relationship is a hybrid of both approaches. For the remainder of this section we consider a family of models that make the assumption that the relative position of components drives the relationship.

2.3.1 Pairwise Ordering Models

Van Nostrand (1995) and Voelkel (2019) studied order-of-addition experiments by creating a set of psuedo-factors $\{I_{ij},\ 0 \le i < j \le m-1\}$ such that each corresponds to the pairwise ordering of the components. For example, in the case of $m = 4$ components, the six pairwise ordering factors are $I_{01}, I_{02}, I_{03}, I_{12}, I_{13}, I_{23}$. Each factor I_{ij} has two levels, 1 and -1, indicating whether or not, respectively, component i is added before component j. Furthermore, they considered the following pairwise ordering (PWO) model

$$y = \beta_0 + \sum_{i<j} \beta_{ij} I_{ij} + \varepsilon, \tag{2.3.8}$$

where β_{ij} represents the relative position effect between components i and j and $\varepsilon \sim N(0, \sigma^2)$ is a random error.

Voelkel (2019) also constructed a class of designs for this PWO model called Order-of-Addition Orthogonal Arrays (OofA-OAs). We use $\boldsymbol{OA}_{n,m}$ to refer to an OofA-OA in n runs and m components. These designs have the same information matrix as the full design $\mathbf{F}_m$ when n is a multiple of 12 and $m = 4, 5$ or when n is a multiple of 24 and $m = 6$. Each OofA-OA in m components with n runs has the property that each pair of psuedo-factors

Table 2.2: Component matrix of $\boldsymbol{OA}_{12,4}$ developed by Voelkel (2019) and its set of six pseudo-factors.

Run	a_1	a_2	a_3	a_4	I_{01}	I_{02}	I_{03}	I_{12}	I_{13}	I_{23}
1	0	1	3	2	1	1	1	1	1	-1
2	0	2	1	3	1	1	1	-1	1	1
3	0	3	1	2	1	1	1	1	-1	-1
4	1	0	2	3	-1	1	1	1	1	1
5	1	2	3	0	-1	-1	-1	1	1	1
6	1	3	2	0	-1	-1	-1	1	1	-1
7	2	0	3	1	1	-1	1	-1	-1	1
8	2	1	0	3	-1	-1	1	-1	1	1
9	2	3	0	1	1	-1	-1	-1	-1	1
10	3	0	2	1	1	1	-1	-1	-1	-1
11	3	1	0	2	-1	1	-1	1	-1	-1
12	3	2	1	0	-1	-1	-1	-1	-1	-1

(I_{ij}, I_{kl}) meets the following conditions:

1. If $i \neq k$, $i \neq l$, $j \neq k$, and $j \neq l$, the factors are orthogonal.
2. If $i = k$ or $j = l$, the inner product of the factors is $n/3$.
3. If $i = l$ or $j = k$, the inner product of the factors is $-n/3$.

An example of the component matrix and corresponding set of pseudo-factors for a particular design, $\boldsymbol{OA}_{12,4}$, are given in Table 2.2. This table demonstrates how we can easily convert the component matrix $\boldsymbol{A}$ into the corresponding model matrix for the PWO model (2.3.8).

Peng et al. (2019) showed that the full design $\mathbf{F}_m$ is optimal for the PWO model under any concave and signed permutation invariant criterion. The authors also constructed a class of fractional designs that are optimal under these same conditions. However, their designs often have an excessive number of runs and may be less useful in practice. Zhao et al. (2020a) constructed minimally-supported designs for the PWO model containing only one point per parameter, and Chen et al. (2020) used small OofA-OAs as blocks to construct designs with a large number of components ($7 < m < 16$) that are efficient under the PWO model. Recently, Schoen and Mee (2021) proposed an algorithmic approach to enumerating all OofA-OAs for a given n and m. A summary of the standard PWO model is provided in Lin and Peng (2019).

2.3.2 Triplets and Other Expansions

Several extensions of the PWO model have been considered to improve its performance. For example, in addition to the main effects present in (2.3.8), it is common in practice to find that a few two-factor interactions are also significant. For this reason, Mee (2020) considered expanded pairwise models that include interactions of the pairwise factors I_{jk}. The model that includes only factor interactions that involve exactly three components is dubbed the triplets model and is given by

$$y = \beta_0 + \sum_{j<k} \beta_{jk} I_{jk} + \sum_{j=1}^{m-2} \sum_{k=j+1}^{m-1} \sum_{l=k+1}^{m} [\beta_{jk\star jl} I_{jk} I_{jl} + \beta_{jk\star kl} I_{jk} I_{kl}] + \varepsilon, \tag{2.3.9}$$

where $\beta_{jk\star jl}$ captures the interaction effect between the relative position of components j and k and components j and l. This model contains many more parameters than the standard PWO model, so it requires a much larger run size to estimate; however, this also gives it additional flexibility that may produce a better fit.

In addition to the prevalence of higher-order terms it may be the case that the relative position effect between a pair of components grows weaker the further apart the components are in the sequence. To account for this effect, Peng et al. (2019) considered tapered PWO models with various levels of tapering. They also found that designs constructed via standard optimal design algorithms are robust to the magnitude of the tapering.

2.4 Absolute Position Models

Instead of assuming that the ordering effect is driven by the relative position of components we can instead assume that the relationship depends only on each individual component's absolute position. This assumption is generally valid when there is a temporal relationship between a component and its effect on the response. For example, in sequential drug studies, administering an individual drug earlier in the sequence gives the drug more time to take effect, regardless of which drugs come before or after. In this case the absolute position of the drug drives its relationship with the response. As with the relative position assumption, it could be that this assumption is incorrect, so it is important to base the choice of model on the application details.

2.4.1 Component-Position Models

The Component-Position (CP) model was established by Yang et al. (2021) as an alternative to the PWO model. For this model the authors incorporate the absolute position effect assumption by using a one-way layout. They constructed an indicator $z_{kj}^{(i)}$ for each component-position pair (k, j), such that $z_{kj}^{(i)}$ is 1 if $a_{ij} = k$, and 0 otherwise. Because exactly one component is used at each position, we have $\sum_{k=0}^{m-1} z_{kj}^{(i)} = 1$ for any i and j. Thus, $m-1$ contrasts are needed to represent the effects of m components for each position. Because each run is a permutation of m distinct components, we also have $\sum_{j=1}^{m} z_{kj}^{(i)} = 1$ for any i and k. As a result, we can only include $m-1$ positions in the model. With these constraints the CP model is given by

$$y = \gamma_0 + \sum_{k=1}^{m-1} \sum_{j=1}^{m-1} z_{kj}\gamma_{kj} + \varepsilon, \tag{2.4.10}$$

where y is the response, γ_0 is the intercept, z_{kj} is an indicator for the component-position pair (k, j), as described above, γ_{kj} is the parameter representing the effect of component k being added at the j^{th} position, and ε is an independent normal random error. The authors showed that the full design $\mathbf{F}_m$ is D-optimal under the CP model.

Apart from the CP model, few other absolute position effect models have been proposed. Two examples are the response surface models proposed by Piepho and Williams (2021) and the Gaussian Process-based models by Xiao and Xu (2021) that attempt to estimate the distances between positions instead of treating them as equidistant. Otherwise, the focus has been on developing designs for the CP model.

2.4.2 Component Orthogonal Arrays

Yang et al. (2021) proposed a class of designs for the CP model (2.4.10) built from the ideas of orthogonal arrays, the Component Orthogonal Array (COA). An $n \times m$ matrix with entries from $\{0, 1, \ldots, m-1\}$ is a COA, denoted by COA(n, m), if each row is a permutation of $\{0, 1, \ldots, m-1\}$ and, for any subarray of two columns, each level combination (i, j), with $i \neq j$ and $i, j = 0, 1, \ldots, m-1$, appears equally often.

Fom this definition, every level combination (i, j), with $i \neq j$ and $i, j = 0, 1, \ldots, m-1$, must appear equally often in every two-column sub-array of a COA(n, m). Thus, to be a COA, a design must have $n = \lambda m(m-1)$ runs, where λ is an integer. Thus, COAs are orthogonal arrays of type I, as defined by Rao (1961). An example of a COA(12,4) from Yang et al. (2021) with the corresponding CP model matrix is given in Table 2.3

Yang et al. (2021) proposed an algorithm that for any m such that m is prime or a prime power, produces a COA$(m(m-1),\, m)$. Zhao et al. (2020b) created a new criteria and used it to find COAs with $m = 6$, and Huang (2021) presented a new recursive construction that can generate COAs for any m.

2.5 Related Literature

In addition to the published literature on order-of-addition experiments, there are several other research areas that tackle order-of-addition problems, such as studying the relationship between a response and a set of permutations, or locating the permutation that minimizes

Table 2.3: COA(12,4) presented by Yang et al. (2021) and the corresponding CP model matrix.

Run	a_1	a_2	a_3	a_4	z_{11}	z_{21}	z_{31}	z_{12}	z_{22}	z_{32}	z_{13}	z_{23}	z_{33}
1	0	1	2	3	0	0	0	1	0	0	0	1	0
2	0	2	3	1	0	0	0	0	1	0	0	0	1
3	0	3	1	2	0	0	0	0	0	1	1	0	0
4	1	0	3	2	1	0	0	0	0	0	0	0	1
5	1	2	0	3	1	0	0	0	1	0	0	0	0
6	1	3	2	0	1	0	0	0	0	1	0	1	0
7	2	0	1	3	0	1	0	0	0	0	1	0	0
8	2	1	3	0	0	1	0	1	0	0	0	0	1
9	2	3	0	1	0	1	0	0	0	1	0	0	0
10	3	0	2	1	0	0	1	0	0	0	0	1	0
11	3	1	0	2	0	0	1	1	0	0	0	0	0
12	3	2	1	0	0	0	1	0	1	0	1	0	0

the response. These related techniques come from research in designs for other controlled experiments, active learning on a pool of permutations, and general optimization on discrete spaces. We briefly review each of these areas.

2.5.1 Alternative Experiments

There are several research areas in experimental design that have a similar premise to order-of-addition experiments but vary in their applicability to the problems proposed here. The major discrepancies between these methods and the design of order-of-addition experiments involve the underlying assumptions placed on the component relationships that allow for unique construction methods and models.

One such related area is that of crossover experiments. Designs for crossover experiments are used to study multiple treatments that are applied to units over several periods (Jones and Kenward, 2015). While this appears similar to our objectives, there are some major differences. In order-of-addition experiments, a single response is recorded after all components have been added to the treatment. In contrast, a single crossover trial has multiple observed responses, as each component is considered to be a treatment on its own. Additionally, the primary objective differs between the two classes of experiments. For our case, interest is often in finding the addition order that optimizes an endpoint response. In crossover experiments the goal is to instead compare the effects of individual components. Thirdly, the underlying assumptions placed on the component effects differ between the two situations. In an order-of-addition experiment, the effects are believed to be dependent on the order of component administration. In a crossover trial, each component has a fixed effect, which may carry over to the next period but does not depend on the order of administration. Due to these differences, the results of design and analysis for crossover trials are not applicable to order-of-addition experiments. For an introduction to the models and designs for crossover experiments see Bose and Dey (2015).

Order-of-addition experiments can also be viewed as a constrained version of mixture experiments (Box and Draper, 2007; Cornell, 2011). Mixture experiments are those in which each run is a set of proportions detailing how much of a certain ingredient to include in the mixture. The response is modeled as a function of these proportions. This means that for m components, design points are selected from the continuous $(m-1)$-simplex. The most common models used for these experiments are Scheffé polynomials (Scheffé, 1958). However, order-of-addition experiments are more complicated in that each run must be a permutation of m components. If we divide each row of the component matrix $\boldsymbol{A}$ by $\sum_{j=0}^{m-1} a_{ij}$, then we get a component mixture matrix $\boldsymbol{C} = (c_{ij})$ that covers only a small, discrete subspace of the simplex. The optimal design points for modeling mixture experiments with Scheffé polynomials often fall along the vertices and boundaries of the full simplex, few of which are included in this subspace. As a result, optimal designs for these models cannot be used explicitly for order-of-addition experiments. However, the models and methods of the mixture experiment literature may be useful in cases in which the relationship between both the order and concentration of components with the response is being investigated. For example, Piepho and Williams (2021) recently proposed regression models for order-of-addition experiments based on the principles of mixture experiments.

2.5.2 Pool-based Active Learning

As an alternative to a designed experiment, we can also consider sequential experimentation with active learning to solve order-of-addition problems. These techniques are useful when we have a large pool of candidates (in our case a set of permutations), and the goal is to optimize a response, yet obtaining the response for a specific candidate is quite expensive. For these cases we instead collect a small initial sample of responses and then iteratively sample at new locations to gain more information about the problem.

There are two distinct bodies of literature that tackle the problem of active learning over the space of the permutations. The first focuses on the problem of recovering the true, optimal ordering of a set of components by querying the pairwise preferences. Collecting just a small, yet targeted set of these pairwise orderings can lead to accurate recovery of the best sequence. There are several algorithms for performing this type of active learning, many of which rely on some type of clustering or statistical learning paradigm (Mitliagkas et al., 2011; Ailon, 2012). These methods are especially relevant in practice for real-time matching of user preferences to search result or product recommendation orderings. However, since we typically consider the entire experimental process holistically in order-of-addition experiments, we rarely have access to the response for individual pairs of components.

The second body of active learning literature that deals with the space of permutations involves optimizing a response given the entire permutation sequence. This is usually done either for a given set of features, such as in the case of matching individual users to items they will be interested in, or generally for black-box optimization problems. In the first case, several state-of-the-art Reinforcement Learning solutions have been proposed, whereas in the second case it is more common to use Bayesian Optimization (Baptista and Poloczek, 2018; Emami and Ranka, 2018; Mena et al., 2018). While the first problem generally requires more data than we have access to prior to running the experiment, the second problem has been studied in the context of order-of-addition experiments in Xiao et al. (2020). In this work the authors present an active learning framework in conjunction with order-of-addition models and initial designs to optimize the response when the experimenter is interested in both the ordering effect and additional covariates associated with each component. The development of active learning algorithms to solve order-of-addition problems is a growing area of research. In Chapter 4 we explore the ability of our proposed designs to improve the speed of convergence of such algorithms.

2.5.3 Optimization over Permutations

In addition to active learning optimization approaches, there have been many solvers proposed for optimization over the space of permutations generically. Buchhei and Jünger (2005) reviews many of these approaches, but this is still only a small sample of the many methods that have been studied for discrete optimization on the finite symmetric group. In addition to general optimization on the space of permutations, there has been extensive work done in the creation of solvers for specific problems. Two primary examples of this are the traveling salesman problem (TSP) and permutation flow shop scheduling (PFS). For examples of some of the solvers that have been proposed for these two problems see Rego et al. (2011) and Kumar and Jadon (2014), respectively. While these solvers are adept at quickly locating the optimal value of the response, they often provide no insight into the dynamics of the relationship between the ordering and the response and sometimes require access to data that we often do not have available. For this reason, they provide interesting insight into approaches we may incorporate into our models and designs, but are not directly applicable.

2.6 Chapter Summary

In this chapter we have looked at the foundations of experimental design including useful criteria that we will apply throughout the remainder of this work. We have also detailed the primary developments and key motivations behind order-of-addition experiments in terms of existing models and designs. Lastly, we have covered alternative approaches for tackling the ordering effect problem including other design frameworks, active learning paradigms, and brute force search algorithms. With this foundation, we now present several methodologies that fill crucial gaps in the current statistical literature on the design and analysis of order-of-addition experiments.

CHAPTER 3

A Position-Based Approach for the Design and Analysis of Order-of-Addition Experiments

Despite the progress that has been made through the developments presented in Chapter 2, the statistical literature for designing and analyzing order-of-addition experiments remains quite limited. In the first major contribution of this work, we present a new, position-based framework for modeling order-of-addition problems. We then examine the limitations of the existing models in the context of real, sequential drug administration experiments and show how the proposed position methods are parsimonious, yet provide an interpretable, appropriate fit.

In addition to a new class of flexible models, we present a construction algorithm for producing optimal and near-optimal designs for both the proposed models and the existing CP and PWO models. We evaluate the performance and properties of the designs from our algorithm using the design criteria established in the previous chapter and compare them with those of existing designs. This includes a study that demonstrates that the proposed designs are highly efficient and robust to algorithm tuning and certain model misspecification.

Before elaborating on our new approach to modeling order-of-addition problems, we first introduce two real data sets that showcase both the prevalence of the ordering effect in critical real-life problems and the necessity for additional modeling techniques. Several recent works have focused on the problem of choosing an optimal sequence for drug administration described in Chapter 1. Figure 3.1 shows the component-position effects plots for four-drug (left) and five-drug (right) order-of-addition experiments from Yang et al. (2021) and Mee

(2020). These experiments considered four and five chemotherapeutics, respectively, for treating lymphoma that have received FDA approval for clinical testing (Wang et al., 2020). Each drug was tested at a fixed dosage estimated from a preliminary dose-response study. For each sequence, a drug was administered every four hours in the four-drug study, and every three hours in the five-drug study.

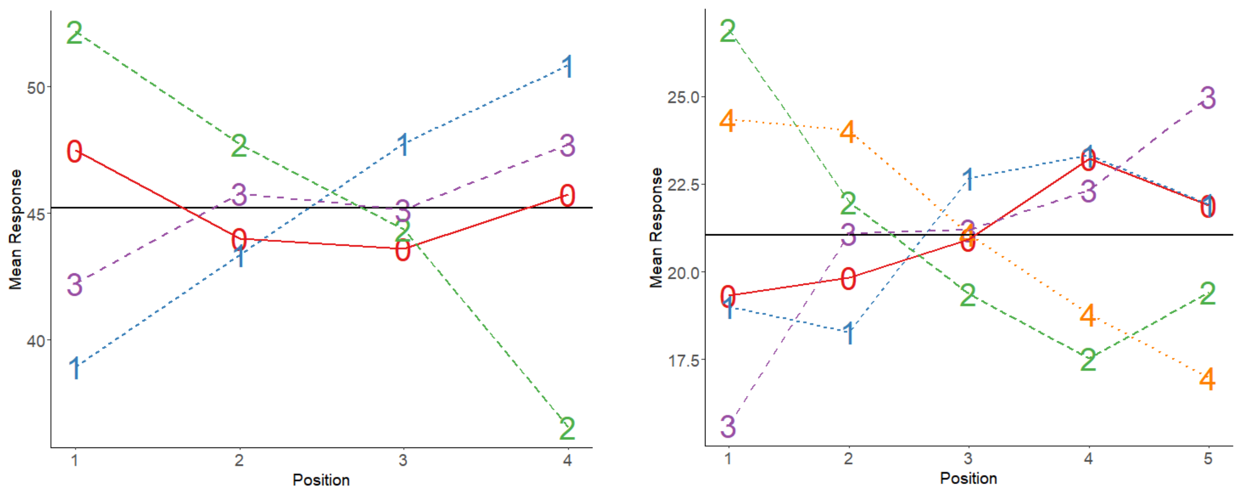


Figure 3.1: Component-position effects plots for four-drug (left) and five-drug (right) order-of-addition experiments.

In each plot, the horizontal axis denotes the position at which a drug is added, and the vertical axis denotes the mean response, in this case, a measure of cancer cell inhibition 24 hours after the first drug was administered. Each point denotes the mean response of all runs in which the labeled drug is administered in the fixed position. For each drug, the m dots corresponding to m different positions are connected to visualize the trend as that drug is shifted to a later position in the sequence. The solid horizontal line, used as a reference, represents the average response of all observations.

Both plots show that the effect of a drug on tumor inhibition depends on its position. The four-drug plot suggests that the component effects have a nearly linear relationship with the position. In this case, the authors found that both the PWO model (2.3.8) and the CP model (2.4.10) fit the data well, with predictive R^2 of 0.67 and 0.54, respectively. On the other hand, the five-drug plot suggests that the relationship between the component effects and the position is nonlinear. Neither model fits the data well, with predictive R^2 of 0.20 and 0.09, respectively. The triplets model described in Section 2.3 demonstrates better performance on this problem, but it involves many additional model parameters that necessitate a larger design. Our goal is thus to create new models and designs that can handle increasingly complex situations, such as the five-drug example, without requiring an excessive number of runs.

3.1 Flexible Position Models

We have so far presented the design matrix for order-of-addition experiments as an $n \times m$ matrix $\boldsymbol{A} = (a_{ij})$, where each row is a permutation of the components $0, 1, \ldots, m-1$. However, we now define a new $n \times m$ matrix $\boldsymbol{B} = (b_{ik})$ as follows: $b_{ik} = j$ if $a_{ij} = k-1$, for $k = 1, \ldots, m$. Note that a_{ij} is the component used at the jth position of the ith run, while b_{ik} is the position of component $k-1$ in the ith run. For example, the left side of row 14 in Table 3.1 indicates that the four components should appear in the order (2, 0, 3, 1), while the right-hand side equivalently states that positions (2, 4, 1, 3) should be assigned to components 0, 1, 2, and 3, respectively. Each row of $\boldsymbol{B}$ is a permutation of the m positions $1, \ldots, m$. To maintain the previous notation, we refer to $\boldsymbol{B}$ as the position matrix.

To compare our new models against the existing ones, we use the previously discussed four- and five-drug data from Yang et al. (2021) and Mee (2020), respectively. The data from these two experiments are given in Table 3.1 (matrices $\boldsymbol{A}$ and $\boldsymbol{B}$) and Table 3.2 (matrix $\boldsymbol{A}$), respectively. Recall that the existing methods are not sufficient for estimating the interaction effects between the drugs without sacrificing efficiency. Thus, we propose the following broader class of linear models based on the position matrix $\boldsymbol{B} = (b_{ik})$ that overcomes this

Table 3.1: Design and data for a four-drug order-of-addition experiment.

	Components				Positions				
Run	a_1	a_2	a_3	a_4	b_1	b_2	b_3	b_4	y
1	0	1	2	3	1	2	3	4	41.1
2^*	0	1	3	2	1	2	4	3	37.5
3^*	0	2	1	3	1	3	2	4	55.4
4	0	2	3	1	1	4	2	3	56.5
5	0	3	1	2	1	3	4	2	43.3
6^*	0	3	2	1	1	4	3	2	51.2
7^*	1	0	2	3	2	1	3	4	46.1
8	1	0	3	2	2	1	4	3	27.8
9	1	2	0	3	3	1	2	4	39.5
10^*	1	2	3	0	4	1	2	3	46.4
11^*	1	3	0	2	3	1	4	2	34.4
12	1	3	2	0	4	1	3	2	39.4
13	2	0	1	3	2	3	1	4	53.5
14^*	2	0	3	1	2	4	1	3	51.2
15^*	2	1	0	3	3	2	1	4	50.8
16	2	1	3	0	4	2	1	3	51.4
17	2	3	0	1	3	4	1	2	52.9
18^*	2	3	1	0	4	3	1	2	53.4
19^*	3	0	1	2	2	3	4	1	39.1
20	3	0	2	1	2	4	3	1	46.4
21	3	1	0	2	3	2	4	1	37.2
22^*	3	1	2	0	4	2	3	1	42.1
23^*	3	2	0	1	3	4	2	1	46.8
24	3	2	1	0	4	3	2	1	41.8

Note: The 12 ($*$) runs were used in Section 3.1.1 to fit the models and compare the quality of the out-of-sample predictions.

weakness:

$$y = \boldsymbol{f}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\beta} + \varepsilon, \tag{3.1.1}$$

where $\boldsymbol{x}$ is a row of the position matrix $\boldsymbol{B}$, $\boldsymbol{f}(\boldsymbol{x})$ is a vector of some basis functions, $\boldsymbol{\beta}$ is a vector of unknown coefficients, and $\varepsilon \sim N(0, \sigma^2)$. All error terms are independent. Using $\boldsymbol{B}$, we can represent the two existing models, PWO and CP, as special cases of this model. Specifically, the PWO model uses a set of basis functions that return the sign of $b_k - b_l$ for each pair of components $k-1$ and $l-1$ when $\boldsymbol{x} = (b_1, \ldots, b_m)$, where b_k gives the position of component $k-1$. The CP model similarly includes one indicator function for every component-position pair (k, j). However, these methods do not take full advantage of the benefit provided by this new position-based perspective.

Because positions have a natural order, we can study their effects using polynomial functions (*e.g.*, Wu and Hamada (2009)). Such a model was proposed by Anderson-Cook and Lu (2019), but no framework or details were given. We define the orthogonal polynomials of degree 1 and 2 over the set of positions as

$$p_1(x) = c_1\left(x - \frac{m+1}{2}\right) \text{ and } p_2(x) = c_2\left[\left(x - \frac{m+1}{2}\right)^2 - \left(\frac{m^2-1}{12}\right)\right],$$

respectively, where c_1 and c_2 are scalars that ensure that the length of each contrast vector is $\sqrt{m}$. For example, when $m = 4$, $c_1 = 2/\sqrt{5}$ and $c_2 = 2$, and $(p_1(x), p_2(x)) = (-1.5c_1, 1), (-0.5c_1, -1), (0.5c_1, -1)$, and $(1.5c_1, 1)$, for $x = 1$, 2, 3, and 4, respectively. When $m = 5$, $c_1 = \sqrt{1/2}$ and $c_2 = \sqrt{5/14}$, and $(p_1(x), p_2(x)) = (-2c_1, 2c_2), (-c_1, -c_2), (0, -2c_2)$, $(c_1, -c_2)$, and $(2c_1, 2c_2)$, for $x = 1$, 2, 3, 4, and 5, respectively.

The orthogonal polynomials have the following constraints:

$$(a)\ \sum_{x=1}^{m} p_j(x) = 0, \quad (b)\ \sum_{x=1}^{m} p_j^2(x) = m, \tag{3.1.2}$$

for $j = 1, 2$. These constraints complicate the modeling and the study of the design optimality for order-of-addition experiments, because each row of the position matrix $\boldsymbol{B}$ is a permutation of $\{1, \ldots, m\}$.

Table 3.2: Design and data for a five-drug order-of-addition experiment.

Run	Components					y	Run	Components					y
1	3	1	0	2	4	4.93	21	3	1	2	4	0	5.53
2	1	0	2	3	4	13.63	22	1	0	3	4	2	7.72
3	3	0	1	4	2	15.57	23	0	1	3	2	4	10.96
4	3	2	4	0	1	18.47	24	1	3	2	0	4	12.09
5	4	3	0	1	2	19.5	25	3	0	4	2	1	13.84
6	0	1	4	3	2	20.23	26	0	3	4	1	2	16.25
7	1	3	4	2	0	21.47	27	0	4	2	3	1	16.37
8	0	4	1	2	3	21.59	28	3	2	0	1	4	17.97
9	0	2	3	1	4	23.55	29	4	2	3	0	1	19.71
10	0	3	2	4	1	23.61	30	4	3	1	2	0	20.35
11	1	2	0	4	3	23.85	31	1	4	0	2	3	20.4
12	3	4	2	1	0	25.23	32	0	2	1	4	3	22.06
13	4	2	1	3	0	25.62	33	2	1	4	0	3	22.35
14	2	1	3	4	0	26.08	34	2	0	1	3	4	23.37
15	4	0	3	2	1	26.75	35	3	4	1	0	2	23.4
16	1	4	3	0	2	28.38	36	4	1	0	3	2	24.31
17	2	3	1	0	4	29.43	37	1	2	4	3	0	24.65
18	2	4	0	3	1	30.52	38	2	3	0	4	1	25.99
19	2	0	4	1	3	31.27	39	2	4	3	1	0	26.3
20	4	1	2	0	3	31.96	40	4	0	2	1	3	26.49

Using these polynomials, we consider three specific models:

$$\begin{aligned} y &= \beta_0 + \sum_{k=1}^{m-1} p_1(b_k)\beta_k + \varepsilon, & (3.1.3) \\ y &= \beta_0 + \sum_{k=1}^{m-1} p_1(b_k)\beta_k + \sum_{k=1}^{m-1} p_2(b_k)\beta_{kk} + \varepsilon, & (3.1.4) \\ y &= \beta_0 + \sum_{k=1}^{m-1} p_1(b_k)\beta_k + \sum_{k=1}^{m-2} p_2(b_k)\beta_{kk} \\ &+ \sum_{1\le k<l\le m-1} p_1(b_k)p_1(b_l)\beta_{kl} + \varepsilon, & (3.1.5) \end{aligned}$$

where y is the response, $b_1, \ldots, b_m$ are the positions of the m components, β_0 is the intercept, β_k, β_{kk}, and β_{kl} are unknown parameters, and $\varepsilon \sim N(0, \sigma^2)$ is a random error. We can interpret the main effect parameters as the expected change in the response after moving the specified component one position later in the sequence. Because each row of the position matrix is a permutation of $\{1, \ldots, m\}$ and the orthogonal polynomials obey the constraints in (3.1.2), we must remove one component effect from models (3.1.3) and (3.1.4) in order to make the models estimable. Furthermore, model (3.1.5) only includes β_{kk}, for $k = 1, 2, \ldots, m-2$, and does not contain any interaction terms involving component $m-1$. We can similarly craft more complicated models with higher-order terms, if needed. For convenience, we refer to models (3.1.3), (3.1.4), and (3.1.5) as the first-order, quadratic, and second-order position models, with m, $2m-1$, and $(m-1)(m+2)/2$ parameters, respectively.

Table 3.3 shows the number of parameters of the three models, the PWO model and the CP model for $m = 3, \ldots, 10$. The first-order and quadratic position models have fewer parameters than the others when $m > 4$. The second-order position model has a few more parameters than the PWO model, but has fewer parameters than the CP model as m increases. The new position models are both parsimonious and flexible. We demonstrate these traits using both drug sequencing experiments, each of which has two objectives: fitting an accurate model, and locating the optimal drug sequence.

Table 3.3: Number of parameters of models for m = 3–10.

	m							
Model	3	4	5	6	7	8	9	10
PWO Model	4	7	11	16	22	29	37	46
CP Model	5	10	17	26	37	50	65	82
First-order Model	3	4	5	6	7	8	9	10
Quadratic Model	5	7	9	11	13	15	17	19
Second-order Model	5	9	14	20	27	35	44	54

3.1.1 Example: Sequential Administration of Four Drugs

Consider the four-drug order-of-addition experiment in Table 3.1. We first fit the five models to the full data. The PWO and CP models have predictive R^2 of 0.67 and 0.54, respectively. The first-order, quadratic, and second-order position models have predictive R^2 of 0.69, 0.66, and 0.65, respectively. The root mean squared errors (RMSEs) for the PWO and CP models are 2.97 and 2.86, respectively, and 3.34, 3.00, and 2.67, respectively, for the position models. From this, we see that all five models have a similar goodness of fit. The first-order model with four parameters is the simplest and achieves the best predictive R^2 value.

To further compare the predictive accuracy of the models, we train each on the COA(12,4) given by the runs with $*$ in Table 3.1, and predict across all 24 sequences. The PWO and CP models have predicted versus observed correlations of 0.90 and 0.87, respectively, while the position models have correlations 0.87, 0.88, and 0.89, respectively. All models achieve comparable prediction accuracy, but the first-order model is able to do so with fewer parameters and has a better predictive R^2 when considering the full data set. Thus, for the simpler data set in which the relationship appears linear (Figure 3.1), our succinct models fit well and produce accurate predictions.

In order to interpret the position models, we first simplify each model (fit to all 24 runs) using forward and backward stepwise variable selection with respect to the AIC. We start from a constant model, and instead of removing the last effect, as in (3.1.3)-(3.1.5), we allow for the choice of any effect. The resulting models are

$$\hat{y} = 45.22 + 2.03B - 5.55C - 1.81A, \tag{3.1.6}$$

$$\hat{y} = 45.22 - 1.81A + 2.03B - 5.55C + 1.41A^2, \tag{3.1.7}$$

$$\hat{y} = 44.68 - 1.81A + 2.03B - 5.55C + 0.98A^2 - 1.62AB, \tag{3.1.8}$$

where each drug has been replaced with a letter to make the conclusions clearer (*e.g.*, A and A^2 represent the linear and quadratic effects, respectively, of drug 0). The predictive R^2 values for these three models are 0.69, 0.72, and 0.74, and the RMSEs are 3.34, 3.03, and 2.76, respectively. Further examination reveals that A, A^2, and A^2 and AB are not significant at the 5% level in models (3.1.6)-(3.1.8), respectively. After removing the nonsignificant terms, we have the reduced model $\hat{y} = 45.22 + 2.93B - 4.65C$.

In this model, the negative coefficient of drug C can be interpreted as the response being maximized when it comes earlier in the sequence, and the positive coefficient of drug B signifies that the response increases when it is placed later. These interpretations reflect the linear trends we see in the four-drug component-position effects plot in Figure 3.1. Mee (2020) and Yang et al. (2021) also performed a stepwise regression to simplify the PWO and CP models, respectively. Their simplified PWO and CP models are comparable to models (3.1.6)-(3.1.8) in terms of their predictive R^2.

3.1.2 Example: Sequential Administration of Five Drugs

Consider the five-drug order-of-addition experiment in Table 3.2. The experiment was conducted in batches. The first 20 runs were used in a batch, and the second 20 runs were used in another batch. After fitting each model to all 40 runs, including a block variable representing the batch effect, the PWO and CP models have predictive R^2 values of 0.20 and 0.09, respectively, and the first-order, quadratic, and second-order position models have predictive R^2 values of 0.44, 0.41, and 0.52, respectively. The RMSEs for the PWO and CP models are 4.11 and 3.45, respectively, and 4.18, 3.80, and 2.85, respectively, for the position models. The position models show a greater ability to capture the nonlinear trends present in Figure 3.1. The second-order model not only produces the overall best fit, but also generalizes well.

In order to improve interpretability and keep the final model concise, variable selection is used to choose the most appropriate effects to include from the second-order model. Starting with a constant model, forward and backward stepwise regressions are used to produce a model with a small AIC. Because the choice of which effects to remove from the position models was arbitrary, we allow for the selection of any linear, quadratic, or two-factor interaction effects, as in the previous example. We also allow for the selection of a block effect that represents the two batches. With this in mind, the resulting model has a total of eight terms, has a predictive R^2 of 0.68 (larger than any competitor), and is given by

$$\hat{y} = 23.13 - 4.08\Delta + 3.19A + 3.45B + 4.49D + 1.05C^2 + 1.82BE - 1.64CE. \tag{3.1.9}$$

In this model, the block variable is given by Δ, and each drug is again represented by a letter to facilitate substantive conclusions. We see that the quadratic effect of drug C and two interactions involving drug E are included in the final model. Owing to the constraints in (3.1.2), we have $A+B+C+D+E=0$. Therefore, if we replace A with $-B-C-D-E$, then we get an equivalent model that follows the effect hierarchy principle (Wu and Hamada, 2009).

A direct interpretation of these significant effects is complicated by the inclusion of C^2, BE, and CE, so we consider the top 10 predicted sequences: CEBAD, CAEBD, CEABD, CBEAD, CADEB, CEBDA, EBADC, CAEDB, CABED, and EBACD. While most of these sequences are not in the Table 3.2 design, the sequence CAEBD has the second-highest predicted response and the second-highest observed response.

In order to overcome the shortcomings of the PWO model when fitting to the data, Mee

(2020) considered the triplets ordering model given in (2.3.2). Also using a forward stepwise regression, the author found two models that include some of the additional interaction terms. Our stepwise model (3.1.9) with df=32, predictive $R^2 = 0.68$, and RMSE = 3.32 is competitive with both of these triplets models (df = 24, predictive $R^2 = 0.60$, RMSE = 3.14, and df = 26, predictive $R^2 = 0.51$, RMSE = 3.73), and becomes more appealing when considering the use of fewer parameters. Furthermore, the top two predicted sequences from both of these models are CAEBD and CEBAD, aligning with the top two predicted sequences from the position model. Our model is also better than the PWO and CP models with interactions reported by Yang et al. (2021) in terms of various measures, including the predictive R^2 and RMSE. This further substantiates our claim that the second-order model is able to achieve an intuitive and cost-effective fit on complex order-of-addition data, which until now has not been possible.

Note that while our models fit well to the real data in these examples, they assume that the absolute rather than the relative component positions are most predictive of the response. While the substantive conclusions are similar between the two model types, it is important to recognize that, in practice, the details of the application should be considered when assuming a model. For example, the absolute position assumption may be more valid in the drug administration problem, in which early exposure to a drug may produce better results. On the other hand, the relative position assumption makes more sense in cases where the components are known to react with each other, as in the experiments considered by Voelkel and Gallagher (2019).

3.2 Design Construction and Optimality

The encouraging results from the previous examples inspire us to study the properties of these new models and the possibility of constructing optimal or highly-efficient designs for them. Because we do not know in advance which model is the best for a given application, we would like to have a class of designs that can perform well under different model assumptions and various run sizes. To achieve this goal, we develop a construction method based on the properties of Latin squares. For many values of m, this construction can quickly generate efficient designs of any run size and in certain cases achieves D-optimality for the position-based models.

3.2.1 Design Construction Algorithm

For a prime or a prime power m, let $GF(m) = \{\omega_0, \omega_1, \dots, \omega_{m-1}\}$ be a Galois field of order m, with ω_0 being the zero element (Barker, 1986). When m is prime, $GF(m) = \{0, 1, \dots, m-1\}$ is a ring of integers modulo m. The following algorithm constructs an $n \times m$ design for any $n \leq m!$.

Algorithm 3.1.

Step 1. *For $k = 1, \dots, m-1$, define an $m \times m$ matrix $\boldsymbol{L}_k$ such that its $(i, j)^{th}$ element is $\omega_i + \omega_k * \omega_j$, for $i, j = 0, \dots, m-1$, where the addition and multiplication are defined on $GF(m)$.*

Step 2. *Construct an $(m^2 - m) \times m$ matrix $\boldsymbol{C}_1$ by row-wise concatenating $\boldsymbol{L}_1$, ..., $\boldsymbol{L}_{m-1}$.*

Step 3. *Keep the first two columns of $\boldsymbol{C}_1$ fixed and permute the last $m-2$ columns of $\boldsymbol{C}_1$ in a systematic way. There are $(m-2)!$ permutations, admitting a total of $(m-2)!$ permuted matrices, denoted as $\boldsymbol{C}_1$, ..., $\boldsymbol{C}_{(m-2)!}$.*

Step 4. *Construct an $m! \times m$ matrix $\mathbf{F}_m$ by row-wise concatenating $\boldsymbol{C}_1$, ..., $\boldsymbol{C}_{(m-2)!}$ and replacing ω_i with number i, for $i = 0, \dots, m-1$.*

Step 5. *Let $\mathbf{F}_{n,m}$ be the $n \times m$ design formed by the first n rows of $\mathbf{F}_m$.*

Step 6. *Permute the columns of $\mathbf{F}_{n,m}$ to improve its performance under a chosen criterion.*

Each $\boldsymbol{L}_k$ in Step 1 is an $m \times m$ Latin square, and the $(m-1)$ Latin squares ($\boldsymbol{L}_1$, ..., $\boldsymbol{L}_{m-1}$)

are mutually orthogonal.(Two Latin squares are orthogonal if, when they are superimposed, each pair (i, j) appears exactly once for any $i, j = 0, \ldots, m-1$.) Mutually orthogonal Latin squares are traditionally used to construct balanced incomplete block designs and orthogonal arrays. We use them for a different purpose.

The design $\boldsymbol{C}_1$ constructed in Step 2, as well as any $\boldsymbol{C}_i$ in Step 3, is a COA$(m^2 - m, m)$. Any pair of $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ in Step 3 do not share any common permutations. The $m! \times m$ matrix $\mathbf{F}_m$ constructed in Step 4 consists of all $m!$ permutations of m components. Step 5 simply chooses the first n rows of $\mathbf{F}_m$ as a candidate design, which often has good properties already. Specifically, Anderson-Cook and Lu (2019) outline the benefits of constructing designs from Latin squares and choosing a run size that is a multiple of m. Step 6, to be discussed later, can be used to further improve the design according to a specific criterion.

We consider the $m = 4$ case, where the full design $\mathbf{F}_m$ consists of 24 permutations in the order given in Table 3.4. The first four permutations form a 4×4 Latin square $\boldsymbol{L}_1$, the next four permutations form another Latin square $\boldsymbol{L}_2$, and so on. The first 12 permutations form a COA(12,4). The last 12 permutations are obtained from the first 12 by permuting the last two columns.

When $n = m(m-1)$, $\mathbf{F}_{n,m}$ is equivalent to the COA$(m(m-1), m)$ constructed by Yang et al. (2021). However, their construction does not provide designs with other run sizes. When m is not a prime power, a Galois field of order m does not exist. In these cases, an exchange algorithm may produce good designs or the recursive construction of COA designs for any m presented in Huang (2021) may be used as a starting point.

3.2.2 Theoretical Guarantees

To assess the orthogonality properties of different types of designs, we use the GWLP function in the R package DoE.base (Groemping et al., 2014) to compute the GWLP of the component matrix $\boldsymbol{A}$. Recall that the GWLP measures the contamination of lower-order effect estimates

Table 3.4: The full 24-run design $\mathbf{F}_4$, as generated by Algorithm 3.1.

Run			a_1	a_2	a_3	a_4	Run		a_1	a_2	a_3	a_4
1			0	1	2	3	13		0	1	3	2
2		$\boldsymbol{L}_1$	1	0	3	2	14		1	0	2	3
3			2	3	0	1	15		2	3	1	0
4			3	2	1	0	16		3	2	0	1
5			0	2	3	1	17		0	2	1	3
6	$\boldsymbol{C}_1$	$\boldsymbol{L}_2$	1	3	2	0	18	$\boldsymbol{C}_2$	1	3	0	2
7			2	0	1	3	19		2	0	3	1
8			3	1	0	2	20		3	1	2	0
9			0	3	1	2	21		0	3	2	1
10		$\boldsymbol{L}_3$	1	2	0	3	22		1	2	3	0
11			2	1	3	0	23		2	1	0	3
12			3	0	2	1	24		3	0	1	2

by higher-order effects and can tell us about the orthogonality, strength and robustness properties of a design. We also note that the component matrix $\boldsymbol{A}$ and the position matrix $\boldsymbol{B}$ have the same GWLP, provided that every component appears in every position at least once. With these ideas in mind, the designs $\mathbf{F}_{n,m}$ produced by Algorithm 3.1 have several desirable properties. All proofs can be found in Appendix A.

Theorem 3.1. *The design $\mathbf{F}_{n,m}$ has the following properties:*

(i) For any $n = qm+r$ with integers $q > 0$ and $0 \leq r < m$, $\mathbf{F}_{n,m}$ has $W_1 = mr(m-r)/n^2$, which is the minimum among all possible designs with n runs, m columns, and m levels.

(ii) If n is a multiple of m, then $\mathbf{F}_{n,m}$ has $W_1 = 0$.

(iii) If $m \leq n \leq m(m-1)$, then $\mathbf{F}_{n,m}$ has generalized minimum aberration among all possible designs with n runs, m columns, and m levels.

(iv) If n is a multiple of $m(m-1)$, then $\mathbf{F}_{n,m}$ is a COA(n,m).

Theorem 3.1 (i) and (ii) imply that our designs always have the desirable property of being level balanced or nearly balanced for each column. Theorem 3.1 (iii) and (iv) indicate that our designs tend to minimize the correlation between columns and reduce aliasing among first-order and second-order effects. Finally, having shown in Theorem 3.1 (iv) that $\mathbf{F}_{n,m}$ is a COA when n is a multiple of $m(m-1)$, Theorem 3.2 shows that these designs are D-optimal under the first-order and quadratic models.

Cheng et al. (2002) and Mandal and Mukerjee (2005) showed that generalized minimum aberration designs have high efficiency under model uncertainty for factorial experiments. Thus, Theorem 3.1 implies that the designs constructed from our algorithm have high efficiency under various models for order-of-addition experiments. Examples in the next section provide evidence of this property.

We can also evaluate the constructed designs using the D- and A-optimality criteria (See Section 2.1.3). It is known that the full design $\mathbf{F}_m$ with all $m!$ permutations is D-optimal for both the PWO model and the CP model (Peng et al., 2019; Yang et al., 2021). Additionally, Peng et al. (2019) showed that this design is also A-optimal for the PWO model. We can therefore compare the quality of any proposed design to this optimal one using D- and A-efficiency.

However, we need to determine whether the full design $\mathbf{F}_m$ is indeed optimal under the three position models in order to compare candidate designs. For this we rely on the checking condition for optimality provided by the general equivalence theorem (Silvey, 1980). In the case of the three position models, each $\boldsymbol{x}$ is a permutation, and $\mathcal{X}$ is the space of all permutations of $\{1, \ldots, m\}$. We have the following important results regarding the full design and COAs.

Theorem 3.2. *The full design* $\mathbf{F}_m$ *is D-optimal under the first-order and quadratic position models, as is every* $\mathrm{COA}(n, m)$*.*

Theorem 3.3. *The full design* $\mathbf{F}_m$ *is D-optimal for the second-order position model.*

Note that the result of Theorem 1 from Peng et al. (2019), which shows the optimality of $\mathbf{F}_m$ under the PWO model for any concave and signed permutation invariant criteria, does not apply to the three position models. As a counterexample, we consider the A-optimality criterion. If the result of their theorem held for the position models, then we would be able to confirm A-optimality of the full design numerically. However, as shown below, this is not the case. Furthermore, the information matrices for the first-order and quadratic models are block diagonal (see the proof in Appendix A), yet the closed form of the information matrix for the second-order model is too complex to work with directly. However, the proof of Theorem 3.3 is general and can be applied to the PWO and CP models, along with other models, such as a third-order model that includes all estimable terms.

The D-efficiency of a design varies with respect to column permutations; that is, permuting the columns of a design may lead to different D-efficiencies. For this reason, we can permute the columns in Step 6 to maximize the D-efficiency for a specific model. We consider other opportunities for improved efficiency using level and $\boldsymbol{C}_i$ permutations in the next section. In contrast, the GWLP is invariant with respect to column permutations, and instead studies the combinatorial properties of the design, such as balance and orthogonality.

Through preliminary investigation, we have found that the design $\mathbf{F}_m$ does not satisfy the checking condition (2.1.5), and is thus not A-optimal, for any of the position models. Using a popular metheuristic algorithm, Differential Evolution (Storn and Price (1997); Chakraborty (2008)), we have found nearly A-optimal designs for the position models for several values of m. Table 3.5 shows the relative efficiency of $\mathbf{F}_m$ to these designs and indicates that the full design is indeed sub-optimal in most cases, with its efficiency growing worse as m increases. Furthermore, we have found that A-optimal designs under our models depend on specifically which component effects are removed from the model to make it estimable. Since our decision to remove the effect of component $m-1$ was in large part arbitrary, this phenomenon requires further study with the goal of producing A-optimal designs which are robust to this choice. D-optimality remains the most popular design criterion, so we focus

Table 3.5: Relative A-efficiency of $\mathbf{F}_m$ to near-optimal designs.

m	first-order	quadratic	second-order
3	0.951	1	0.951
4	0.909	0.987	0.885
5	0.879	0.934	0.812

on it for the remainder of this chapter, but we consider the problem of finding A-optimal designs for the position models more thoroughly in Chapter 5.

3.3 Efficiency and Robustness of Designs

With this new construction in hand, and its positive theoretical properties in mind, we can compare the performance of the resulting designs to that of the existing methods. This includes studying the efficiency of these and other designs for non-fractional run sizes and measuring the orthogonality properties that allow for clear effect estimation. Furthermore, there are some structural decisions to be made in the construction, leading to uncertainty in the algorithm. We would like to understand how robust the construction is to these decisions in addition to the different assumptions about the structure of the ordering effect present in the models.

3.3.1 Design Efficiency Comparison

We start by comparing our designs with those from Voelkel (2019) with $m = 4, 5, 7$ and various run sizes. We also compare them with the design given in Table 3.2 from Mee (2020) with $(n, m) = (40, 5)$. In order to present a fair comparison, we consider one design $\mathbf{F}_{n,m}$, which is generated by taking the first n rows of the full design $\mathbf{F}_m$ in Step 5, and another design $\mathbf{F}^*_{n,m}$, which permutes the columns of $\mathbf{F}_{n,m}$ in Step 6 to maximize the geometric mean efficiency of the models of interest. Table 3.6 compares the D-efficiencies of these designs under the five models and the first two terms (W_1 and W_2) of the GWLP. Designs $\mathbf{F}^*_{n,m}$ for which there is no improvement over $\mathbf{F}_{n,m}$ are omitted. While Voelkel (2019) used many other criteria to compare his order-of-addition designs, many of these are derivatives of the two we consider here, and are thus not necessary to include. Because our algorithm is able to generate designs with variable run sizes, we also include the efficiencies of designs with various n between $m(m-1)$ and $m!$.

Voelkel's designs are constructed for the PWO model, and thus perform well under this model. However, they exhibit poor performance under the CP model and have large W_1 or W_2 values. In contrast, our designs are robust and perform well under all models (with the exception of the PWO model in certain situations) and always have small W_1 and W_2 values. Recall $W_1 = 0$ if and only if a design is level balanced for each column. Voelkel's designs are not level balanced, as can be seen in Table 3.7, except for one case (Voelkel.12a), while our designs are all level balanced or nearly balanced. Mee's design performs comparably to $\mathbf{F}_{40,5}$ for all models except the PWO model, which it outperforms. When allowing for column permutations, Mee's design has similar properties to those of $\mathbf{F}^*_{40,5}$. In general, the $\mathbf{F}^*_{n,m}$ designs in Table 3.6, which maximize the geometric mean efficiency, may not necessarily be optimal for any of the five models, but they are model robust and have high D-efficiencies for all models, a point we will now explore further.

Figure 3.2 shows the maximal D-efficiency obtained using Algorithm 3.1 for each model across a range of run sizes, n, by using a brute force search over all column permutations in Step 6. From these plots, we find that the algorithm is able to produce highly efficient designs for many values of n. Specifically, we find that with a proper selection of a column permutation, our designs achieve high D-efficiency ($> 85\%$) for every model. However, we are also curious about the general effect of the choice of column permutation, the permutation of the design levels, and the ordering of the $\boldsymbol{C}_i$. These questions, along with robustness to model misspecification are briefly considered in the next section.

Table 3.6: Comparison of D-efficiency and GWLP across designs and models.

			D-efficiency					GWLP	
n	m	Design	D_{PWO}	D_{CP}	D_{FO}	D_{PQ}	D_{SO}	W_1	W_2
12	4	Voelkel.12a	1	0.758	1	0.955	0.953	0	4.667
		Voelkel.12b	1	0	1	0.916	0.877	0.361	2.514
		$\mathbf{F}_{12,4}$	0.909	1	1	1	1	0	2
20	5	Voelkel.20a	0.903	0.588	0.997	0.945	0.861	0.35	10.35
		Voelkel.20b	0.97	0.623	0.993	0.931	0.854	0.625	8
		$\mathbf{F}_{20,5}$	0	1	1	1	0.959	0	2.5
		$\mathbf{F}^*_{20,5}$	0.898	1	1	1	0.950	0	2.5
24	5	Voelkel.24a	1	0.094	1	0.969	0.933	0.573	8.281
		Voelkel.24b	1	0	1	0.967	0.94	0.639	7.635
		Voelkel.24c	1	0.668	1	0.969	0.944	0.226	8.368
		$\mathbf{F}_{24,5}$	0.545	0.961	0.99	0.982	0.949	0.035	3.75
		$\mathbf{F}^*_{24,5}$	0.926	0.961	0.996	0.981	0.950	0.035	3.75
40	5	Mee.40	0.969	1	1	1	0.994	0	2.5
		$\mathbf{F}_{40,5}$	0.889	1	1	1	0.999	0	2.5
		$\mathbf{F}^*_{40,5}$	0.969	1	1	1	0.995	0	2.5
48	7	Voelkel.48	0.986	0.587	1	0.950	0.872	0.924	17.099
		$\mathbf{F}_{48,7}$	0	0.967	0.993	0.985	0.876	0.018	5.688
		$\mathbf{F}^*_{48,7}$	0.943	0.967	0.995	0.989	0.798	0.018	5.688

Note: D-efficiency: the larger, the better; GWLP: the smaller, the better. D_X is the D-efficiency under model X (PWO, CP, first-order, pure quadratic, and second-order, respectively). In some cases, the chosen run size does not permit estimation of some models (with D-efficiencies marked by "-"). Designs $\mathbf{F}_{n,m}$ are obtained via Algorithm 3.1 without Step 6, while $\mathbf{F}^*_{n,m}$ are obtained with column permutations in Step 6 to maximize the geometric mean efficiency of the estimable models.

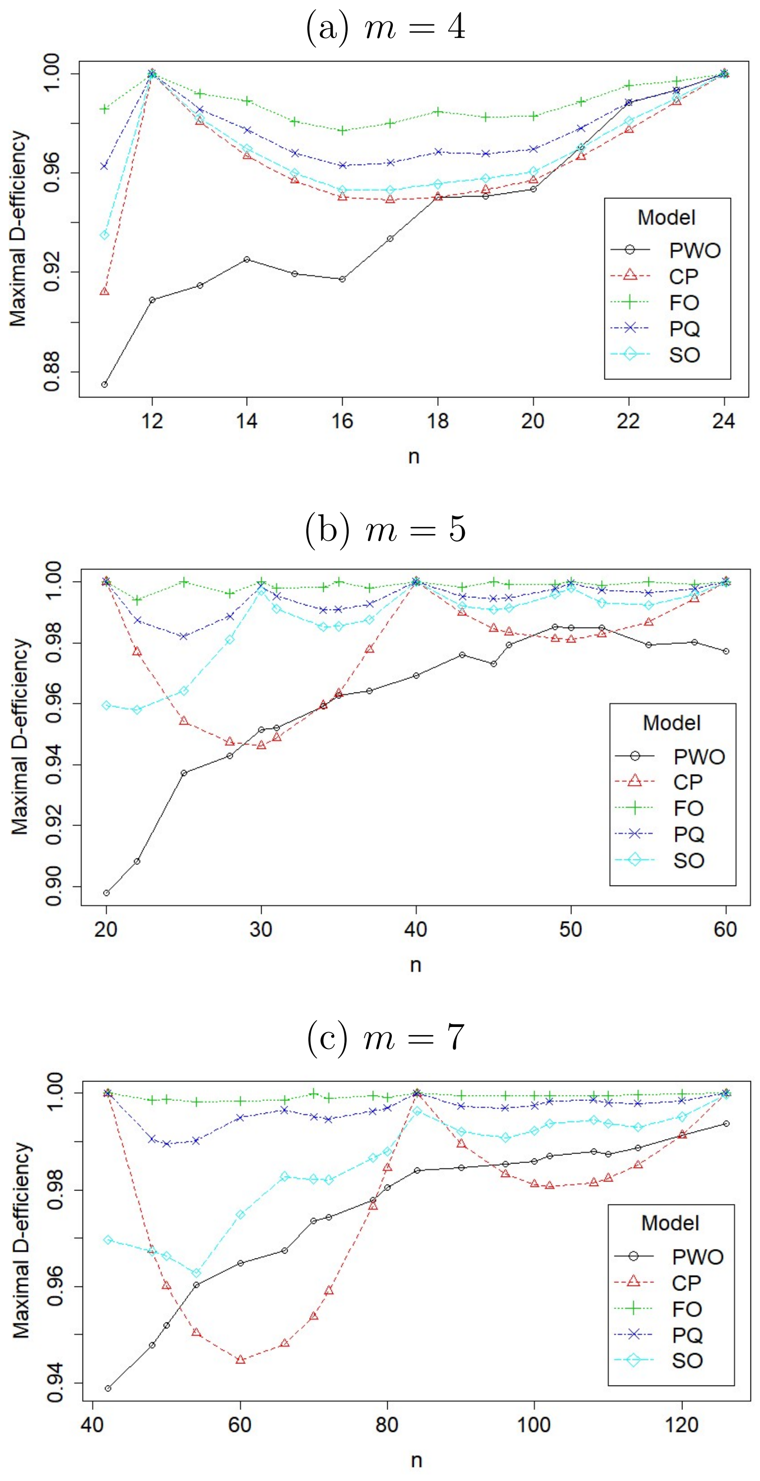


Figure 3.2: Maximal D-efficiency of $\mathbf{F}^*_{n,m}$ under column permutations for variable run sizes for (a) $m = 4$, (b) $m = 5$, and (c) $m = 7$.

Table 3.7: The 4-component, 12-run designs from Voelkel (2019).

Voelkel.12a										Voelkel.12b									
Run	a_1	a_2	a_3	a_4	Run	a_1	a_2	a_3	a_4	Run	a_1	a_2	a_3	a_4	Run	a_1	a_2	a_3	a_4
1	0	1	3	2	7	2	0	3	1	1	0	2	1	3	7	1	3	0	2
2	0	2	1	3	8	2	1	0	3	2	0	2	3	1	8	2	0	1	3
3	0	3	1	2	9	2	3	0	1	3	0	3	1	2	9	2	3	1	0
4	1	0	2	3	10	3	0	2	1	4	1	0	3	2	10	3	0	1	2
5	1	2	3	0	11	3	1	0	2	5	1	2	0	3	11	3	2	0	1
6	1	3	2	0	12	3	2	1	0	6	1	2	3	0	12	3	2	1	0

3.3.2 Robustness Properties

In addition to column permutations, the assignment of values to $\omega_0, \omega_1, \ldots, \omega_{m-1}$ in Step 1 and the order of permutations of the last $m-2$ columns to create the $\boldsymbol{C}_i$ in Step 3 can both be manipulated. To understand the effects of these permutations, we repeat the algorithm many times, with each iteration using a different combination of level, $\boldsymbol{C}_i$, and column permutations. This detailed study has demonstrated that for small values of m, the effect of the choice of permutations on efficiency is large under the PWO model, and small for the other models. Upon studying each choice of permutation in turn, we find that improvements to the D-efficiency of the best column-permuted design are small when allowing for level and $\boldsymbol{C}_i$ permutations. This justifies the inclusion of column permutations in Step 6 of Algorithm 3.1. The full results of this study can be found in Appendix B.

Furthermore, having shown that Algorithm 3.1 can, in general, produce designs that are optimal or near-optimal for many models when accounting for choices in the algorithm, we now examine the robustness of our designs to model misspecification. For example, we see from Table 3.6 that Voelkel's designs have lower efficiency under the CP model, while our designs $\mathbf{F}_{n,m}$ have lower efficiency for the PWO model. If we design our experiment under the assumption that one of these models fits the experimental data, when in reality a different model captures the trend, then we run the risk of choosing an inefficient design. To test the ability of our designs to withstand such an error, we consider the trade-off in efficiency for different levels of confidence in our selection of the true model. Here, we find that our designs are robust to misspecification, and specifically to the assumption that relative/absolute position effects are most relevant. The full details of this study can also be found in Appendix B.

3.4 Chapter Summary

In this chapter, we have proposed succinct models and cost-effective order-of-addition designs for accurately capturing important trends. Through careful research, we have seen that our models yield a superior fit and interpretable estimates, while our designs are optimal in many cases and robust to model misspecification. Note, however, that all of the models we consider, with the exception of the PWO model, are based on the absolute position effects assumption. Were we to consider further extensions of the PWO model, such as those proposed in Voelkel and Gallagher (2019) or Mee (2020), we may see different results. Furthermore, Schoen and Mee (2021) have recently found designs for $m = 5, 6, 7$ that are optimal under the PWO model and exhibit stronger balance than Voelkel's. Such designs may be more appropriate if there is strong confidence in the relative position assumption, but we do not consider them here.

Applications to sequential drug administration have further demonstrated the scientific value of these methods to the broader research community. However, there is still much work left to be done in this field. Mee (2020) briefly discussed the idea of ordering restrictions, yet there are many constrained situations for which no appropriate designs exist. Additionally, standard approaches for combining designs for the ordering effect with those for additional covariates result in experiments too large to be of much practical use. In the next chapter, we consider one such related problem in which the size of the component pool is larger than the number of available positions, so we must screen the pool of components to find the effect of each component's inclusion in addition to its position. There are many practical applications of this problem, yet there are no designs for studying it.

CHAPTER 4

Component Screening Designs for Order-of-Addition Experiments

We have seen that the literature on the modeling and design of order-of-addition problems is growing to meet the needs of researchers across a diverse set of applied areas. However, despite the growing amount of literature there are many open problems for which designed experiments are not available. For example, in some cases a researcher may have a larger pool of components than the number of available positions can accommodate. Putting this in the context of sequential drug administration, when working with a large collection of anti-tumor drugs, the practical consideration of a patient's willingness to take many drugs at once may limit the maximum number of positions available in the order. In such a situation the experimenter needs to screen the drug combinations to determine which set produces the best result, while also understanding the impact of the drug sequence on the response. To our knowledge this problem has not yet been studied in the context of order-of-addition, yet the same combinatorial explosion that impedes the standard order-of-addition problem is present here. Thus, in the second major contribution of this dissertation, we formalize this setup and develop efficient, robust designs for conducting such experiments.

In this chapter, we first make adjustments to two prominent order-of-addition models that can then be used to capture the effects of the component subset and sequence on the measured response in the screening problem. Next, we offer several design construction algorithms for choosing D-optimal subsets of the full design for different choices of the model, size of the component pool, and number of available positions by leveraging the properties of existing designs for the standard order-of-addition problem. To demonstrate the value of these designs we consider the practical application of our order-of-addition screening designs to synthetic job scheduling problems in the context of both a single-shot experiment and an active learning framework for sequential experimentation. We find that the proposed designs offer clear effect estimation and accurate predictions when treated as a single-shot experiment and fast convergence to the optimal ordering in sequential experiments. All proofs from this chapter are given in Appendix A.

4.1 Screening Models and Full Design Optimality

We assume that the number of available positions in the order is fixed throughout the experiment and is represented by q with $1 < q < m$, where m is the total number of available components. The m components are denoted for convenience as $0, 1, \ldots, m-1$. Under this setup there are a total of $\binom{m}{q}q! = \frac{m!}{(m-q)!}$ possible ways to assign the m components to the q positions. Each order-of-addition design is given in terms of a component matrix $\boldsymbol{A}$ in which each column $\boldsymbol{a}_j$ represents position j and a_{ij} gives the component added in position j of run i. We refer to the design which contains every possible subset/permutation pair as the full screening design, denoted by $\mathbf{S}_{m,q}$. For $m = 5, q = 3$ the component matrix of $\mathbf{S}_{5,3}$ is given in Table 4.1. It is clear from this table that the number of possible sequences grows quickly as the number of components increases. With no existing designs to cover this case, our aim is to construct efficient order-of-addition screening designs that include only a small fraction of the total set of sequences.

To analyze data from screening experiments, we consider two existing order-of-addition models with slight changes to accommodate the screening problem. First is the CP model from Yang et al. (2021) given in (2.4.10). Typically the constraint $\sum_{k=1}^{m} z_{kj}^{(i)} = 1$ for any i and j necessitates that we remove the effect of one component and one position. However, in our case each run is a permutation of at most $m-1$ distinct components instead of a full

Table 4.1: Full screening design $\mathbf{S}_{5,3}$ with $m = 5$ components and $q = 3$ positions.

Run	a_1	a_2	a_3
1	0	1	2
2	0	2	1
3	1	0	2
4	1	2	0
5	2	0	1
6	2	1	0
7	0	1	3
8	0	3	1
9	1	0	3
10	1	3	0
11	3	0	1
12	3	1	0
13	0	1	4
14	0	4	1
15	1	0	4
16	1	4	0
17	4	0	1
18	4	1	0
19	0	2	3
20	0	3	2
21	2	0	3
22	2	3	0
23	3	0	2
24	3	2	0
25	0	2	4
26	0	4	2
27	2	0	4
28	2	4	0
29	4	0	2
30	4	2	0
31	0	3	4
32	0	4	3
33	3	0	4
34	3	4	0
35	4	0	3
36	4	3	0
37	1	2	3
38	1	3	2
39	2	1	3
40	2	3	1
41	3	1	2
42	3	2	1
43	1	2	4
44	1	4	2
45	2	1	4
46	2	4	1
47	4	1	2
48	4	2	1
49	1	3	4
50	1	4	3
51	3	1	4
52	3	4	1
53	4	1	3
54	4	3	1
55	2	3	4
56	2	4	3
57	3	2	4
58	3	4	2
59	4	2	3
60	4	3	2

permutation of all m components. Thus, we do not need to remove the effect of one of the positions to make the model estimable. The component-position screening model (CPS) is then

$$y = \gamma_0 + \sum_{k=1}^{m-1} \sum_{j=1}^{q} z_{kj}\gamma_{kj} + \varepsilon, \tag{4.1.1}$$

where y is the response, γ_0 is the intercept, z_{kj} is an indicator for the component-position pair (k, j), γ_{kj} is the parameter representing the effect of component k being added at the j^{th} position, and ε is an independent normal random error. In addition to this version of the CP model we will also use the COA designs proposed in Yang et al. (2021) and discussed in Section 2.4 as building blocks for order-of-addition screening designs.

We also consider the pairwise ordering model first introduced by Van Nostrand (1995) and Voelkel (2019) given in (2.3.8). In this model a set of psuedo-factors $\{I_{ij},\ 0 \le i < j \le m\}$ is created such that each corresponds to the pairwise ordering of the components. In the standard model each factor I_{ij} has two levels, 1 and -1, indicating whether or not component i is added before component j. However, in the screening case not every component is present in the sequence, so for each I_{ij} in which component i or j is missing from the sequence we assign a value of 0. With this change, the pairwise ordering screening model (PWOS) is given by

$$y = \beta_0 + \sum_{i<j} \beta_{ij} I_{ij} + \varepsilon, \tag{4.1.2}$$

with random error $\varepsilon \sim N(0, \sigma^2)$.

In addition to this version of the model we will use the OofA-OA class of designs that was proposed by Voelkel and Gallagher (2019) and has since been expanded by Mee (2020) and Schoen and Mee (2021). The properties of OofA-OAs are given in Section 2.3. These designs will provide a basis for the construction of order-of-addition screening designs under the PWOS model.

Note that while we have relabeled the CP and PWO models as CPS and PWOS, respectively, this is done only to differentiate this study from the one in the previous chapter concerning the standard order-of-addition problem. The fundamental structure of each model is largely unchanged up to the minor alterations required to accommodate the data from component screening experiments. It remains the subject of future research to consider other modeling approaches (*e.g.*, position-based models, Gaussian Process models, etc.). With this in mind, the first step in finding smaller designs for order-of-addition screening experiments is to show that the full design is optimal for the two models discussed above, so that we may use it as a reference design for future designs. To meet this end we have the following results:

Theorem 4.1. *The full design $\mathbf{S}_{m,q}$ is D-optimal for the CPS model (4.1.1) with $m \geq 3$ and $1 < q < m$.*

Theorem 4.2. *The full design $\mathbf{S}_{m,q}$ is D-optimal for the PWOS model (4.1.2) with $m \geq 3$ and $1 < q < m$.*

With these results, we can now compare the quality of any proposed design to this optimal one. We can calculate the *D*-efficiency under the chosen model using (2.1.3) where

$\boldsymbol{\xi}_2 = \mathbf{S}_{m,q}$. With these preliminary steps complete, we are ready to construct smaller, optimal and near-optimal order-of-addition screening designs.

4.2 Component Screening Design Constructions

Considering the two models described in the previous section, we offer two primary design constructions and a third construction for a special case not covered by the other two. The two primary constructions are built upon existing order-of-addition designs. To motivate the construction of order-of-addition screening designs for the CPS model, Alogirthm 4.1 utilizes the flexible construction of COA-based designs proposed in Chapter 3. For the PWOS model Algorithm 4.2 considers the special case of $q = 3$ with even m while Alogirthm 4.3 takes advantage of the properties of the OofA-OAs constructed in Schoen and Mee (2021) to cover the remaining cases. Each of these constructions produces fractional designs for variable m and q that are D-optimal under one or both models. For each method, we establish settings of m, q and n under which the resulting design is D-optimal. To understand the robustness of our designs to model misspecification, we explore the efficiency of the designs produced for one model under the other.

4.2.1 Optimal Design Construction for the CPS Model

In Chapter 3, we showed that the designs generated by Algorithm 3.1 for the standard order-of-addition problem, denoted by $\mathbf{F}_{n,m}$ for n runs in m components, can achieve high efficiency on the standard CP and PWO models with minor tuning. We base our construction of efficient designs for the CPS model on these designs. Using the $\mathbf{F}_{n,m}$ designs we propose the following algorithm for constructing order-of-addition screening designs with a pool of m components, n runs and q positions.

Algorithm 4.1.

Step 1. *Generate the $n \times m$ matrix $\mathbf{F}_{n,m}$ using Algorithm 3.1.*

Step 2. *Construct an $n \times q$ matrix ${\mathbf{S^c}}_{n,m,q}$ by taking the first q odd-numbered columns of $\mathbf{F}_{n,m}$ if $q \leq m/2$. Otherwise take the $\lceil m/2 \rceil$ odd-numbered columns followed by the first $q - \lceil m/2 \rceil$ even-numbered columns.*

Step 3. *Permute the columns of ${\mathbf{S^c}}_{n,m,q}$ to improve its performance under a chosen criterion.*

In the development of Algorithm 4.1 we have found that the choice of which columns of the design to take in Step 2 does not affect the endpoint efficiency of the design under the CPS model. However, taking the odd-numbered columns of $\mathbf{F}_{n,m}$ first followed by the even-numbered columns produces pairwise pseudo-factors in the PWOS model with better balance properties, and in turn yields much higher D-efficiency, than taking the first or last q columns or taking a random subset of columns. For small m and q, a complete search over all $\binom{m}{q}$ sub-designs to maximize the chosen criterion could be beneficial.

To illustrate the construction we consider the case $m = 5, q = 3$ in which the full design $\mathbf{S}_{5,3}$ has 60 runs as shown in Table 4.1. Instead of using this full design we construct a design in 20 runs by first generating the design $\mathbf{F}_{20,5}$ (Table 4.2a), then taking the three odd-numbered columns, and finally permuting the columns to maximize the efficiency under the PWOS model (Table 4.2b). This design is D-optimal under the CPS model and has high efficiency under the PWOS model (approximately 0.91).

Figure 4.1 further demonstrates the performance of our algorithm on the PWOS and CPS models for different settings of m and q. Specifically, the cases $(m = 5, q = 3)$, $(m = 7, q = 3)$, and $(m = 7, q = 4)$ are considered. It is important to note that the order-of-addition design $\mathbf{F}_{n,m}$ can only be constructed when m is prime or a prime power. This is a limitation that could be addressed by using the recursive construction of COA designs with a non-prime number of components presented in Huang (2021) in Step 1 of Algorithm 4.1.

From this figure we see that the construction yields designs with high D-efficiency relative

Table 4.2: (a) 20-run optimal order-of-addition design with $m = 5$, $\mathbf{F}_{20,5.}$ (b) 20-run D-optimal screening design under the CPS model with $m = 5, q = 3$, $\mathbf{S^c}_{20,5,3}$, generated from Algorithm 4.1.

(a)

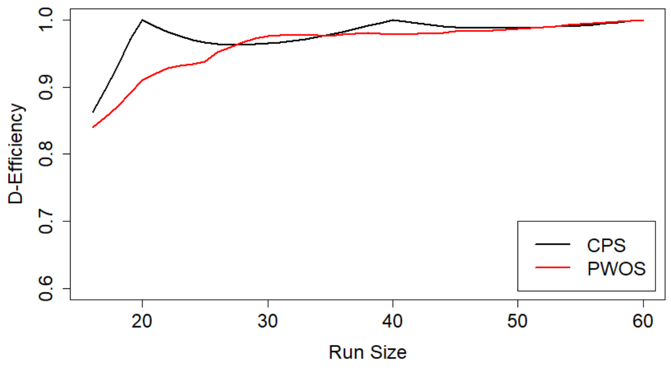
Run	a_1	a_2	a_3	a_4	a_5
1	0	1	2	3	4
2	1	2	3	4	0
3	2	3	4	0	1
4	3	4	0	1	2
5	4	0	1	2	3
6	0	2	4	1	3
7	1	3	0	2	4
8	2	4	1	3	0
9	3	0	2	4	1
10	4	1	3	0	2
11	0	3	1	4	2
12	1	4	2	0	3
13	2	0	3	1	4
14	3	1	4	2	0
15	4	2	0	3	1
16	0	4	3	2	1
17	1	0	4	3	2
18	2	1	0	4	3
19	3	2	1	0	4
20	4	3	2	1	0

(b)

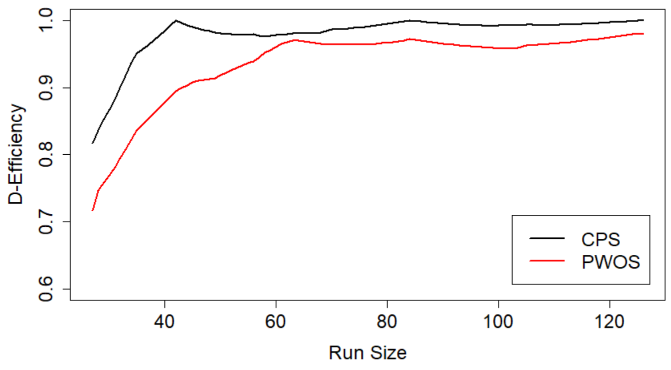
Run	a_1	a_2	a_3
1	2	0	4
2	3	1	0
3	4	2	1
4	0	3	2
5	1	4	3
6	4	0	3
7	0	1	4
8	1	2	0
9	2	3	1
10	3	4	2
11	1	0	2
12	2	1	3
13	3	2	4
14	4	3	0
15	0	4	1
16	3	0	1
17	4	1	2
18	0	2	3
19	1	3	4
20	2	4	0

to the full screening design $\mathbf{S}_{m,q}$ for both models. Under the PWOS model our designs perform well, achieving efficiency over 0.90 in most cases where the run size is suitable for estimating the model. However, in no case does our algorithm achieve the optimal design under the PWOS model. This is to be expected though, as the $\mathbf{F}_{n,m}$ designs used to generate these designs are similarly only near-optimal under the standard PWO model (See Section 3.3). On the other hand, the designs generated under the CPS model achieve high D-efficiency and in some cases optimality. In fact, we have the following general result about designs generated from Algorithm 4.1 for the CPS model.

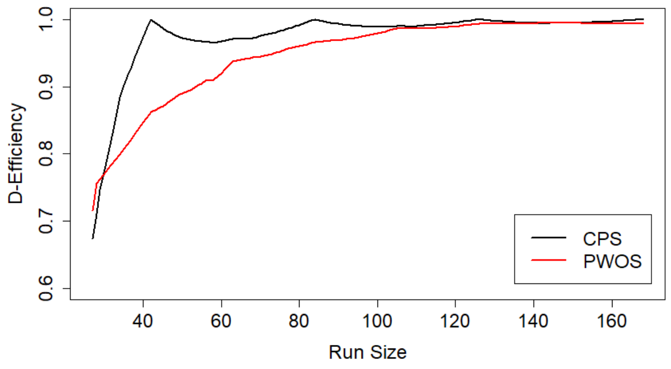
Theorem 4.3. *For $n \leq m!$ and $1 < q < m$ with n divisible by $m(m-1)$ and m a prime or*



(a)



(b)



(c)

Figure 4.1: *D*-efficiency of designs $\mathbf{S^c}_{n,m,q}$ relative to the full design $\mathbf{S}_{m,q}$ generated for variable run sizes with (a) $m = 5, q = 3$, (b) $m = 7, q = 3$, and (c) $m = 7, q = 4$.

prime power, $\mathbf{S^c}_{n,m,q}$ *is D-optimal under the CPS model.*

These results indicate that for all possible values of q our construction algorithm can generate optimal designs under the CPS model with affordable run sizes. We also see from Figure 4.1 that our designs are fairly robust to violations of the assumption of whether absolute or relative position effects are more suitable for capturing the true relationship between the order and the response.

4.2.2 Optimal Design Constructions for the PWOS Model

While the construction presented in Algorithm 4.1 produces near-optimal designs under the PWOS model, none of the resulting designs are D-optimal. To fill this gap we propose two separate constructions that can produce fractional D-optimal designs under this model. For $q \geq 4$ our designs are constructed from OofA-OAs, such as those presented in Voelkel and Gallagher (2019) and Schoen and Mee (2021). However, for $q < 4$ a separate construction is required. The following result establishes the basis of our method for constructing D-optimal designs under the PWOS model with $q = 3$.

Theorem 4.4. *For $m > 2$ the run size n of any design with D-efficiency of 1 relative to the full design under the PWOS model has the following constraints:*

(i) If $q = 2$, the minimum run size of a design with D-efficiency 1 is $n = 2\binom{m}{2}$.

(ii) If $q = 3$ and m is odd, the minimum run size of a design with D-efficiency 1 is $n = 6\binom{m}{3}$.

(iii) If $q = 3$ and m is even, the only run size for which a design with D-efficiency 1 exists with $n < 6\binom{m}{3}$ is $n = 3\binom{m}{3}$.

Considering this result, we know that no fractional design with D-efficiency 1 exists when $q = 2$ or when $q = 3$ and m is odd. For $q < 4$ this leaves only the case that $q = 3$ with even m. In this case only a half fraction optimal design exists. For any even m, the following algorithm generates this design along with an efficient design for any $n < 3\binom{m}{3}$.

Algorithm 4.2.

Step 1. *Generate the set of* $\binom{m}{3}$ *3-component combinations* $\{i, j, k\}$ *with* $0 \leq i < j < k \leq m-1$.

Step 2. *For every 3-component combination such that* $i+j+k$ *is even, construct the* 3×3 *matrix* $\boldsymbol{D}_{ijk}$ *given by*

$$\boldsymbol{D}_{ijk} = \begin{bmatrix} i & k & j \\ j & i & k \\ k & j & i \end{bmatrix}.$$

Step 3. *For every 3-component combination such that* $i+j+k$ *is odd, construct the* 3×3 *matrix* $\boldsymbol{D}_{ijk}$ *given by*

$$\boldsymbol{D}_{ijk} = \begin{bmatrix} i & j & k \\ j & k & i \\ k & i & j \end{bmatrix}.$$

Step 4. *Construct* $\mathbf{S^p}_{n,m,3}$ *by first row-wise concatenating the* $\boldsymbol{D}_{ijk}$ *for all* $0 \leq i < j < k \leq m-1$ *such that* $i+j+k$ *is even, then concatenating the remaining* $\boldsymbol{D}_{ijk}$ *and taking the first* n *rows.*

Table 4.3 demonstrates this construction method in the case of $m = 4, q = 3$. For this scenario there are 4, 3-component combinations, with $(0, 1, 3)$ and $(1, 2, 3)$ summing to an even number and $(0, 1, 2)$ and $(0, 2, 3)$ summing to odd. Concatenating the respective $\boldsymbol{D}_{ijk}$ for each of these combinations produces $\mathbf{S^p}_{12,4,3}$, a 12-run, D-optimal design for the PWOS model, a half-fraction of the 24-run full design.

Algorithm 4.2 covers the special case of generating half-fraction screening designs with $q = 3$. We note that the construction as presented can be used to generate designs for odd m, but the result will only achieve high efficiency, not optimality, as this case is not covered in the following theorem.

Table 4.3: 12-run, half-fraction D-optimal screening design for the PWOS model with $m = 4, q = 3$, $\mathbf{S^p}_{12,4,3}$, produced by Algorithm 4.2.

Run		a_1	a_2	a_3
1		0	3	1
2	$\boldsymbol{D}_{013}$	1	0	3
3		3	1	0
4		1	3	2
5	$\boldsymbol{D}_{123}$	2	1	3
6		3	2	1
7		0	1	2
8	$\boldsymbol{D}_{012}$	1	2	0
9		2	0	1
10		0	2	3
11	$\boldsymbol{D}_{023}$	2	3	0
12		3	0	2

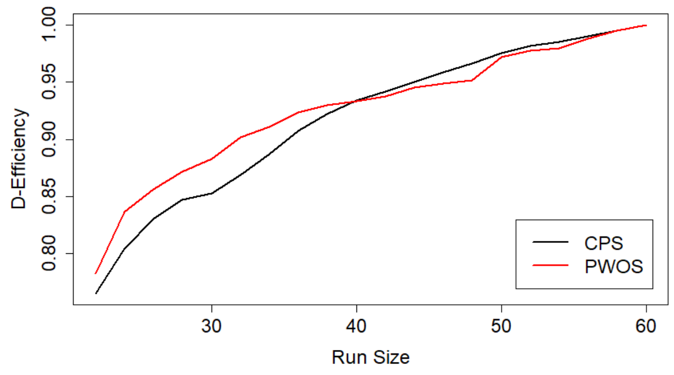
Theorem 4.5. *For $n = 3\binom{m}{3}$ and m even, $\mathbf{S^p}_{n,m,3}$ is D-optimal under the CPS and PWOS models.*

To visualize the performance of these designs we consider the cases $(m = 6, q = 3)$, $(m = 8, q = 3)$, and $(m = 10, q = 3)$ in Figure 4.2. We see from this figure that the designs generated by Algorithm 4.1 achieve high D-efficiency across both models for all run sizes considered, with the half fraction design having $n = 3\binom{m}{3}$ runs being D-optimal as determined in Theorem 4.5.

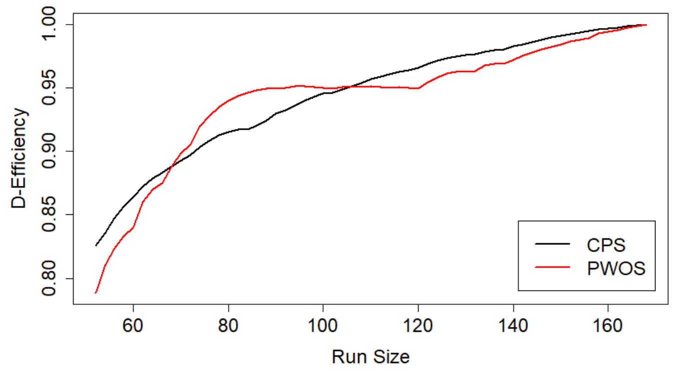
For $q \geq 4$ we instead use OofA-OAs in q components as the building blocks of D-optimal designs, leveraging the special properties of these designs detailed in Section 2.3. From Voelkel and Gallagher (2019) we know that optimal OofA-OAs must have a run size that is divisible by 12. Thus, for a given q the smallest optimal order-of-addition design has $N = 12\lceil(\binom{q}{2}+1)/12\rceil$ runs. With these properties in mind, Algorithm 4.3 generates optimal or near optimal designs for any $n \leq N\binom{m}{q}$ with $m > 4$ and $3 < q < m$.

Algorithm 4.3.

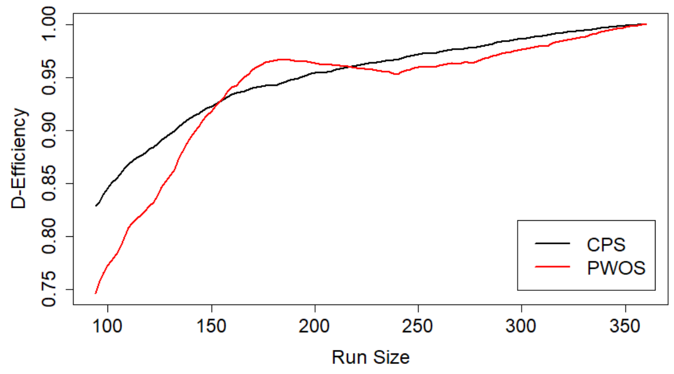
Step 1. *Construct $\boldsymbol{OA}_{N,q}$, the smallest optimal OofA-OA in q components $\{0, 1, \ldots, q-1\}$,*



(a)



(b)



(c)

Figure 4.2: D-efficiency of designs $\mathbf{S}^{\mathbf{P}}{}_{n,m,3}$ relative to the full design $\mathbf{S}_{m,3}$ generated for variable run sizes with (a) $m = 6$ (b) $m = 8$, and (c) $m = 10$.

with $N = 12\lceil (\binom{q}{2} + 1)/12 \rceil$ *runs.*

Step 2. *Generate the set of* $\binom{m}{q}$ *q-component combinations* $\{i_1, i_2, \ldots, i_q\}$ *with* $0 \leq i_1 < i_2 < \ldots < i_q \leq m-1$

Step 3. *For every q-component combination create the* $N \times q$ *matrix* $\boldsymbol{OA}_{N,q,i_1 i_2 \ldots i_q}$ *by substituting the levels of* $\boldsymbol{OA}_{N,q}$ *according to the permutation*

$$\begin{pmatrix} 0 & 1 & \ldots & q-1 \\ i_1 & i_2 & \ldots & i_q \end{pmatrix}.$$

Step 4. *Construct* $\mathbf{S^p}_{m,q}$ *by row-wise concatenating the* $\boldsymbol{OA}_{N,q,i_1 i_2 \ldots i_q}$ *for all* $0 \leq i_1 < i_2 < \ldots < i_q \leq m-1$.

Step 5. *Generate* $\mathbf{S^p}_{n,m,q}$ *by rearranging the rows of* $\mathbf{S^p}_{m,q}$ *according to the following process:*

1. *Index each q-component combination* $j = \{0, \ldots, \binom{m}{q} - 1\}$.
2. *Set the index of the starting run from* $\boldsymbol{OA}_{N,q,j}$ *to be* $r_j = j \pmod{N}$.
3. *For* $k \in \{0, 1, \ldots, N-1\}$ *and* $j \in \{0, \ldots, \binom{m}{q} - 1\}$ *add run* $r_j + k \pmod{N} + 1$ *of* $\boldsymbol{OA}_{N,q,j}$ *to* $\mathbf{S^p}_{n,m,q}$ *until the chosen run size n is met.*

With this construction we can generate optimal designs for any combination of m and q. Specifically, Table 4.4 demonstrates the process of creating a 72-run design for the case $m = 6, q = 5$. We start with the 12-run order-of-addition design in 5 components given in Schoen and Mee (2021). This is represented as $\mathbf{OA}_{12,5,01234}$ in the left panel of Table 4.4. Next, we consider the five other 5-component combinations and substitute the levels according to the permutation given in Step 3 to create each $\mathbf{OA}_{12,5,i_1 i_2 \ldots i_5}$. After concatenating these six arrays we rearrange the rows as described in Step 5 to ensure a sufficient amount of information is present in the first n rows if n is less than the size of the complete design. We do this by selecting the first run from $\mathbf{OA}_{12,5,01234}$ followed by the second run from $\mathbf{OA}_{12,5,01235}$, and so on, cycling through runs 1 through 12 and each $\mathbf{OA}_{12,5,j}$ for $j = 1, \ldots, 6$ until all runs are accounted for. The result is a 72-run D-optimal design that is the one-fifth fraction of the full design with 360 runs given in Table 4.4.

Table 4.4: 72-run, fifth-fraction D-optimal screening design, $\mathbf{S^p}_{72,6,5}$, for the PWOS model with $m = 6, q = 5$ produced by Algorithm 4.3.

Run	a_1	a_2	a_3	a_4	a_5	Run	a_1	a_2	a_3	a_4	a_5	Run	a_1	a_2	a_3	a_4	a_5
	$\mathbf{OA}_{12,5,01234}$						$\mathbf{OA}_{12,5,01235}$						$\mathbf{OA}_{12,5,01245}$				
1	0	1	2	3	4	68	0	1	2	3	5	63	0	1	2	4	5
7	0	1	4	3	2	2	0	1	5	3	2	69	0	1	5	4	2
13	0	2	4	3	1	8	0	2	5	3	1	3	0	2	5	4	1
19	0	3	4	2	1	14	0	3	5	2	1	9	0	4	5	2	1
25	4	2	3	0	1	20	5	2	3	0	1	15	5	2	4	0	1
31	3	2	4	0	1	26	3	2	5	0	1	21	4	2	5	0	1
37	4	1	3	0	2	32	5	1	3	0	2	27	5	1	4	0	2
43	3	1	4	0	2	38	3	1	5	0	2	33	4	1	5	0	2
49	4	1	2	0	3	44	5	1	2	0	3	39	5	1	2	0	4
55	3	1	2	0	4	50	3	1	2	0	5	45	4	1	2	0	5
61	2	1	4	0	3	56	2	1	5	0	3	51	2	1	5	0	4
67	2	1	3	0	4	62	2	1	3	0	5	57	2	1	4	0	5
	$\mathbf{OA}_{12,5,01345}$						$\mathbf{OA}_{12,5,02345}$						$\mathbf{OA}_{12,5,12345}$				
58	0	1	3	4	5	53	0	2	3	4	5	48	1	2	3	4	5
64	0	1	5	4	3	59	0	2	5	4	3	54	1	2	5	4	3
70	0	3	5	4	1	65	0	3	5	4	2	60	1	3	5	4	2
4	0	4	5	3	1	71	0	4	5	3	2	66	1	4	5	3	2
10	5	3	4	0	1	5	5	3	4	0	2	72	5	3	4	1	2
16	4	3	5	0	1	11	4	3	5	0	2	6	4	3	5	1	2
22	5	1	4	0	3	17	5	2	4	0	3	12	5	2	4	1	3
28	4	1	5	0	3	23	4	2	5	0	3	18	4	2	5	1	3
34	5	1	3	0	4	29	5	2	3	0	4	24	5	2	3	1	4
40	4	1	3	0	5	35	4	2	3	0	5	30	4	2	3	1	5
46	3	1	5	0	4	41	3	2	5	0	4	36	3	2	5	1	4
52	3	1	4	0	5	47	3	2	4	0	5	42	3	2	4	1	5

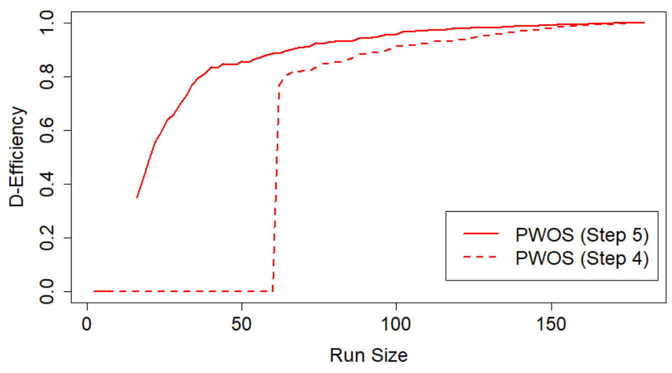
Theorem 4.6. *For* $n = 12\lceil (\binom{q}{2} + 1)/12 \rceil \binom{m}{q}$, $\mathbf{S^p}_{n,m,q}$ *is D-optimal under the PWOS model.*

Following the results of Theorems 4.5 and 4.6 we achieve significant savings by using the screening designs produced by Algorithms 4.2 and 4.3. To further demonstrate the efficiency of these designs, Figure 4.3 shows the D-efficiency of the designs $\mathbf{S^p}_{n,m,q}$ under the PWO model for several values of m and q. We study the three following situations: $(m = 6, q = 4), (m = 7, q = 4)$, and $(m = 8, q = 5)$.

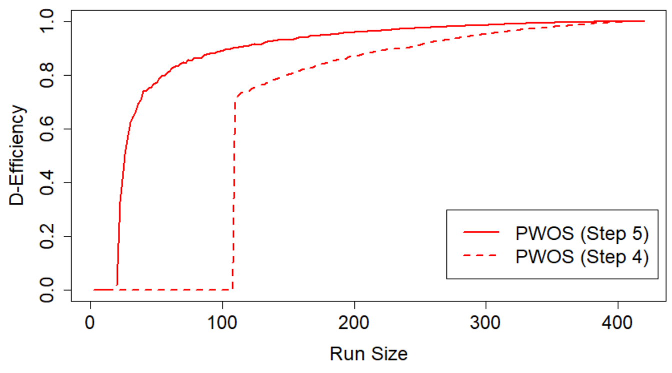
It is important to note that for $q > 3$ the designs generated by Algorithm 4.3 have efficiency 0 under the CPS model due to the fact that at least one component is absent from at least one column of every OofA-OA. Given this natural limitation, we can only consider the efficiency of our designs under the CPS model when $q = 3$. For $q \geq 4$ we instead demonstrate the value of the row rearrangement in Step 5 of Algorithm 4.2 by comparing the final n run design to one that results from taking the first n runs of $\mathbf{S^p}_{m,q}$ in Step 4. We can see from Figure 4.3 that the row rearrangement in Step 5 greatly improves the efficiency of the resulting design, especially for small run sizes. In general, we find that once the run size is sufficiently large enough to estimate the PWOS model, the design after row rearrangement is quite efficient, with D-efficiency greater than 0.80. From these results we conclude that between the three construction methods, we can generate designs that are efficient, parsimonious, and in some cases optimal for estimating one or both of the screening models. Our goal is now to see how researchers can use these designs in practice through a simulated order-of-addition screening experiment.

4.3 Order-of-Addition Screening Experiments in Practice

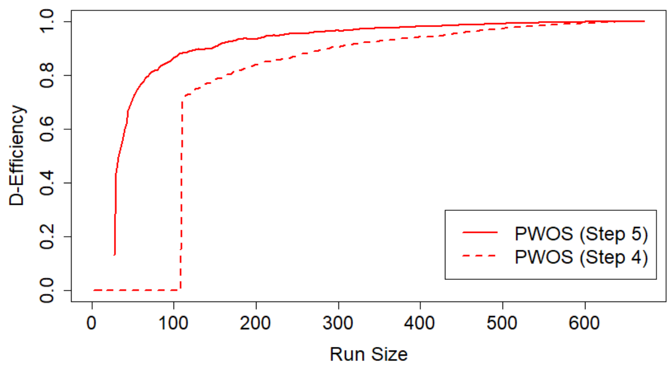
In order to demonstrate the value of our proposed designs in practice, we consider a collection of job scheduling problems of varying complexity. Job scheduling problems are a class of well known NP-hard problems that have been critically studied in operations research (Garey



(a)



(b)



(c)

Figure 4.3: D-efficiency of designs $\mathbf{S^P}_{n,m,q}$ relative to the full design $\mathbf{S}_{m,q}$ generated for variable run sizes with (a) $m = 6, q = 4$, (b) $m = 7, q = 4$, and (c) $m = 8, q = 5$.

et al., 1976). Specifically, we borrow the setup from Zhao et al. (2020a) in which we consider a single machine which is tasked with sequentially processing jobs, each of which takes some fixed time to complete and requires some fixed costs to perform. Our goal is to complete q of the m available jobs in a specific order such that a given penalty is minimized, indicating that the sequence is in some sense the most efficient. The penalty we choose is a quadratic penalty given by $y = \sum_{i=1}^{q} c_i (\sum_{j=1}^{i} t_j)^2$, where t_j and c_i are the processing time of job j and the cost of job i, respectively (Townsend, 1978). Considering this task, we demonstrate the value that our proposed order-of-addition screening designs provide in capturing the relationship between the job sequence and the endpoint penalty as well as in cheaply and efficiently uncovering the optimal job sequence.

4.3.1 Screening Experiments for Modeling Job Scheduling Problems

To evaluate how the proposed designs can be used to model the relationship between the component selection and sequence, we consider two situations ($m = 4$, $q = 3$ and $m = 6$, $q = 4$) based on the problems considered in Zhao et al. (2020a). The scheduling matrices for these problems are given in Table 4.5.

Table 4.5: Job scheduling matrices for $m = 4$ and $m = 6$ problems from Zhao et al. (2020a).

job	0	1	2	3
t	1	5	5.5	7
c	7	3	2	6

job	0	1	2	3	4	5
t	5	3	2	10	3	1
c	4	5	3	1	6	2

For the four-job problem we consider two designs which we have found are D-optimal for the respective models. We use the 12-run design from Algorithm 4.2 for training the PWOS model and the 12-run design from Algorithm 4.1 for training the CPS model. The PWOS model gives average performance ($R^2 = 0.65$, AIC = 188.83) while the CPS model yields high performance ($R^2 = 0.96$, AIC = 116.40). To test the predictive performance of each model we predict across all 24 sequences and measure the observed versus predicted correlation.

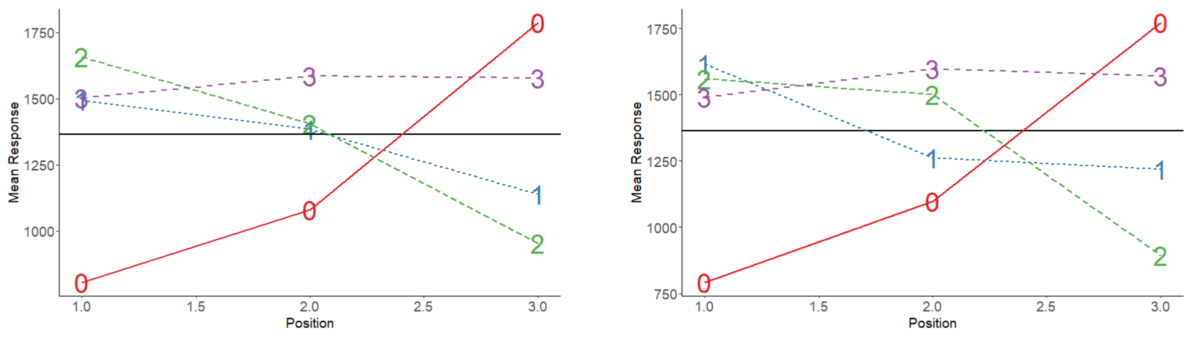


Figure 4.4: Component-position effects plots for the true data (left) and CPS predictions (right) for the job scheduling problem with $m = 4$, $q = 3$. The CPS model is trained on a 12-run D-optimal design and predictions across all 24 runs are used to generate the plot.

The CPS model outperforms the PWOS model with a correlation of 0.91 compared to 0.54 for the PWOS model.

We can also visually interpret these models and draw some preliminary conclusions from the component-position effects plots in Figure 4.4. For a detailed explanation of how these plots are generated see the discussion of Figure 3.1. This figure shows the component-position effects plot for the true data and for the CPS model predictions. Each plot is constructed from 24 observations. The left plot is built from the 24 true values and acts as a benchmark while the right plot is built from the 24 predictions of the fitted CPS model. We omit the plot of the PWOS model predictions due to its average performance; however, we note that the overall interpretations are similar. Interpreting these plots we see that the CPS model is adept at picking up the trends of the true data. If our aim is to minimize the penalty, then we should process job 0 first and job 2 third. The CPS model then indicates that we should process job 1 second and not process job 3 at all. These interpretations align with the plot of the true data and the sequence that obtains the true minimum penalty, $\{0, 1, 2\}$. This visual analysis of course is only a first attempt at how researchers may use the proposed designs in concert with the screening models to draw substantive conclusions. Further study may be

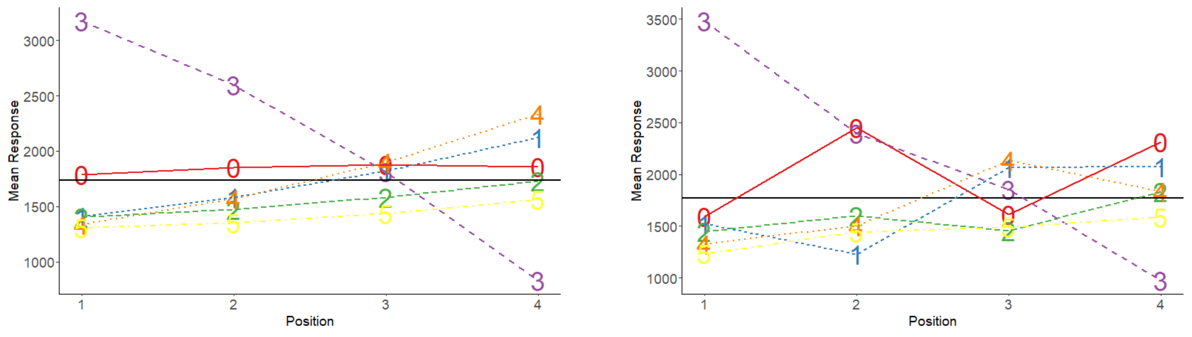


Figure 4.5: Component-position effects plots for the true data (left) and the CPS predictions (right) for the job scheduling problem with $m = 6$, $q = 4$. The CPS model is trained on a 30-run design and predictions across all 360 runs are used to generate the plot.

required to uncover and interpret any potential interactions between the components.

Considering the six-job problem, we again start from two designs inspired by the proposed algorithms. For the PWOS model this is the 30-run D-optimal design from Algorithm 4.3. Since this problem has a non-prime number of components, we cannot directly apply a design generated by Algorithm 4.1 for the CPS model. For this reason we instead replace $\mathbf{F}_{n,m}$ in Step 1 of the algorithm with the first 30 runs of the six-component COA from Huang (2021) and use the resulting design. While this design is not D-optimal under the CPS model, it achieves high efficiency relative to the full design.

Training the models on their respective designs, we find that the CPS model demonstrates good performance, having $R^2 = 0.98, AIC = 431.60$ while the PWOS model's performance is average, having $R^2 = 0.80, AIC = 491.17$. Considering the predictive performance, we predict across all 360 sequences and see similar trends to the previous problem, with the CPS and PWOS models having observed versus predicted correlations of 0.88 and 0.47, respectively. From these results we can see that, even as the number of components grows, the CPS model, in concert with efficient designs, is capable of capturing the relationships between the job sequence and the endpoint penalty.

Finally, we can again visually interpret the output of the models. For this, we can use the 360 predicted values from the CPS model to generate the component-position effects plot. This plot, along with the plot built from the 360 true values are given in Figure 4.5 and can be interpreted in a similar manner to the previous problem. We again omit the predictions from the PWOS model due to its poorer performance and to conserve space. Due to the increased complexity of this problem, we can only deduce general conclusions from the plot of the CPS predictions. For example, the model indicates that job 3 should be processed last while job 0 is perhaps too costly to process at all. Furthermore, the plot of the true data does not show a clear distinction between the other jobs when considering the remaining positions of the sequence, while the model indicates that jobs 1 and 5 should be processed early in the sequence. These discrepancies could be due to the fact that the 30-run design inspired by Huang (2021) is not a COA and is only near-optimal. Further study of this construction is necessary to improve the performance of designs for the CPS model with non-prime m.

We again note that these conclusions, while quite general, would require further study to understand the nuances in the underlying relationship; however, from the analysis of both of these problems we have seen that our proposed designs can be quite successful in advancing a critical application of order-of-addition experiments.

4.3.2 Sequential Screening Experiments for Optimizing Job Scheduling Performance

While we have seen that the proposed designs lead to stable, interpretable models, we have so far focused on single-shot designs where the entire budget of the experiment is used at once. However, in some cases it may be of interest to run a sequential experiment in which the goal is to find the best sequence as quickly as possible by first obtaining the responses from an initial design and then adding points to the design one at a time. We now aim to demonstrate the benefit of the proposed designs for this problem. For this procedure we only consider the CPS model and designs generated from Algorithm 4.1. We further assume that we have the full job sequencing dataset with a known global minimum. Within this setup, we consider the benefit of choosing the designs we have constructed as the initial design over a random one as follows:

1. First collect the response from the design $\mathbf{S^c}_{n,m,q}$ obtained from Algorithm 4.1 with $n = q(m-1)+5$ and record the minimum response.
2. Next fit the CPS model to the data and calculate the Expected Improvement for all remaining sequences in the pool (Jones et al., 1998). The Expected Improvement for a given sequence $\boldsymbol{x}$ can be calculated as
$$EI(\boldsymbol{x}) = (y^* - \hat{y}(\boldsymbol{x}))\Phi\Big(\frac{y^* - \hat{y}(\boldsymbol{x})}{\hat{\sigma}}\Big) + \hat{\sigma}\phi\Big(\frac{y^* - \hat{y}(\boldsymbol{x})}{\hat{\sigma}}\Big),$$
where y^* is the minimum value observed so far, ϕ and Φ are the PDF and CDF of the normal distribution, respectively, and $\hat{\sigma}$ is the estimate of the standard error of the prediction.
3. Add the design point with the highest EI value to the design and calculate its response.
4. Fit the CPS model again with the updated design and record the minimum response found so far.
5. Repeat 2-4 until the maximum number of iterations is reached.

In the first step of our sequential experiment framework, note that the number of initial runs is set to $n = q(m-1)+5$. This is done in order to keep the number of runs in the experiment low while ensuring sufficient degrees of freedom for estimation. To compare this approach to a random initial design we repeat the process above with 100 random designs and average the resulting curves. We consider two different job scheduling problems with $m = 7$ and $m = 11$. Table 4.6 gives the values of t and c for each job for the two problems. These values were uniformly sampled for each problem from $\{0, 1, \dots, 3m\}$.

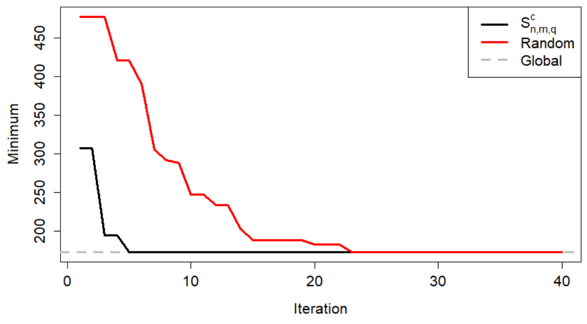
Table 4.6: Job scheduling matrices for active learning sequential experiments with $m = 7$ (left) and $m = 11$ (right).

Job	0	1	2	3	4	5	6
t	6	1	11	1	2	21	2
c	7	19	3	4	10	20	18

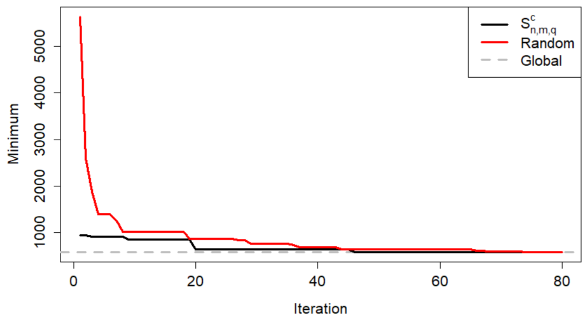
Job	0	1	2	3	4	5	6	7	8	9	10
t	6	27	13	11	20	20	5	10	20	21	17
c	17	18	19	29	28	4	24	30	10	8	1

We first consider the problem with $m = 7$ jobs. We generate the designs $\mathbf{S^c}_{n,7,q}$, where q takes the values $3, \ldots, 6$ and $n = q(m-1) + 5$. Applying each design in the sequential experimentation outlined above, the results for each value of q are presented in Figure 4.6. In these plots the dashed gray line represents the true global minimum. Each point on the red curve represents the average minimum value obtained at the specified iteration across 100 random initial designs. From these plots we can see that initializing the experiment with the design $\mathbf{S^c}_{n,7,q}$ leads to convergence that is at least as fast as when starting from an average random design. In fact, for most cases we see that convergence is much faster under the proposed design. Specifically in the cases of $q = 3$ and $q = 6$ we see that convergence to the global minimum occurs with less than half the number of iterations required for the random design. After accounting for the size of the initial designs this translates to roughly a 40% and 30% reduction in the total budget required for the experiments, respectively.

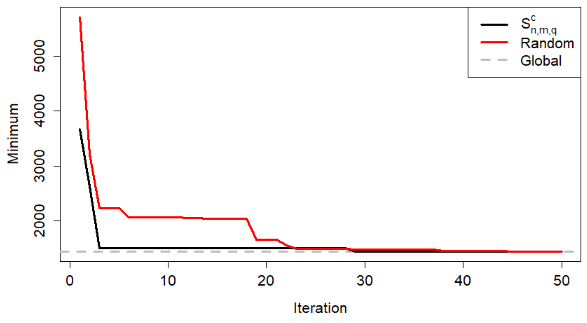
Considering the more difficult problem with $m = 11$ jobs, we again start from the $\mathbf{S^c}_{n,11,q}$ designs for $q = 3, \ldots, 6$. The results of running the sequential experiment for each of these initial designs and 100 random designs are given in 4.7. The true global minimum is again given as a dashed gray line in each figure. As with the simpler 7 job problem, we see that for all situations the convergence of the algorithm when starting from the proposed design is at least as fast as when starting from a random design. In fact, we see that in the four cases we have considered the convergence is actually faster under our design, and in one case, $q = 6$, the average random design is unable to converge with 100 additional runs. For the other three cases, this faster convergence leads to a substantial reduction in the total budget of



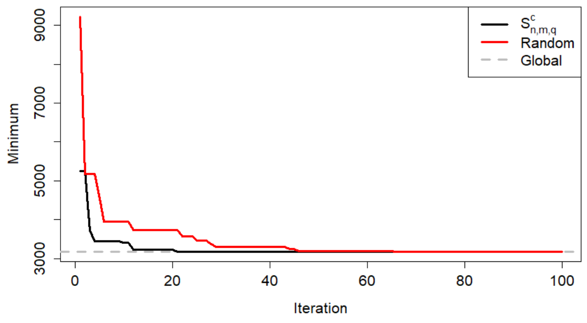
$q = 3$



$q = 4$



$q = 5$



$q = 6$

Figure 4.6: Convergence of the sequential job scheduling experiment with $m = 7$ and variable q with different initial designs.

the experiment, between roughly 9% and 27%.

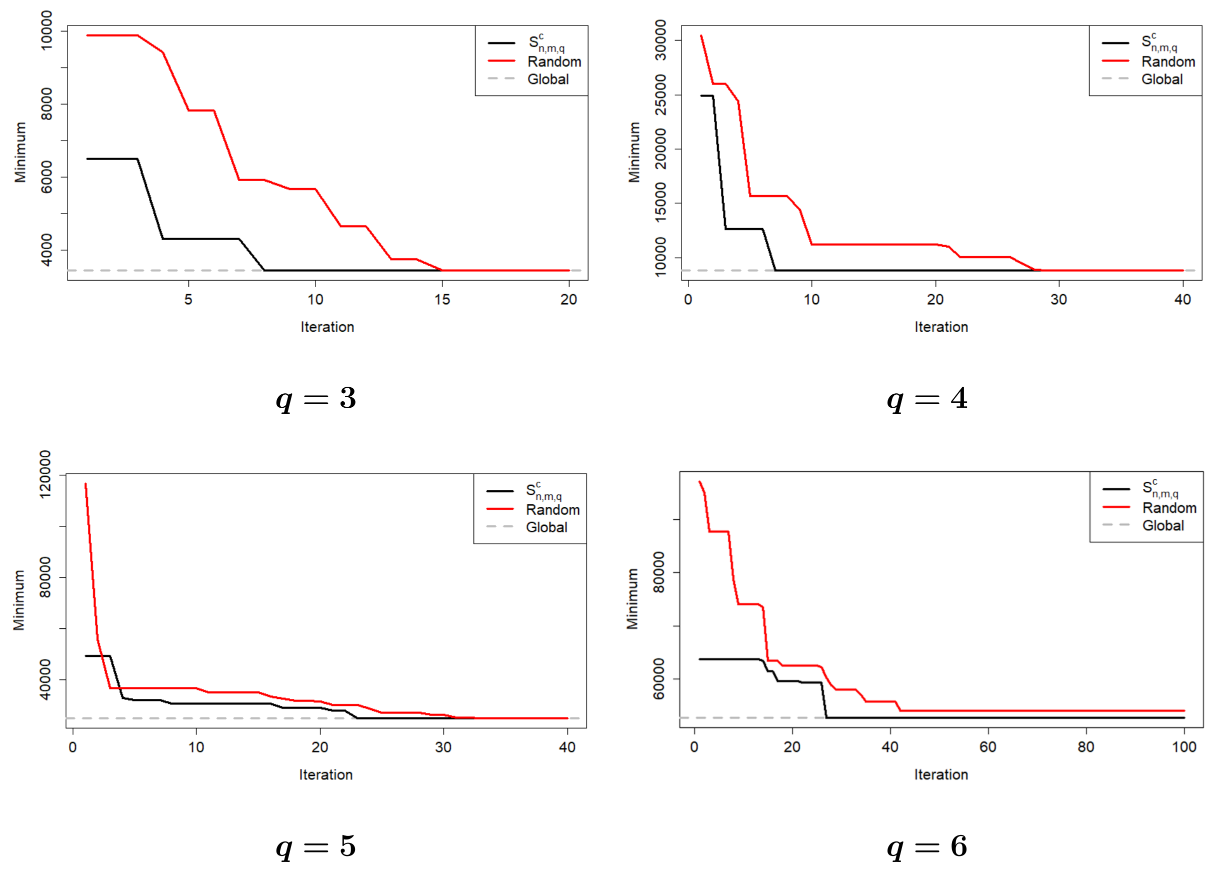


Figure 4.7: Convergence of the sequential job scheduling experiment with $m = 11$ and variable q with different initial designs.

We can see from these two examples that starting from a design generated by Algorithm 4.1 often leads to much faster convergence than a random design, even as the total number of components or the value of q increase. In some cases we have seen that the total budget required for the experiment when starting from this design is only a fraction of that under the random design. These examples have demonstrated the potential value of our proposed designs. We could repeat this procedure with the optimal designs generated for the PWOS model by Algorithms 4.2 and 4.3, but as we have seen in the previous section, the CPS model seems more appropriate for the job sequencing application, so we postpone this study until there is an appropriate application. However, in all of the problems considered here we also notice that the minimum achieved by $\mathbf{S^c}_{n,m,q}$ before any additional points are added (iteration 0) tends to be much smaller than that of the average random design. This is likely due to improved space-filling properties of the design $\mathbf{S^c}_{n,m,q}$, an observation we explore further in Chapter 5.

4.4 Chapter Summary

In this chapter we have studied the problem of designing order-of-addition experiments in cases where the number of components of interest m outnumbers the number of available positions in each sequence q. Like the standard order-of-addition problem, the full design that includes all possible sequences grows too quickly to be appropriate in most cases, necessitating smarter, simpler designs. However, this problem differs from the standard one in that our goal is not only to understand the relationship between the component sequence and the response, but also to screen the components to find the q that have the largest effect. While only small modifications of the standard order-of-addition models are necessary to accommodate this new component screening problem, new designs are required in order to keep costs low.

We have proposed three constructions for order-of-addition screening designs and have shown that each guarantees D-optimality for certain run sizes. For the CPS model Algorithm 4.1 generates D-optimal designs by leveraging the balance properties of the $\mathbf{F}_{n,m}$ designs generated in Chapter 3 for the standard order-of-addition problem. On the other hand, Algorithms 4.2 and 4.3 generate D-optimal for different choices of m and q under the PWOS model. Algorithm 4.2 considers the case that m is even and $q = 3$. For this case the algorithm generates highly efficient designs including a D-optimal half-fraction design. Algorithm 4.3 on the other hand covers the remaining cases where $m > 4$ and $3 < q < m$. Using the OofA-OA designs developed for the standard order-of-addition problem under the PWO model, this construction generates D-optimal designs with only a small fraction of the number of runs in the full design, with the savings growing larger as m increases. Collectively these three constructions fill an important gap in the order-of-addition literature.

To demonstrate the value of our methods, we have studied several synthetic job sequencing data sets with varying m and q. We have found that for simpler problems, training each model on the proper optimal design from our constructions results in a suitable fit with stable coefficients and strong predictive performance. Furthermore, to showcase the cost-saving potential of our designs, we have considered sequential job scheduling experiments in an active learning framework. The results of this study indicate that the proposed designs lead to much faster convergence of the algorithm to the true optimal job sequence when compared to a random initial design, even as m increases. By providing efficient designs for the study of order-of-addition screening experiments it is our hope that researchers will soon find many other applications of these results and continue to explore new approaches for modeling this problem.

CHAPTER 5

Metaheuristic Solutions to Order-of-Addition Design Problems

In the previous chapters we have solved several critical problems in the design and analysis of order-of-addition experiments. However, many important problems remain unsolved and approaches for solving them with theoretical guarantees are not always available or easy to obtain. For these problems we resort to nature-inspired, metaheuristic approaches that have become popular for solving optimization problems, especially in engineering and computer science (Whitacre, 2011a,b). Some key reasons for their popularity are their speed, simplicity, flexibility and ease of implementation. Additional compelling reasons for their widespread use are: (a) a lack of technical assumptions, so they can be applied to solve problems where, for example, the objective function is non-differentiable, multi-modal, multi-objective or high dimensional, and (b) they tend to provide exact or approximate solutions of high quality and if necessary, they can be improved either by better choice of the values for the tuning parameters or by hybridization with one of more algorithm of the same or different types; see Blum and Roli (2001).

There are two primary classes of nature-inspired metaheuristic algorithms: swarm intelligence and evolutionary algorithms. Both classes involve a set of agents exploring the space. In swarm intelligence algorithms the agents share information as necessary. The class of swarm intelligence algorithms has become quite large recently, but one of the key representatives of the class is Particle Swarm Optimization (PSO), which is used frequently in practice with the number of publications applying PSO growing exponentially since its creation (Kennedy and Eberhart, 1995; Poli, 2008). In evolutionary algorithms, each agent is made up of chromosomes and gene, in an adaptive, survival-of-the-fittest battle to be part of the best generation. Differential Evolution (DE) is a representative algorithm from this class that has found widespread usage for modern applications (Storn and Price, 1997; Chakraborty, 2008).

Recently, these algorithms have been applied to solve challenging optimal design problems. For example, Wong et al. (2015) used PSO to construct D-optimal designs for a variety of mixture models, Chen et al. (2015) tackled maximin and standardized optimal design problems, and Phoa et al. (2016), applied PSO to find optimal supersaturated designs. DE has similarly been used to locate optimal designs. For example, Paredes-García and Castaño-Tostado (2017) compared DE-derived designs for well-known models with those currently available, Feoktistov et al. (2017) used DE to find sparse split-plot designs, and Xu et al. (2019) found D-optimal designs for logistic models. Motivated by these recent successes, we want to test the ability of these algorithms and their variants to find quality designs for open order-of-addition problems.

In this chapter, we are the first to consider the efficacy of DE, PSO, and several of their most popular variants for finding solutions to two critical order-of-addition problems. First we introduce the key concepts of both algorithms and various alterations that have been made to improve their performance on complex optimization problems. Next we outline two important order-of-addition problems: A-optimal order-of-addition designs for position-based models and space-filling order-of-addition designs. Finding quality designs for these problems would help to advance the field and address relevant applications, but analytic solutions are too difficult to derive.

Section 5.2 presents our proposed optimality criteria and applications to find two types of order-of-addition designs. Section 5.3 compares results using the DE and PSO algorithms, along with 3 of each of their variants. We find that across many situations the algorithms are able to produce order-of-addition designs with dramatically improved optimality and space-filling properties, with Jingqiao's Adaptive Differential Evolution (JADE) and Quantum Particle Swarm Optimization (QPSO) generating particularly outstanding results for several cases. The designs generated by these algorithms outperform the existing ones developed in the previous chapters in terms of our chosen criteria and a future study of their properties could aid in the development of general constructions.

5.1 Differential Evolution, Particle Swarm Optimization, and Their Popular Variants

We first review two popular metaheuristic algorithms: DE and PSO. These algorithms have achieved widespread support from the optimization community for their simplicity, efficiency and high performance. We outline the key details of each algorithm including their nature-inspired structure, considerations for tuning their parameters, and their variants that have found widespread usage. As expected, variants generally perform better in some ways than the original version or are particularly suited for tackling a specific type of optimization problems.

Without loss of generality, assume that the goal is to minimize a real-valued fitness function with v variables $h : \mathbb{R}^v \to \mathbb{R}$ by finding $\boldsymbol{x}^* \in \mathbb{R}^v$ such that $h(\boldsymbol{x}^*) \leq h(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^v$. The search space of candidate solutions is defined by the limits of each of the v variables and constitutes the landscape of the fitness function. These limits may be specified naturally by the application or selected by the experimenter. In our specific case of order-of-addition experiments the variables may be the individual permutations or columns of the component matrix $\boldsymbol{A}$ that make up the candidate design. Examples of fitness functions of interest here are the negative log of the determinant or the trace of the information matrix.

5.1.1 Differential Evolution

In the class of metaheuristic algorithms based on evolutionary mechanics, DE was proposed as a general purpose evolutionary algorithm as a means of quickly optimizing functions that are not necessarily differentiable or continuous. It has since found widespread application in many areas (Chakraborty, 2008). The goal of evolutionary algorithms is to simulate survival-of-the-fittest dynamics to gradually converge to the global optimum. DE does this by treating each candidate or agent $\boldsymbol{x}$ as a chromosome made up of v genes and implementing mutation and crossover procedures to allow beneficial genes to persist into future generations (Storn, 1996).

The first step in this process is to choose the number of candidate solutions per generation, np, also known as the population size. Each solution is represented by a vector of length v, so each generation of candidate solutions has dimension $v \times np$. In addition to the population size, there are two parameters, a mutation weighting factor wf and crossover constant cr, to be selected. The final step in the initialization process is to specify a stopping rule or condition. For example, a maximum number of iterations or a target value may be set. Once these preliminary steps are complete, the five steps for the standard DE algorithm are as follows:

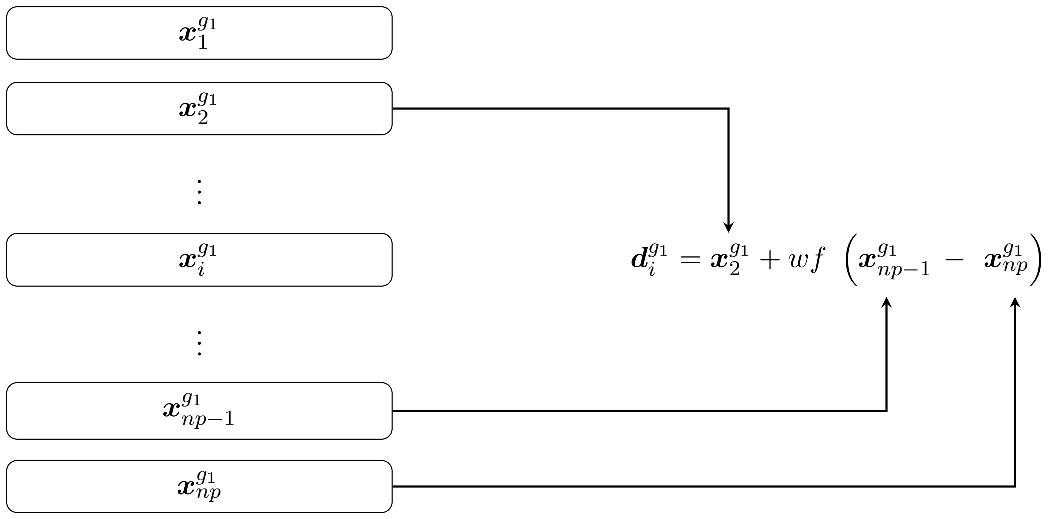
1. **Genetic Representation:** The initial population must be of size $np > 4$ to ensure that there is enough genetic diversity. Each agent is represented by a vector of length v and labeled as $\boldsymbol{x}_1^{g_1}$,$\boldsymbol{x}_2^{g_1}, \ldots, \boldsymbol{x}_{np}^{g_1}$. The initial value for each entry in each agent is randomly chosen over the interval specified for the particular variable.

2. **Mutation:** Mutation expands the search space of DE. For each agent $\boldsymbol{x}_i^{g_1}$, mutation produces a donor vector $\boldsymbol{d}_i^{g_1}$ by adding the weighted difference of two agents to a third, all randomly chosen and distinct from the target. For this process, a weighting factor wf is chosen on the interval $[0, 2]$. Thus, with this constant, np vectors are created

according to

$$\boldsymbol{d}_i^{g_1} = \boldsymbol{x}_{r_0}^{g_1} + wf(\boldsymbol{x}_{r_1}^{g_1} - \boldsymbol{x}_{r_2}^{g_1}), \tag{5.1.1}$$

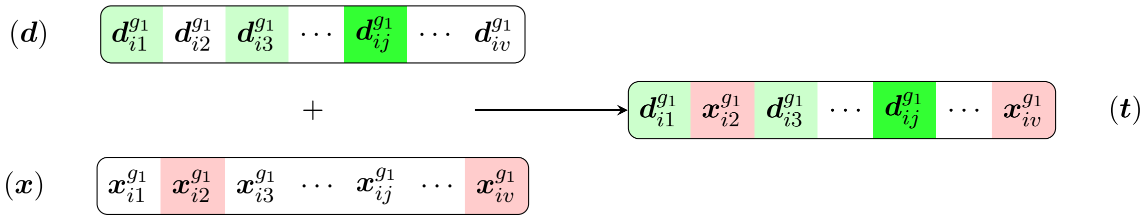
where $r_0 \neq r_1 \neq r_2 \neq i$. Depending on the bounds and choice of f this process may lead to values that fall outside of the acceptable region for each dimension. There are many strategies in the general optimization literature for dealing with this problem, but we do not explicitly discuss them.



$\boldsymbol{x}_1^{g_1}$
$\boldsymbol{x}_2^{g_1}$
$\boldsymbol{x}_i^{g_1}$
$\boldsymbol{x}_{np-1}^{g_1}$
$\boldsymbol{x}_{np}^{g_1}$
$\boldsymbol{d}_i^{g_1} = \boldsymbol{x}_2^{g_1} + wf\left(\boldsymbol{x}_{np-1}^{g_1} - \boldsymbol{x}_{np}^{g_1}\right)$

Figure 5.1: An illustration of mutation for a single agent in the DE algorithm where a donor vector $\boldsymbol{d}_i^{g_1}$ is created by blending 3 randomly drawn agents. In this case $\boldsymbol{x}_2^{g_1}$, $\boldsymbol{x}_{np-1}^{g_1}$ and $\boldsymbol{x}_{np}^{g_1}$ were chosen.

3. **Crossover:** Crossover blends the current generation of agents with the population of donor vectors in order to form candidates for the next generation known as trial vectors. This technique requires the crossover constant cr, chosen from $[0, 1]$. For each i from 1 to np, one of the v elements of $\boldsymbol{d}_i^{g_1}$ is randomly selected to directly enter the trial vector $\boldsymbol{t}_i^{g_1}$. In this way one variable is forced to change so that each $\boldsymbol{t}_i^{g_1}$ will certainly differ from its original target vector. Next, with probability cr more elements are taken from $\boldsymbol{d}_i^{g_1}$ and placed in the trial vector. Whichever variables do not take their



$(\boldsymbol{d})$
$\boldsymbol{d}_{i1}^{g_1}$ $\boldsymbol{d}_{i2}^{g_1}$ $\boldsymbol{d}_{i3}^{g_1}$ $\cdots$ $\boldsymbol{d}_{ij}^{g_1}$ $\cdots$ $\boldsymbol{d}_{iv}^{g_1}$
$+$
$(\boldsymbol{x})$
$\boldsymbol{x}_{i1}^{g_1}$ $\boldsymbol{x}_{i2}^{g_1}$ $\boldsymbol{x}_{i3}^{g_1}$ $\cdots$ $\boldsymbol{x}_{ij}^{g_1}$ $\cdots$ $\boldsymbol{x}_{iv}^{g_1}$
$\boldsymbol{d}_{i1}^{g_1}$ $\boldsymbol{x}_{i2}^{g_1}$ $\boldsymbol{d}_{i3}^{g_1}$ $\cdots$ $\boldsymbol{d}_{ij}^{g_1}$ $\cdots$ $\boldsymbol{x}_{iv}^{g_1}$
$(\boldsymbol{t})$

Figure 5.2: An illustration of the crossover procedure for a single agent in the Differential Evolution algorithm. A trial vector $\boldsymbol{t}_i^{g_1}$ is created by combining $\boldsymbol{x}_i^{g_1}$ and $\boldsymbol{d}_i^{g_1}$. The light green elements of $\boldsymbol{t}_i^{g_1}$ come from the donor vector $\boldsymbol{d}_i^{g_1}$ and the light red elements come from the target vector $\boldsymbol{x}_i^{g_1}$. The jth element of $\boldsymbol{t}_i^{g_1}$ is marked in dark green as it comes from $\boldsymbol{d}_i^{g_1}$ with probability one.

value from the donor vector inherit their original value from $\boldsymbol{x}_i^{g_1}$. Assuming variable j is randomly chosen to take its value from the trial this process is driven by the following equation:

$$\boldsymbol{t}_i^{g_1} = \begin{cases} \boldsymbol{d}_i^{g_1} & & \text{for } i = j, \\ \boldsymbol{d}_i^{g_1} & \text{with probability } \; CR, & \text{for } i \neq j, \\ \boldsymbol{x}_i^{g_1} & \text{with probability } \; 1 - CR, & \text{for } i \neq j. \end{cases} \tag{5.1.2}$$

4. **Selection:** Selection creates the next generation of agents by comparing each target vector to its respective trial vector. Whichever is measured to be the most fit using h becomes $\boldsymbol{x}_i^{g_2}$. In the case of minimization this process is given by

$$\boldsymbol{x}_i^{g_2} = \begin{cases} \boldsymbol{t}_i^{g_1} & \text{if} \quad h(\boldsymbol{t}_i^{g_1}) < h(\boldsymbol{x}_i^{g_1}), \\ \boldsymbol{x}_i^{g_1} & \text{otherwise.} \end{cases} \tag{5.1.3}$$

5. **Repeat:** Repeat steps 2 through 4 over many generations until the specified stopping condition is satisfied.

The standard DE algorithm defined above has only 3 tuning parameters, np, wf, and cr. The selection of these values is critical to the success of the algorithm. This process is in part driven by the application of interest and prior knowledge of the function landscape; however, there are several rules of thumb we can apply to most prolems . For example, it is recommended that np be at least 10 times the number of variables v to ensure sufficient diversity. Under the standard algorithm, empirical evidence has indicated that wf should be set between 0.6 and 0.9 for general optimization problem (Zaharie, 2002). Finally, the creators of DE suggests setting $cr \in \{[0, 0.3] \cup [0.8, 1]\}$, choosing from the smaller range for separable problems and the larger range otherwise. One approach for reducing the effect of choosing a poor value of wf or cr is dithering and jittering (Dawar and Ludwig, 2014). Dithering involves choosing new parameter values for each generation while jittering chooses a new value for each variable in the optimization by adding a small perturbation γ to the value for that generation. These approaches can be implemented simultaneously and this is one extension of the basic algorithm that we will consider in our analysis. Price et al. (2005) provides an investigation into how the selection of individual parameters can affect performance of the standard DE.

5.1.2 Particle Swarm Optimization

Since its creation in 1995 PSO has exploded in popularity. It has been applied to tackle a diverse set of optmization problems in healthcare, investment banking, engineering and many others in the last decade (Poli, 2008; Elbes et al., 2019). It is the premier algorithm in the class of swarm intelligence metaheuristics and operates by mimicking the process of flocking birds or schooling fish, allowing for the transfer of information between the flock or agents $\boldsymbol{x}$. This structure allows the particles to converge to the global optimum while also sufficiently exploring the space (Chen et al., 2015).

Akin to DE, to initialize PSO we first need to decide how many agents will make up the population np. Additionally, there are several other parameters that need to be initialized, including an inertia weight τ, two independent random vectors β_1 and β_2, a cognitive learning factor cl and a social learning factor sl. Given the fitness function, the search space and a user-selected set of values for the tuning parameters, the standard PSO algorithm has the following steps:

1. **Swarm Generation:** The initial positions for the agents $(\boldsymbol{x}_1^{t=0}, \boldsymbol{x}_2^{t=0}, \ldots, \boldsymbol{x}_{np}^{t=0})$ are randomly generated over the specified interval of dimension d. An initial velocity $\boldsymbol{v}_i^{t=0}$ is also generated for each agent. This velocity is often $\mathbf{0}_d$ or randomly drawn from $[0,1]^v$ for all agents. By definition, this vector gives the magnitude and direction of the agents movements between iterations. For each agent, its personal best position is set to $\boldsymbol{p}_i = \boldsymbol{x}_i^{t=0}$ and the global best position so far is set to $\boldsymbol{p}_g = \arg\min_i h(\boldsymbol{x}_i^{t=0})$ by calculating the fitness function of each agent's position.

2. **Particle Velocity Calculation:** As each agent moves through the space its velocity changes. The updated velocity can be calculated using the inertia τ, cognitive and social learning factors cl, sl and random vectors $\beta_1, \beta_2 \in [0,1]^v$. The relationship between these parameters and the next velocity step for each agent $\boldsymbol{x}_i^t$ is given by

$$\boldsymbol{v}_i^{t+1} = \tau \boldsymbol{v}_i^t + cl\beta_1 \odot (\boldsymbol{p}_i - x_i^t) + sl\beta_2 \odot (\boldsymbol{p}_g - x_i^t), \tag{5.1.4}$$

   where $\odot$ represents hadamard multiplication. The inertia factor τ can control the effect of the current velocity and allow the agent to explore parts of the space that are closer to the global or personal best positions.

3. **Particle Position Update:** Using the calculated velocity, the agent explores the space in the direction of its velocity $\boldsymbol{v}_i^{t+1}$ and its position is updated accordingly as follows:

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1}. \tag{5.1.5}$$

   As with DE's mutation, there is a chance that this procedure will produce positions that are outside of the search region. Many techniques exists for dealing with this problem, but we will not discuss them here and simply use the default clipping mechanism of the original PSO.

4. **Global and Personal Best Refresh:** Following the update of each agent's position, the agent's best position so far $\boldsymbol{p}_i$ is updated as well as the global best position $\boldsymbol{p}_g$ by calculating the fitness function for each agent's new position. In the t^{th} iteration, the personal best of an agent is the best position it has attained up to time t, and the global best position is the best position the flock has attained up to time t. Here best means the most optimal value of the fitness function.

5. **Repeat:** Repeat steps 2 through 4 over many iterations until the specified stopping criterion is satisfied.

Typically the 6 parameters $np, \tau, cl, sl, \beta_1$, and β_2 are chosen randomly over a uniform interval, often $[0, 1]$ for all parameters besides np, but there has been much research into more complex approaches for setting their values. Of course, the application of interest and prior knowledge of the search space can also be useful in initializing these parameters. As a rule of thumb the number of agents is generally between 20 and 50, with fewer agents required for simpler problems (He et al., 2016). Furthermore, Shi and Eberhart (1998) suggested that choosing $\tau \in [0.9, 1.2]$ gives good performance without requiring too many iterations; however, it is also common to linearly decrease the inertia over the course of the algorithm as the agents get closer to the optimum. The cognitive and social learning factors are known to have a large impact on the convergence properties of the algorithm, and while there is no clear guidance on the best values to choose, Eberhart and Shi (2000) found that a value between 1.496 and 2 works well for general problems. While these rules provide some assistance in choosing appropriate parameter values, it is clear that the number of parameters and their sensitivities will be an important consideration in the evaluation of their performance on order-of-addition design problems. For a detailed look at the process of tuning the parameters in metaheuristic algorithms, see Birattari (2009).

5.1.3 Popular Variants

A common feature of metaheuristic algorithms is that there are many variants of them. These are modified versions of the original algorithm that enhance the algorithm's performance in different ways. For example, they propose ways to bring rogue agents back to the search space, strategies for faster convergence or attempts to escape local optima. DE and PSO are no exceptions to this, and in the years since their creation there have been many advanced versions of each that have been shown to produce higher quality results for general problems. In our search for efficient order-of-addition designs, we also consider some of the most popular variants. Of course, there are far too many variants to consider here, so we choose some of the most impactful and widely-used variants to briefly review and compare.

For each algorithm, we have already discussed a few simple changes that are commonly used in practice. For DE, dithering and jittering help to reduce the effect of choosing poor values for wf and cr, while for PSO a linear schedule is often used to reduce the value of τ across generations to allow for finer convergence. In addition to these simple changes, there are more complex revisions of each algorithm in our comparison. For DE, this often involves exploring other strategies for mutation and crossover with adaptive parameter search. One example is Jingqiao's Adaptive Differential Evolution (JADE) (Zhang and Sanderson, 2009). For this algorithm a new set of parameters is generated for every agent in every generation, and a new mutation strategy is implemented in Step 2, in which the donor vector $\boldsymbol{d}_i^{g_1}$ is generated by the following equation:

$$\boldsymbol{d}_i^{g_1} = \boldsymbol{x}_i^{g_1} + wf_i(\boldsymbol{x}_{best,g_1}^{p} - \boldsymbol{x}_i^{g_1}) + wf_i(\boldsymbol{x}_{r_0}^{g_1} - \boldsymbol{x}_{r_1}^{g_1}), \tag{5.1.6}$$

where $\boldsymbol{x}_{best,g_1}^{p}$ is randomly chosen as one of the top $100p\%$ current best agents, with $p \in (0, 1]$ being a new parameter, and wf_i is the value of the weighting factor chosen for agent i in the particular generation. Each new value of wf_i is initially chosen from a Cauchy distribution with mean 0.5 and scale parameter 0.1, truncated at 1. At the end of each generation the distribution is updated based on which wf values led to trial vectors that were accepted into the next generation over the target vector. Similarly, the individual values of cr_i are randomly drawn from $N(0.5, 0.1)$, truncated at 1, and the mean is updated after each generation based on which cr_i values were successful. In each of these cases a parameter $c \in [0, 1]$ controls how much weight is given to the successful values when shifting the location parameter.

Another popular variant of DE is the Self-Adaptive Differential Evolution (SADE) (Qin et al., 2009). It differs from DE by allowing two different mutation strategies. For each agent in each generation, the first strategy is chosen with probability p_1, and the second is chosen with probability $p_2 = 1 - p_1$. These probabilities are updated after a specified number of generations g_{update} based on the ratio of successful and unsuccessful mutations. The two mutation procedures are the standard one given in (5.1.1) and one that incorporates the best value so far. Specifically, the second mutation is given by

$$\boldsymbol{d}_i^{g_1} = \boldsymbol{x}_i^{g_1} + wf_i(\boldsymbol{x}_{best}^{g_1} - \boldsymbol{x}_i^{g_1}) + wf_i(\boldsymbol{x}_{r_0}^{g_1} - \boldsymbol{x}_{r_1}^{g_1}), \tag{5.1.7}$$

where $\boldsymbol{x}_{best}^{g_1}$ is the agent that has attained the best value so far. Like JADE, the parameters wf_i and cr_i are chosen for each agent in each generation according to $wf_i \sim N(0.5, 0.3)$ and $cr_i \sim N(0.5, 0.1)$. For cr_i the values are maintained for a few generations g_{reset} before a new set is chosen. After a few iterations $g_{learning}$ of choosing new values, the mean of the distribution is updated to be the mean of the successful values used so far. Using these variants with adaptive parameters we can reduce the effect of tuning and the effort required to perform DE.

Likewise, there are many variants of PSO. One of the most popular variants introduces quantum mechanics to take advanatage of the observation that the algorithm converges when each of the agents converges to a so-called local attractor. This is a point between the agent's personal best position $\boldsymbol{p}_i$ and global best position $\boldsymbol{p}_g$. Developed by Sun et al. (2004), this variant is known as Quantum Particle Swarm Optimization (QPSO) because it removes the velocity attribute of each agent in favor of drawing the updated positions from an exponential distribution with parameters based on each agents local attractor and the average best position found so far among all agents. Unlike the DE variants, QPSO has previously been implemented to solve design problems, so we use a design-focused QPSO variant called d-QPSO (Lukemire et al., 2019). We omit a full discussion of the details of d-QPSO but note that there is one new tuning parameter, α, a contraction-expansion parameter that controls the extent to which particles are pulled towards the average best position. Using this variant we can extract useful information from all agents, even those with fitness function values far from the optimum.

Another popular PSO variant is PSO with Extremal Optimization (PSOEO), proposed in Chen et al. (2010). This hybridized variant aims to steer PSO away from premature convergence by increasing its local search capabilities with Extremal Optimization (EO). This means that at every *inv* iteration, where *inv* is a new tuning parameter, we attempt to mutate each individual component of each agent's position to improve its best value attained so far. It is important to note that this mutation operation differs from that of the DE algorithm in that it does not rely on the position of the other agents. If the mutation is successful then the change is kept, otherwise the position is left unchanged. To perform the mutation we start with Gaussian mutation

$$\boldsymbol{x}'_{i,k} = \boldsymbol{x}_{i,k} + N_k(0,1), \tag{5.1.8}$$

where a new random normal value is drawn for each variable k. If the mutated agent falls outside of the search region, then Cauchy mutation

$$\boldsymbol{x}'_{i,k} = \boldsymbol{x}_{i,k} + \delta_k(1), \tag{5.1.9}$$

where $\delta_k(1)$ is a Cauchy random variable with scale parameter equal to one, is used to pull it back. This algorithm hybridizes the standard PSO algorithm, known for having good global search properties, with the local EO search algorithm to facilitate local exploration.

Alongside the standard algorithms, dithering/jittering of the parameters in DE, and linear annealing for the inertia parameter in PSO, these variants will be used to benchmark the performance of metaheuristic algorithms for finding efficient designs to challenging order-of-addition problems. We are of course cognizant of the no free lunch theorem (Wolpert and

Macready, 1997), which states that no algorithm can possibly be the best for an entire class of problems, including those we consider here. The algorithms we consider here are a small sample of the huge number of metaheuristic algorithms that have been proposed, including many hybrid algorithms that aggregate operations and solutions from many individual algorithms. However, our aim here is to give some guidance as to which algorithms among a few selected competitive ones can quickly find efficient designs, and/or to obtain an initial starting point for more specialized design-based algorithms such as Fedorov's algorithm (Fedorov, 1972) or its various extensions (Yang et al., 2013; Hyun and Wong, 2015). Good initial designs are known to improve the performance of these approaches.

5.2 Optimal Order-of-Addition Designs for Challenging Problems

Now that we have established the various metaheuristic algorithms and their extensions that we will use, we introduce the details of two challenging order-of-addition problems we aim to solve with these methods. Both of these problems involve finding order-of-addition designs in situations that are difficult to solve analytically.

The first situation we consider is an elaboration of the problem introduced in Chapter 3 of finding A-optimal designs under the position-based models (3.1.3 - 3.1.5). Recall that the A-optimality criterion given in (2.1.5) is an important measure of a design's ability to generate stable parameter estimates. As shown in Table 3.5, we have found through simple application of DE that the full design $\mathbf{F}_m$ is not A-optimal for the position models. This means that there exist a design for which the A-efficiency relative to the full design is greater than 1. Furthermore, we have seen that A-optimal designs for each model depend on which component effect is removed. Since the choice of which effects to remove from those models is in large part arbitrary or driven by the application, our designs should be robust to this choice.

Considering these findings, the fitness function we will use is the geometric mean A-

efficiency across the entire family of models relative to the first n rows of $\mathbf{F}_{n,m}$, where n will take a range of values. Maximizing this criterion for a choice of m, n and model family should give us a sufficient design that achieves high efficiency regardless of which component effects are removed. For the three position models: first-order, quadratic, and second-order, the geometric mean A-efficiency $\bar{A}(\boldsymbol{\xi})$ for a design $\boldsymbol{\xi}$ relative to $\mathbf{F}_{n,m}$ is given by

$$\bar{A}(\boldsymbol{\xi}) = \Big(\prod_{i=0}^{m-1} \big(tr(\boldsymbol{M}_{(-i)}^{-1}(\boldsymbol{F}_{n,m}))/tr(\boldsymbol{M}_{(-i)}^{-1}(\boldsymbol{\xi}))\big)\Big)^{1/m} \tag{5.2.10}$$

for the first-order and quadratic models, and

$$\bar{A}(\boldsymbol{\xi}) = \Big(\prod_{i,j=0, i\neq j}^{m-1} \big(tr(\boldsymbol{M}_{(-i,-j)}^{-1}(\boldsymbol{F}_{n,m}))/tr(\boldsymbol{M}_{(-i,-j)}^{-1}(\boldsymbol{\xi}))\big)\Big)^{1/(m(m-1))} \tag{5.2.11}$$

for the second-order model.

In the above equations, we first take the product of the relative A-efficiencies under each model in the chosen class. For the first-order and quadratic models this involves removing the linear or linear and quadratic effects of a single component, respectively. We can see this represented in Equation (5.2.10), where removing the effect involving component i results in the information matrix $\boldsymbol{M}_{(-i)}$, where $\boldsymbol{M}_{(-i)}$ has dimension m for the first order model and $2m-1$ for the quadratic model. To arrive at the geometric mean, we then raise the product of the efficiencies under each of these information matrices to $1/m$. For the second-order model, we must also remove a second quadratic effect, that of component j, and all interactions that involve component i to make the model estimable. Considering each combination (i, j) the resulting information matrix is denoted by $\boldsymbol{M}_{(-i,-j)}$ in Equation (5.2.11) with dimension $(m-1)(m+2)/2$, the number of parameters remaining in the model after removing the effects. We raise the product of the efficiencies under each information matrix to $1/(m(m-1))$ to arrive at the final geometric mean. Our aim is to optimize this criterion and generate designs that are robust to the choice of model from a specified family, reducing the risk of model misspecification.

The second problem we consider is that of finding space-filling order-of-addition designs. As described in Section 2.1.4, space-filling designs are useful for understanding the dynamics of deterministic computer systems. By constructing space-filling designs, we can capture a maximal amount of information about fluctuations in the response throughout the parameter space. There have been many studies done for the construction of space-filling designs for general factorial experiments, yet designs for order-of-addition computer experiments have not been constructed (Xiao and Xu, 2017; Wang et al., 2018; Sun et al., 2019). To begin to fill this gap, we consider the construction of maximin and minimax order-of-addition designs. These two criteria are given in (2.1.6) and (2.1.7), respectively, and aim to fill the space by choosing a set of points that either maximizes the minimum pairwise distance between the points in the design or minimizes the maximum distance between the design points and all points in the set.

It is important to note that in both of these problems there is only a single criterion. However, in practice holistic consideration of many properties of a design are necessary to determine if it is the appropriate choice for the application. For example, if there are additional objectives in the study, then multi-objective metaheuristic algorithms should be used to find the optimal design. Such a goal is beyond the scope of this work. In the next section, we outline the setup and results of applying DE, PSO and their popular variants to these two order-of-addition problems.

5.3 Comparison of Designs Derived from Metaheuristic Algorithms

We now investigate the efficacy of PSO, DE and their variants to find valuable order-of-addition designs for the two challenging classes of problems discussed in the previous section. In total, we consider eight algorithms, each of which is described in Section 5.1. Table 5.1 summarizes the tuning parameters associated with each algorithm. We now apply these algorithms to tackle specific cases within the two problem classes of interest.

Table 5.1: Eight metaheuristic algorithms used to find order-of-addition designs and their tuning parameters.

Algorithm	Parameter
Standard Differential Evolution (DE)	population size np mutation factor wf crossover probability cr
DE with Jittering and Dithering (DEJD)	perturbation γ
Jingqiao's Adaptive Differential Evolution (JADE)	best agent percentage p successful values weight c
Self-Adaptive Differential Evolution (SADE)	cr reset interval g_{reset} cr learning interval $g_{learning}$ p_1, p_2 update interval g_{update}
Standard Particle Swarm Optimization (PSO)	population size np intertia τ random weight vectors β_1, β_2 cognitive learning factor cl social learning factor sl
PSO with Linearly Decreasing Inertia (PSOLD)	final inertia τ^*
Quantum Particle Swarm Optimization (QPSO)	contraction-expansion constant α
PSO with Extremal Optimization (PSOEO)	EO frequency inv

Note: For variants of DE and PSO, the table only includes the parameters that are unique to the variant and omits parameters that are shared with the standard algorithm.

In each case, the rules of thumb from the algorithm's literature, some of which have been outlined previously, are used as the initial values for the parameters. In some of the algorithms, particularly the adaptive DE algorithms (JADE and SADE) and PSOLD, some parameters are adaptively chosen throughout the algorithm, and thus the initial values have less impact on the final results. Some tuning is performed in cases where convergence is especially poor, but in order to present a fair comparison, we rely on the published rules given in Table 5.2 along with the range of values considered for tuning. For PSO-based algorithms with a velocity component, the vectors β_1 and β_2 are drawn randomly from a multivariate uniform distribution.

Another important consideration in comparing these algorithms is the number of iterations/generations that are performed. We choose to fix this value across all algorithms instead of using the number of function evaluations, as the fitness function is evaluated the same number of times in nearly all of the algorithms. However, while we somewhat arbitrarily choose a maximum number of iterations that we believe strikes a balance between allowing for proper convergence and limiting computation time, running all of the algorithms longer may produce better results. For all cases, we run each algorithm for 2000 iterations with 5 repetitions, and keep the best value from across the repetitions for each sample size. The next subsection reports the results of applying these algorithms to specific order-of-addition problems.

5.3.1 A-optimal Order-of-Addition Designs for Position-Based Models

For the first problem, we search for designs for standard order-of-addition experiments with m components, where $m = 4, 5, 6$, that are A-optimal (or near-optimal) across different families of position-based models. We consider these values of m because they strike a balance between the importance to practical applications and the level of difficulty that makes them worthy for testing the capability of nature-inspired metaheurisitc algorithms. For each m we consider maximizing the criteria given in (5.2.10) and (5.2.11) for each family

Table 5.2: Default parameter values for finding Order-of-Addition designs with metaheuristic algorithms. In this table the problem dimension is n, the run size of the desired design.

Parameter	Algorithm(s)	Default	Range
np	All	$10n$	Fixed
wf	DE, DEJD	0.8	[0.6,0.9]
cr	DE, DEJD	0.9	[0.8,1]
γ	DEJD	0.003	[0.001, 0.005]
p	JADE	10	[5,20]
c	JADE	0.1	[0.05,0.2]
g_{reset}	SADE	5	[1,10]
$g_{learning}$	SADE	$5g_{reset}$	$[3g_{reset}, 7g_{reset}]$
g_{update}	SADE	50	[25,75]
τ	PSO, PSOLD, PSOEO	0.9	[0.9,1.2]
cl	PSO, PSOLD, PSOEO	2	[1,3]
sl	PSO, PSOLD, PSOEO	2	[1,3]
τ^*	PSOLD	0.4	[0.2,0.6]
α	QPSO	0.9	[0.4,1.4]
inv	PSOEO	50	[25,75]

of position models (first-order, quadratic, and second-order). For each family of models, we search for designs with variable run sizes $p \leq n \leq m(m-1)$, where p is the number of parameters required to estimate the models in each family. The upper bound of $m(m-1)$ is chosen to keep the final designs small enough to be practically useful. As we are interested in finding designs that are robust to the choice of model within a larger family, we do not assess A-optimality directly with the equivalence theorem. We have found that for cases with small values of n and m, a design which is A-optimal under the entire family of models may exist, but an analytic proof of our empirical findings is beyond the scope of the current experiment.

We consider nine situations, one for each combination of m and model family. The following results are representatives of the full experiment. We focus on a few representative examples to conserve space and discuss the high-level conclusions. Figure 5.3 shows a collection of plots for different choices of m and model family. Specifically, in each panel the left plot shows the results of applying the four DE-based algorithms to the problem, while the right plot shows the results of applying PSO-like algorithms. The black horizontal line indicates a mean relative efficiency of 1. Values above this line are indicative of designs that have better mean A-optimality across the model family when compared to the design $\mathbf{F}_{n,m}$ generated by Algorithm 3.1 in Chapter 3. For the case $m = 6$, we use the recursive definition from Huang (2021) to generate a design with the appropriate number of components and runs to use for the comparison.

There are several interesting conclusions we can draw from the generated designs. First, we observe that with the exception of a few cases, the DE and PSO algorithms tend to perform comparably, with the maximal efficiencies in each plot for each run size being close. The one notable exception to this is the case of $m = 5$ for the quadratic position model. For this problem the PSO-based algorithms find much better designs when the run size is low (roughly $20\times$ more efficient than $\mathbf{F}_{n,m}$). Sticking with general observations, we also find that all algorithms are capable of finding designs that greatly improve on the mean

A-efficiency of $\mathbf{F}_{n,m}$ across the range of problems and run sizes. For some specific problems, it seems that there is more room for improvement. This is especially true for small sample sizes when m is small and for large sample sizes when $m = 6$. This aligns with our conclusion from Table 3.5 that the A-efficiency of $\mathbf{F}_m$, and by extension $\mathbf{F}_{n,m}$, decreases as the size and dimension of the design grows. In these cases it is clear that metaheuristic algorithms can generate designs with better A-optimality properties that are more robust to the choice of effect removal in the position-based model family.

We also observe that within the DE-based algorithms, the standard DE algorithm has the poorest performance, but with the exception of the $m = 4$, second-order problem, the losses tend to be small. On the same problem, JADE tends to outperform the other algorithms. This of course seems reasonable as the DE variants have been shown to outperform the standard version on general problems, and JADE has received much attention for its superior convergence. For the PSO-based algorithms, we find that PSOLD and standard PSO perform comparably while QPSO performs comparably to JADE for the simpler problems, with the exception of large run sizes. However, the hybrid PSOEO algorithm consistently performs worse than the others. This could be due to nuances in the search space that make Extremal Optimization inefficient or simply a feature of the algorithm that it requires more iterations to converge for this problem.

To make the benefits of the designs found via metaheuristic algorithms more concrete, the left panel of Table 5.3 displays the exact design found by JADE for the case that $m = 4$, $n = 12$ and the model family is the second-order position model. This design is represented by $\mathbf{JADE}_{12,4}$. The middle panel contains the design $\mathbf{F}_{12,4}$, which is used as the benchmark for computing the A-efficiency under each model. The A-optimality is given for each design under all models in the second-order position family in the right panel, with the last row giving the geometric mean across all models.

We see from this table that the improvement in A-optimality of the $\mathbf{JADE}_{12,4}$ design is spread fairly across each of the twelve models. However, it is important to note that under

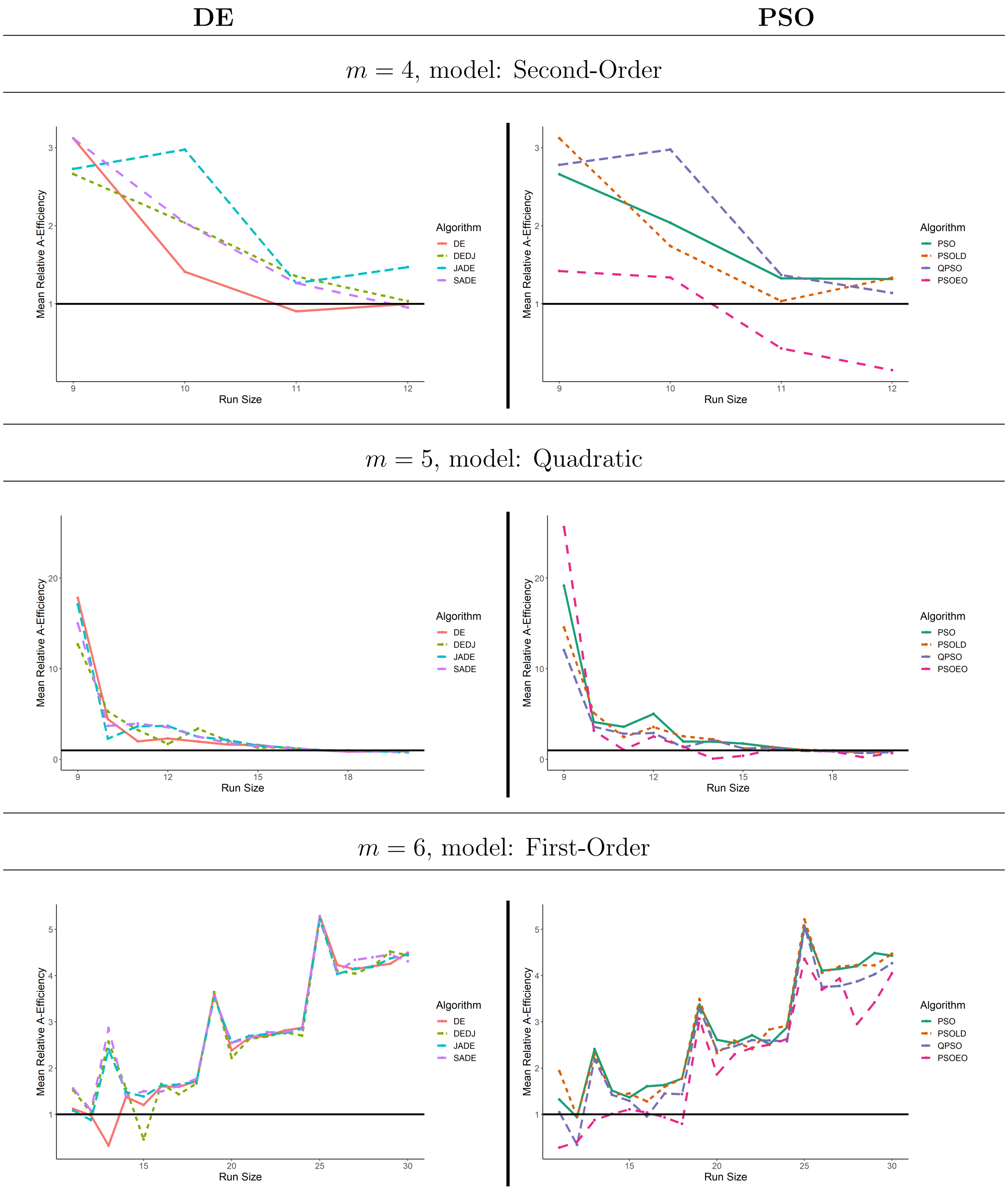


Figure 5.3: Mean relative A-efficiency of designs found via DE, PSO and related algorithms for order-of-addition problems with $m = 4, 5, 6$ under different position-based models and variable run sizes.

Table 5.3: The design found by JADE with $n = 12, m = 4$, $\mathbf{JADE}_{12,4}$, compared to $\mathbf{F}_{12,4}$ constructed via Algorithm 3.1 in terms of geometric mean A-efficiency under the second-order position model family.

$\mathbf{JADE}_{12,4}$

Run	a_1	a_2	a_3	a_4
1	0	1	3	2
2	0	2	3	1
3	0	3	1	2
4	1	0	2	3
5	1	3	2	0
6	2	0	1	3
7	2	1	0	3
8	2	1	3	0
9	2	3	1	0
10	3	0	2	1
11	3	1	0	2
12	3	2	0	1

$\mathbf{F}_{12,4}$

Run	a_1	a_2	a_3	a_4
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2
4	1	0	3	2
5	1	2	0	3
6	1	3	2	0
7	2	0	1	3
8	2	1	3	0
9	2	3	0	1
10	3	0	2	1
11	3	1	0	2
12	3	2	1	0

Model	$\mathbf{F}_{12,4}$	$\mathbf{JADE}_{12,4}$	A-Eff.
$\boldsymbol{M}_{(-0,-1)}$	17.50	11.77	1.49
$\boldsymbol{M}_{(-0,-2)}$	26.53	11.77	2.25
$\boldsymbol{M}_{(-0,-3)}$	17.50	18.17	0.96
$\boldsymbol{M}_{(-1,-0)}$	17.50	12.51	1.40
$\boldsymbol{M}_{(-1,-2)}$	17.50	21.51	0.81
$\boldsymbol{M}_{(-1,-3)}$	26.53	12.51	2.12
$\boldsymbol{M}_{(-2,-0)}$	26.53	11.13	2.38
$\boldsymbol{M}_{(-2,-1)}$	17.50	16.55	1.06
$\boldsymbol{M}_{(-2,-3)}$	17.50	11.13	1.57
$\boldsymbol{M}_{(-3,-0)}$	17.50	17.70	0.99
$\boldsymbol{M}_{(-3,-1)}$	26.53	11.74	2.26
$\boldsymbol{M}_{(-3,-2)}$	17.50	11.74	1.49
Mean	20.10	13.66	1.47

three of the models the A-optimality value of the new design is actually larger. For this reason, it is critical that the context and goals of the experiment be taken into consideration when choosing a design. Our proposed design shows improved robustness to the choice of the removed effect; however, if the true data-generating process follows one of the models for which the A-efficiency is lower, then using this design may result in less stable parameter estimates.

5.3.2 Space-filling Designs for Standard and Screening Order-of-Addition Experiments

Having shown that metaheuristic algorithms are able to find designs with improved A-efficiency for the position-based models, we now turn our attention to the second order-of-addition problem. For the case of space-filling designs for order-of-addition experiments, we again consider $m = 4, 5, 6$ with variable run size $m \leq n \leq m(m-1)$. For each of these situations we search for designs that minimize the maximal pairwise distance between the rows of the design and all points in the space (minimax) and designs that maximize the minimal pairwise distance between rows of the design (maximin). This is equivalent to minimizing (2.1.7) and minimizing the negation of (2.1.6).

We consider each of these situations in the context of two problems: the standard order-of-addition problem from Chapter 3, and the order-of-addition component screening problem of Chapter 4. For the screening problem, we specifically consider the cases $(m = 4, q = 3), (m = 6, q = 3)$ and $(m = 6, q = 4)$. To visualize the improvement of the designs found by each algorithm over the existing ones we compute the relative maximin and minimax efficiencies. The maximin and minimax efficiencies of a design $\boldsymbol{\xi}_1$ relative to $\boldsymbol{\xi}_2$ are given by

$$\min_{x,y\in\boldsymbol{\xi}_1} d(x,y) / \min_{x,y\in\boldsymbol{\xi}_2} d(x,y) \tag{5.3.12}$$

$$\max_{y\in\boldsymbol{\xi}_2} d(y, \boldsymbol{S}_{m,q}) / \max_{y\in\boldsymbol{\xi}_1} d(y, \boldsymbol{S}_{m,q}), \tag{5.3.13}$$

respectively, where $d(x, y)$ is the L_2 distance between points x and y and $d(y, \boldsymbol{S}_{m,q}) = \min_{x\in\boldsymbol{S}_{m,q}} d(x, y)$. In these definitions we have used the full screening design $\mathbf{S}_{m,q}$ to represent the entire set of candidate points. For the standard order-of-addition problem we would instead use $\mathbf{F}_m$, the design with all $m!$ permutations. If the design $\boldsymbol{\xi}_1$ is more space-filling than $\boldsymbol{\xi}_2$ under either criteria, then the respective efficiency will be larger than 1. For the standard order-of-addition problem we use $\mathbf{F}_{n,m}$ from Algorithm 3.1 for $\boldsymbol{\xi}_2$ when m is prime or a prime power and COA designs from Huang (2021) otherwise. For the order-of-addition screening problem we use the appropriate $\mathbf{S^P}_{n,m,q}$ designs constructed for the PWOS model given in Algorithms 4.2 and 4.3. For designs generated using Algorithm 4.3 with $q > 3$ positions we use designs presented in Schoen and Mee (2021) as the OofA-OA in Step 1. For this study our search space is the set of all component matrices $\boldsymbol{A}$ with the specified number of runs. We could instead consider the set of all position matrices $\boldsymbol{B}$, but for the standard order-of-addition problem under the L_2 distance there is a one-to-one mapping between these spaces, and for the screening problem the position matrix is not well-defined. Further research is needed to determine a proper distance for working with position matrices in this case, so we proceed with the component matrices instead.

With these criteria established, we use the eight algorithms to search for both minimax and maximin designs for six specific problems, three standard and three screening. As with the previous problem, we only provide a slice of the results obtained to conserve space, but present both general and specific conclusions. For the standard problem, we display 2 minimax and 1 maximin examples, whereas for the screening problem we display 1 minimax and 2 maximin examples to demonstrate the various conclusions reached for different problems. Also, like the previous problem, we tune the parameters within the ranges displayed in Table 5.2, but omit the full details. All other details of the experiment (number of iterations, initial parameter values, etc.) are retained from the previous problem.

Figure 5.4 presents the results of applying the eight algorithms to find space-filling designs for the standard order-of-addition problem. The left panel in each row shows the efficiency of the designs found by DE-based algorithms under the specified criterion, while the right panel shows the efficiency of the designs found via PSO-based algorithms. The solid black line indicates an efficiency of 1. Points above this line indicate that the corresponding design has better space-filling properties than the existing design.

From these plots we draw several general conclusions. First, we again observe that the DE-based and PSO-based algorithms tend to perform comparably. Notable exceptions are the minimax designs found via QPSO and PSOEO, which tend to have much better performance across small run sizes for $m = 4$ and large run sizes for $m = 5$. Second,

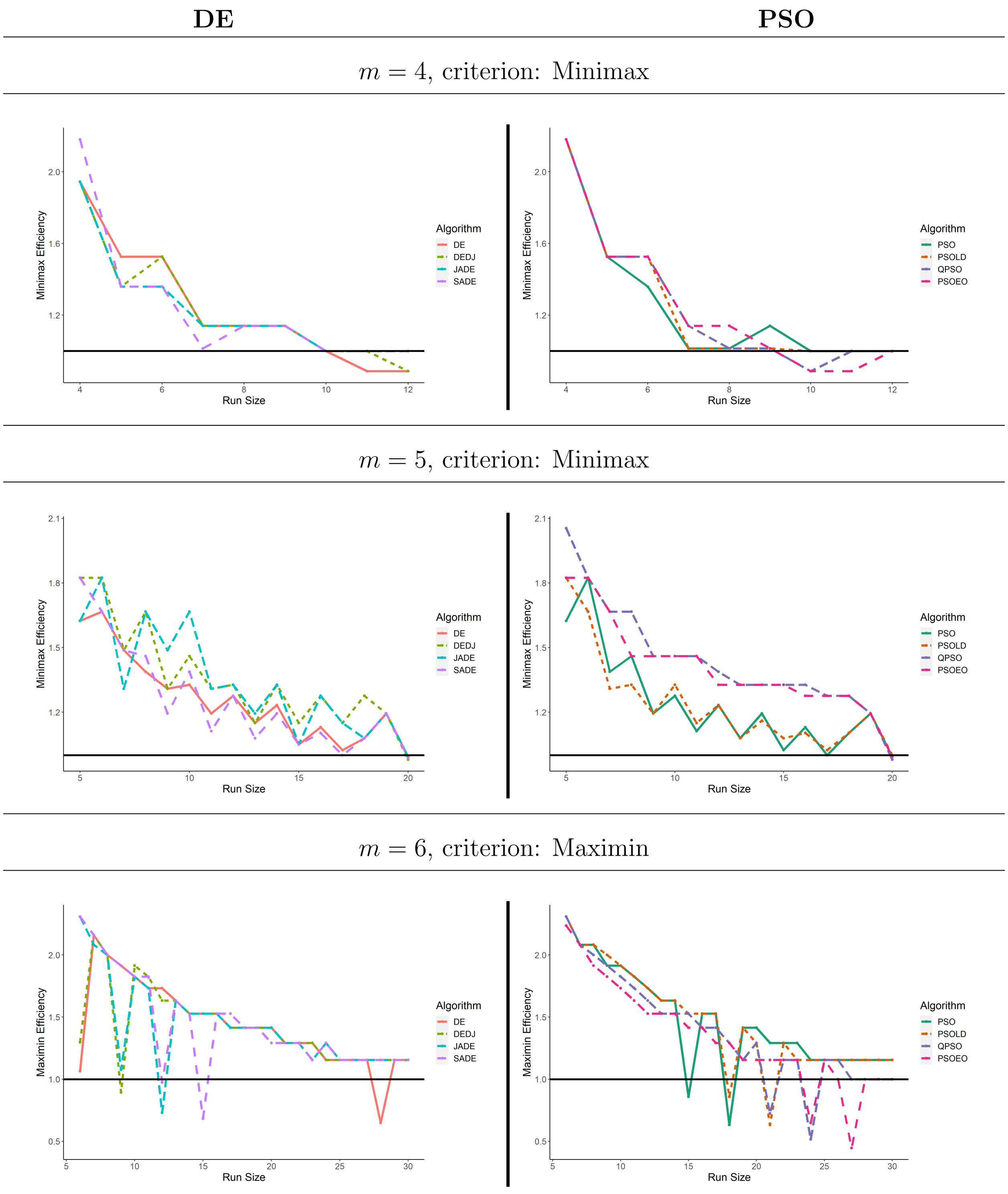


Figure 5.4: Relative space-filling efficiency of designs found via DE, PSO and related algorithms for the standard order-of-addition problem with $m = 4, 5, 6$ under the minimax and maximin criteria.

we observe that across all problems and sample sizes, there is at least one algorithm that produces a design with space-filling properties at least as good as, but often better than, the reference design. This conclusion holds more generally for all algorithms when searching for minimax designs and for smaller run sizes. This may indicate that the algorithms are not suitable for finding maximin designs, or perhaps that the existing designs already have good maximin properties and good space-filling properties in general for larger run sizes. Specifically, we notice that for smaller run sizes, the PSO-based algorithms produce better maximin designs while designs from the DE-based algorithms perform better for larger run sizes. One other notable conclusion is that no algorithm is able to generate a minimax design that outperforms the reference design when $n = m(m-1)$. In fact, it seems that the $\mathbf{F}_{n,m}$ designs may be optimal with respect to the minimax criterion when $n = m(m-1)$, but this claim requires further investigation to verify.

Unlike the previous problem, we notice that apart from the noted exceptions, all algorithms perform comparably. However, when it comes to the maximin problem, we notice that for each run size, one or two of the eight algorithms find suitable designs, while the others fail to converge to a design that outperforms the reference design. This may indicate that for some problems it is more reasonable to consider the output of multiple algorithms or a hybrid version that borrows elements from both DE and PSO.

To gain additional insight into the designs generated by these algorithms, Table 5.4 gives an example of the minimax design generated by QPSO for the case $m = 5$. Since we hypothesize that the reference design is optimal under the minimax criterion, a comparison with $n = m(m-1)$ would not be interesting. Instead we choose a different multiple of m, $n = 15$. The table compares this design, labeled as $\mathbf{QPSO}_{15,5}$, to the reference design $\mathbf{F}_{15,5}$.

We can see from this table that the proposed design has a minimax distance that is roughly 75% that of the existing one, indicating that it is much more space-filling. Studying the distances between the individual design points and the remaining 105 points in the space more carefully reveals that there are five points in the space that are a distance of 3.16 from

Table 5.4: The design found by QPSO with $n = 15, m = 5$, $\mathbf{QPSO}_{15,5}$, compared to $\mathbf{F}_{15,5}$ constructed via Algorithm 3.1 in terms of the maximum distance between each point in the space and its closest point in the design.

	Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	a_1	0	0	0	0	1	1	1	2	2	2	3	3	3	4	4
$\mathbf{QPSO}_{15,5}$	a_2	1	1	1	3	4	4	4	1	3	4	0	1	1	1	3
max: 2.45	a_3	2	2	3	4	0	2	3	3	4	1	4	0	2	0	2
	a_4	3	4	2	2	3	3	2	4	0	0	2	4	0	2	1
	a_5	4	3	4	1	2	0	0	0	1	3	1	2	4	3	0

	Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	a_1	0	0	0	1	1	1	2	2	2	3	3	3	4	4	4
$\mathbf{F}_{15,5}$	a_2	1	2	3	2	3	4	0	3	4	0	1	4	0	1	2
max: 3.16	a_3	2	4	1	3	0	2	3	4	1	2	4	0	1	3	0
	a_4	3	1	4	4	2	0	1	0	3	4	2	1	2	0	3
	a_5	4	3	2	0	4	3	4	1	0	1	0	2	3	2	1

their closest point in the design $\mathbf{F}_{15,5}$. Table 5.5(a) shows these 5 points along with a design point that achieves the minimum distance to each. The two rightmost columns give the points in $\mathbf{QPSO}_{15,5}$ that are closest to these five points and their respective distances.

From this table we can see that the design generated by QPSO has reduced distances to the five problematic points and thus has a smaller minimax criterion value. Meanwhile Table 5.5(b) gives the distributions of the minimum distances between the points in the space and the two designs. There are 25 points in the space that are a distance of 2.45 from the closest point in $\mathbf{QPSO}_{15,5}$ while the design $\mathbf{F}_{15,5}$ only has 15 such points. From the distance distributions we find that the average distance to the design is lower for the existing design (1.78 versus 1.82) even though its minimax criterion is larger. Thus, it is important to remember that in this example we are only considering a single criterion. It could be that the existing design has other valuable space-filling properties that would come to light when considered holistically. Nevertheless, the proposed design provides a window into the structure of the theoretically-optimal minimax design.

When considering the order-of-addition screening problem, we notice similar trends in

Table 5.5: (a) The five points in the set $\mathbf{F}_5$ that are the furthest from $\mathbf{F}_{15,5}$. $\mathbf{F}^*$ and $\mathbf{QPSO}^*$ represent the points in $\mathbf{F}_{15,5}$ and $\mathbf{QPSO}_{15,5}$ that are the closest to each point. The corresponding distances are given by $\mathbf{F}^*_{dist}$ and $\mathbf{QPSO}^*_{dist}$. (b) The distribution of minimum distances between the 105 points not included in the designs and the two different designs.

(a)

	$\mathbf{F}^*$	$\mathbf{F}^*_{dist}$	$\mathbf{QPSO}^*$	$\mathbf{QPSO}^*_{dist}$
04321	12340	3.16	03421	1.41
10432	01234	3.16	30421	2.45
21043	40123	3.16	31042	1.41
32104	34012	3.16	31204	1.41
43210	23401	3.16	43210	0

(b)

distance	1.41	2	2.45	3.16
$\mathbf{QPSO}_{15,5}$	51	29	25	0
$\mathbf{F}_{15,5}$	60	25	15	5

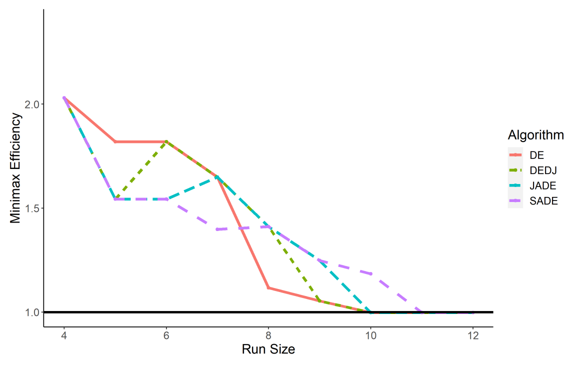
Figure 5.5, which shows a few examples of applying each algorithm to find designs with fixed values of m and q under each fitness function. Specifically, all eight algorithms achieve similar performance, improving on the reference designs. On the maximin problems it appears that the DE-based algorithms slightly outperform the PSO-based algorithms in some cases. For larger sample sizes under the minimax criterion, the opposite seems to be true, with PSO-based algorithms finding designs with better space-filling properties. Both of these observations could also be the beginning of larger trends that occur as m increases.

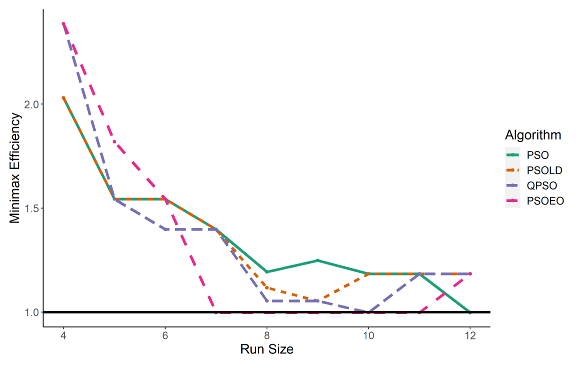
Like the previous problem, we observe that the proposed designs are much more space-filling than the $\mathbf{S^p}_{n,m,q}$ designs under the chosen criterion. This observation holds for each of the situations we consider, with the exception of the maximin designs found by the PSO-based algorithms for a few large run sizes. This difference can likely be attributed to either the increased dimension of the problem, the space-filling properties of the existing designs that make it difficult to beat, or both.

Table 5.6 shows a specific example of a maximin space-filling design for the case $n = 20, m = 6$, and $q = 4$. We compare this design, labeled $\mathbf{PSO}_{20,6,4}$, to the appropriate

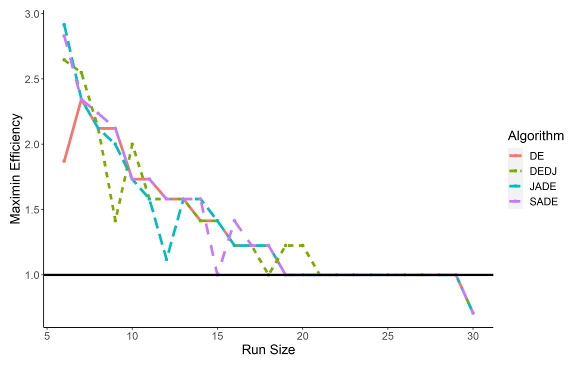
DE **PSO**

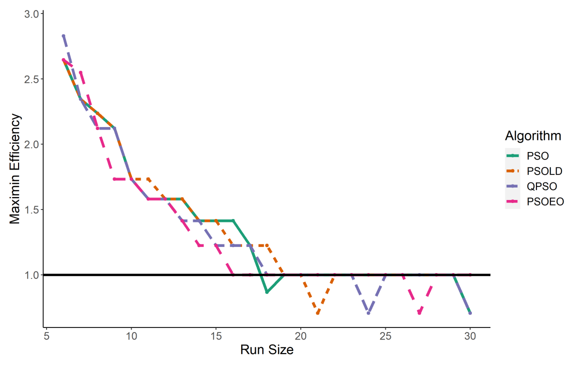
$m = 4, q = 3$, criterion: Minimax



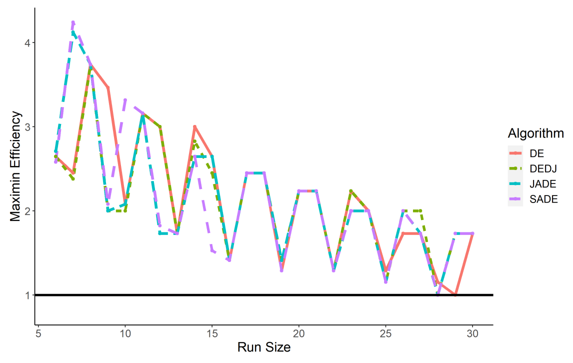


$m = 6, q = 3$, criterion: Maximin





$m = 6, q = 4$, criterion: Maximin



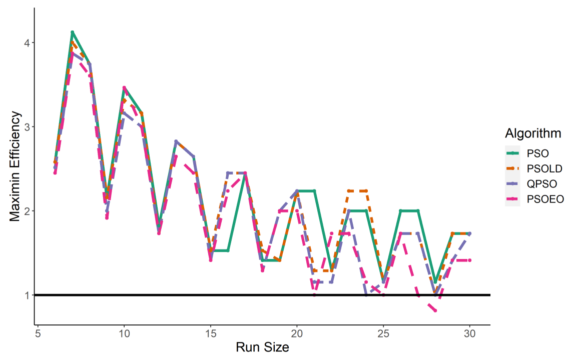


Figure 5.5: Relative space-filling efficiency of designs found via DE, PSO and related algorithms for the order-of-addition screening problem with $(m, q) = (4, 3), (6, 3), (6, 4)$ under the minimax and maximin criteria.

Table 5.6: The design found by PSO with $n = 20, m = 6, q = 4$, $\mathbf{PSO}_{20,6,4}$, compared to $\mathbf{S^p}_{20,6,4}$ constructed via Algorithm 4.3 in terms of the minimum pairwise distance between runs.

	Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	a_1	0	0	0	0	1	2	2	2	2	3	3	3	3	3	3	4	4	5	5	5
$\mathbf{PSO}_{20,6,4}$	a_2	1	1	4	4	3	0	3	3	5	0	1	1	4	5	5	0	5	2	2	4
min: 2.24	a_3	2	3	3	5	0	3	4	5	0	5	0	2	1	1	4	5	1	0	4	1
	a_4	3	5	5	1	5	4	5	1	3	4	5	0	5	0	2	1	3	4	3	0

	Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	a_1	0	0	0	0	0	0	0	1	1	2	2	2	3	3	3	3	5	5	5	5
$\mathbf{S^p}_{20,6,4}$	a_2	1	1	1	2	2	4	5	2	3	4	4	4	2	2	4	5	2	3	3	4
min: 1	a_3	2	3	4	4	5	3	2	4	5	0	0	5	1	5	0	0	0	0	4	0
	a_4	3	2	2	1	1	1	1	5	4	3	5	3	4	1	1	1	3	1	0	1

design generated by Algorithm 4.3, $\mathbf{S^p}_{20,6,4}$. We observe from this table that the minimum pairwise distance between the runs of the proposed design is 2.24 times larger than that of the existing designs, indicating that this new design is much more space-filling in terms of the maximin criterion. To further demonstrate this point, Figure 5.6 shows all two-dimensional projections of each design. The black points come from $\mathbf{PSO}_{20,6,4}$ while the red points come from $\mathbf{S^p}_{20,6,4}$. The size of each point indicates the number of replicates at the site. From these projection plots we can visually conclude that the design found by PSO is much more space-filling, especially with respect to the projection of each design into positions 1 and 3 as well as 1 and 4. In these projections, the existing design has large areas without any observations and other areas with clusters of replicated points. In contrast, the proposed design has fewer replicates and covers more of the space without many noticeably large gaps.

5.4 Chapter Summary

Nature-inspired metaheuristic algorithms have recently become quite popular for solving problems in the field of design of experiments. In this chapter we have considered the efficacy

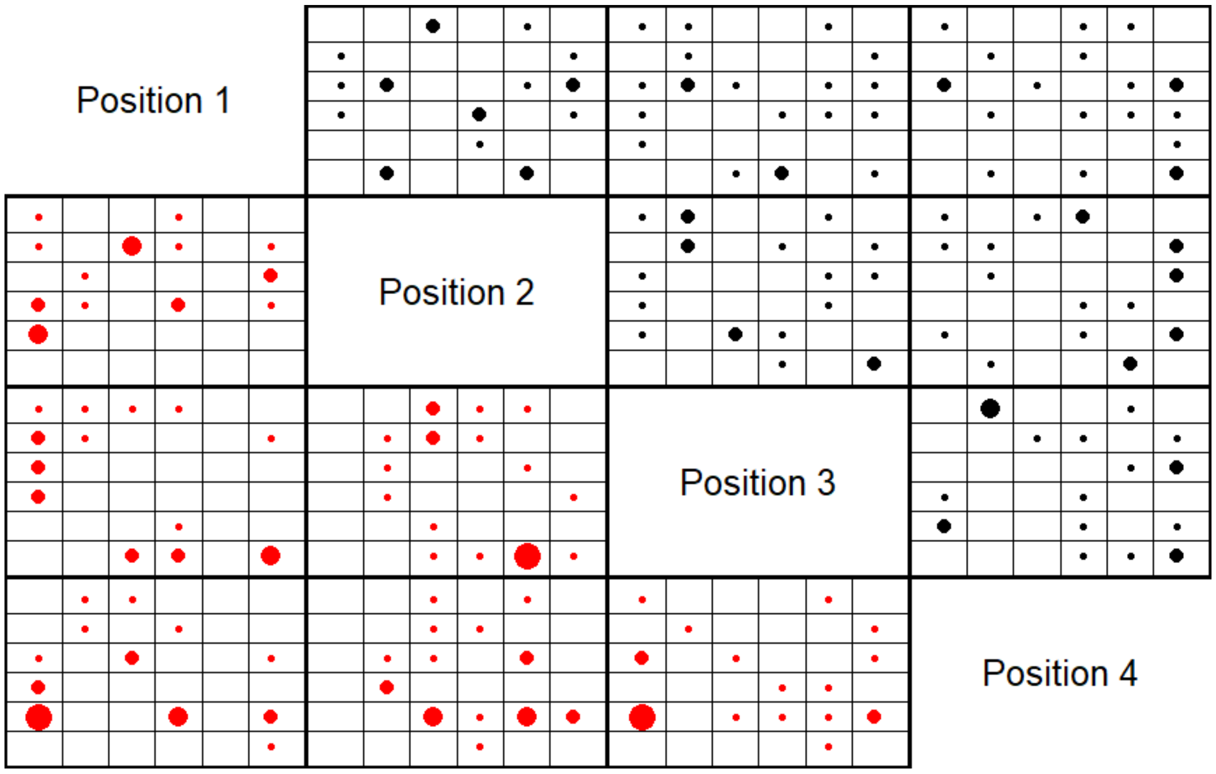


Figure 5.6: Two-dimensional projections for the design found via PSO, $\mathbf{PSO}_{20,6,4}$, in black and the design generated by Algorithm 4.3, $\mathbf{S^p}_{20,6,4}$, in red. The size of each point reflects the number of replicates at the site.

of these algorithms to find designs for challenging order-of-addition problems. Specifically, we considered eight algorithms from the two major classes of nature-inspired metaheuristic algorithms. From the class of evolutionary algorithms we considered Differential Evolution (DE) and three of its variants: DE with dithering and jittering (DEDJ) and two adaptive parameter algorithms, JADE and SADE. From the class of swarm intelligence algorithms we implemented Particle Swarm Optimization (PSO) and three of its variants: PSO with linearly decreasing inertia (PSOLD), quantum PSO (QPSO), and PSO with Extremal Optimization (PSOEO).

We applied these eight algorithms to two critical problems in the design of order-of-addition experiments. First, we considered the problem of finding efficient, A-optimal designs for position-based models that are robust to the choice of effect removal. While deriving these designs analytically is challenging, we have found that JADE and QPSO are well-suited for this problem when working with smaller cases. As m increases we truly see the benefit of working with metaheuristic algorithms as the A-efficiency of all eight designs grows dramatically relative to the existing position-based designs.

Next we considered the problem of finding space-filling designs for use in order-of-addition computer experiments, either for the standard problem or the component screening problem. In both of these cases, we find that for small sample sizes all algorithms are capable of generating designs with greatly improved space-filling properties relative to the existing designs, with the size of the benefit diminishing as the run size of the desired design increases. While none of the algorithms stand out as better or worse than the others in general, except for JADE and PSO in a few cases, we note that it would be prudent to consider multiple algorithms or a DE-PSO hybrid approach for a given problem.

While we have not seen a strong effect of the choice of algorithm class between evolutionary algorithms and swarm intelligence, we have found in all cases that metahueristics in general are capable of quickly producing order-of-addition designs with desirable qualities without requiring substantial parameter tuning. These designs can then be used as the starting point for further manipulation to make them more suitable for the application at hand and may also provide inspiration for the development of theoretical constructions.

CHAPTER 6

Concluding Remarks

Following Fisher's remarkable "Lady Tasting Tea" experiment, problems involving the relationship between a sequence of steps or the sequential administration of components have developed across many fields. Social scientists have studied the effect that candidate name order on election ballots has on the outcome of senate races (Grant, 2017), engineers have considered the relationship between weld sequence and the structural properties of gas turbines (Voutchkov et al., 2005), and chemists have examined the impact of reagent addition order on the endpoint reaction performance in the large-scale preparation of aromatic nitriles (Ryberg, 2008), to name a few. However, despite widespread application, a large gap exists in the statistical treatment of these and other order-of-addition experiments. Without proper experimental designs, these researchers are left to either perform every possible combination or use some ad-hoc selection of runs. This can of course quickly become impossible when, for example, each run requires 32 hours to complete, as in Voutchkov et al. (2005). Furthermore, without robust modeling approaches, analysts are required to adapt existing models from their field, which are often ill-fitted to the ordering effect present in the data.

To address the needs of these researchers, we have filled several sizable gaps in the order-of-addition literature in this dissertation. Specifically, we have developed a class of flexible, position-based models for the standard order-of-addition problem in addition to a robust design construction. These methods can be used in concert to conduct efficient, cost-effective experiments. We are among the first to consider designs for order-of-addition screening experiments, establishing several constructions that produce optimal and near-optimal designs for the standard order-of-addition screening models. Finally, for some problems analytic constructions are difficult to create, yet the resulting designs are greatly needed in practice. To fill this gap we have explored the efficacy of several prominent nature-inspired, metaheuristic algorithms to locate designs with suitable efficiency and space-filling properties, finding that variants and hybridized versions of standard algorithms can find designs that far outperform the existing ones.

6.1 Summary of Results

First, Chapter 3 considers the standard order-of-addition problem. Our main contribution is the idea of position-based models. By altering the perspective for the analysis of order-of-addition data to focus on the effects of individual components, we open up the possibility of modeling with orthogonal polynomials. Applying this new class of position-based models to the critical application of sequential drug administration, we have found that they are capable of elucidating the effect of individual drug's positioning on the endpoint effectiveness of the treatment while also generating accurate predictions as to the best sequence. This is a bar that existing models fall short of without the addition of many costly parameters. To support these new models, we have used the theory of Latin squares to construct optimal designs that dramatically reduce the cost of running order-of-addition experiments. Together these methods provide a framework for the design and analysis of order-of-addition experiments that is robust to assumptions about the relationship between components in the ordering effect.

Next, Chapter 4 tackles the difficulties that accompany the problem of screening a pool of components for the most effective subset when there is also a suspected ordering effect. This screening problem requires adaptations of existing order-of-addition models, namely the component-position and pariwise ordering models. However, the real contribution here is in the three design construction algorithms. Between these constructions, all supported by the theory of optimal design, we have covered many potential cases that a researcher may encounter in order-of-addition screening experiments. Applying these results to the field of operations research through a job scheduling application, we have found that the designs offer a cheap, effective alternative to performing all sequences, and when used in conjunction with an active learning optimization scheme they can lead to earlier convergence to the best job sequence than a random design.

Finally, while we can use our proposed models and design constructions to solve many important problems in the field of order-of-addition experiments, there are still many open problems without theoretically-supported solutions. This includes the development of A-optimal designs that are robust to the choice of position-based model and space-filling designs for computer experiments to study deterministic systems that take a permutation as the input. To solve these important and computationally difficult problems, we turn to the class of nature-inspired metaheuristic algorithms. Specifically, we investigate the performances of two prominent members of this class, Differential Evolution (DE) and Particle Swarm Optimization (PSO), along with several of their popular variants. Through a detailed study of eight algorithms we were able to find designs that show great improvement over the existing ones in terms of the criteria of interest, with adaptive parameter variants of DE and hybridized versions of PSO showing particular promise. Being the first to showcase the benefit of applying these algorithms to order-of-addition problems, we have handled several more problems that researchers may encounter when studying ordering effects in practice. Furthermore, having found initial success, we can now study the properties of the resulting designs from these metaheuristic algorithms with the aim of either obtaining further improvements or generalizing them into a well-defined construction.

6.2 Future Work

This dissertation provides substantial contributions to several pressing problems in the design and analysis of order-of-addition experiments, but the statistical treatment of these experiments is still in its early stages. There are of course many open areas of research for creating or improving methods for tackling other practical problems. This section reviews a few such problems.

We have considered in Chapter 4 how to screen components when the size of the pool is larger than the number of positions available. However, another problem faced in practice is deciding how many positions may be appropriate. For example, in the drug sequencing applications from Yang et al. (2021) and Mee (2020), it is not clear whether five drugs are necessary, or if the maximum benefit can be obtained with four drugs or less. Designing and analyzing an experiment in which the number of drugs administered varies by run continues to be an open area of research. One idea is to concatenate the screening designs we have constructed in Chapter 4 for every possible value of $1 < q < m$ and further adapt the standard order-of-addition models to accommodate a design with a variable number of components per run. Another approach would be to design an adaptive experiment with at least two stages in which the pool of components is reduced in each stage until the relationship between the choice, size, and order of the sequence is understood.

In many applications we are not solely interested in understanding the ordering effect; in fact, we may be more convinced that additional covariates associated with each component are more important than the order. Sticking with the sequential drug administration application, this problem arises in the determination of the proper dosage for each drug. For example, Wang et al. (2020) studied three drugs, with two of the drugs having two dosages per drug and the dosage of the third drug being fixed. To study this relationship, the researchers used a cross-array design, combining an order-of-addition design with an orthogonal array for studying the dosage effect. However, for many situations the resulting design would be too large to be practically useful, and it raises concerns about the interaction effects between the dosages and the ordering. Xiao et al. (2020) instead approached the problem by using a Gaussian Process-based model within an active learning framework to reduce the computational complexity of locating the optimal dosage and position of each drug. Since this approach employs a spatial Gaussian Process model, this is one area where the space-filling, order-of-addition designs we found in Chapter 5 could lead to potential improvements in the convergence speed, but further research is necessary.

Lastly, another practical area of importance in the study of order-of-addition is that of hard-to-change components. In the event that the positions of at least two components are difficult to change, a design with completely randomized run order may lead to additional costs associated with the extra effort required to change the positions. For example, this type of practical problem may occur in operations research and engineering applications if different steps of a job require heavy machinery or additional manpower. Generalizing this, if the positions of all components are difficult to change, then the number of position changes required between runs is an important consideration. As another example, consider the toy problem of sequentially arranging a set heavy stones to build the most efficient makeshift bridge across a river. An improper choice of design could lead to a substantial increase in the number of times each stone must be lifted and the distance it must be carried. The solution to this problem, minimum run-change designs, have been studied for specific design structures, such as two-level factorial designs (Cheng et al., 1998) and Plackett-Burman designs (Quinlan and Lin, 2015); however, the direct study of run order in optimal order-of-addition designs is still an open problem.

APPENDIX A

Proofs

Proof of Theorem 3.1. (i) By Corollary 6 (i) of Xu (2003), we have $W_1 \geq mr(m-r)/n^2$, with equality if and only if each component appears as equally often as possible in every column. When $n = qm + r$, $\mathbf{F}_{n,m}$ contains q $m \times m$ Latin squares, so each level appears in each column q times in the first qm runs of $\mathbf{F}_{n,m}$. Each column of the last r runs of $\mathbf{F}_{n,m}$ contains each level at most once, meaning that the maximum difference in the number of occurrences of each level per column is 1. Thus, $\mathbf{F}_{n,m}$ has minimum $W_1 = mr(m-r)/n^2$ among all possible designs.

(ii) This is a direct result of (i).

(iii) We show that $\mathbf{F}_{n,m}$ is an orthogonal array of weak strength t for all $t \geq 1$. A design is an orthogonal array of weak strength t if all possible level combinations for any t columns appear as equally often as possible (Xu, 2003). From (i) we know that $\mathbf{F}_{n,m}$ has minimum W_1. Since $\mathbf{F}_{m(m-1),m}$ is a COA with the property that every pair of level combinations shows up exactly once, we know that the sub-design $\mathbf{F}_{n,m}$, $n \leq m(m-1)$, contains each pair of level combinations either 0 or 1 times. Since $n \leq m(m-1)$, $\mathbf{F}_{n,m}$ is an orthogonal array of weak strength t for all $t \geq 1$. Hence, by Theorems 2 and 3 of Xu (2003), design $\mathbf{F}_{n,m}$ has generalized minimum aberration among all possible designs.

(iv) If $n = m(m-1)$, then the claim is true by the mutual orthogonality of the Latin squares derived in Step 1 of Algorithm 1. The two COA properties of a design are invariant with respect to column permutation. Therefore, each $\boldsymbol{C}_i, i = 1, \ldots, (m-2)!$, is also a COA$(m(m-1), m)$ and for any $n > 0$ such that $n/(m(m-1)) = \lambda$ is an integer, concatenating the λ COA$(m(m-1), m)$ designs produces a COA(n, m). $\square$

Proof of Theorem 3.2. We prove the claim for any COA(n, m) as the full design $\mathbf{F}_m$ is a special case with $n = m!$. For a point $\boldsymbol{b} = (b_1, \ldots, b_m) \in \mathcal{X}$, the vector of regression functions under the quadratic model (3.1.4) is

$$\boldsymbol{f}(\boldsymbol{b}) = (1, p_1(b_1), \ldots, p_1(b_{m-1}), p_2(b_1), \ldots, p_2(b_{m-1}))^{\mathrm{T}},$$

with the first m terms being the regression functions under the first-order model (3.1.3). For any COA(n, m) the information matrix under the quadratic model and its inverse take the form

$$\begin{aligned}
\boldsymbol{M}(\boldsymbol{\xi}) &= \begin{bmatrix} 1 & \mathbf{0}_{m-1}^{\mathrm{T}} & \mathbf{0}_{m-1}^{\mathrm{T}} \\ \mathbf{0}_{m-1} & \delta\mathbf{J}_{(m-1)} + (1-\delta)\mathbf{I}_{(m-1)} & \mathbf{0}_{(m-1)\times(m-1)} \\ \mathbf{0}_{m-1} & \mathbf{0}_{(m-1)\times(m-1)} & \delta\mathbf{J}_{(m-1)} + (1-\delta)\mathbf{I}_{(m-1)} \end{bmatrix}, \\
\boldsymbol{M}(\boldsymbol{\xi})^{-1} &= \begin{bmatrix} 1 & \mathbf{0}_{m-1}^{\mathrm{T}} & \mathbf{0}_{m-1}^{\mathrm{T}} \\ \mathbf{0}_{m-1} & -(m\delta)^{-1}(\mathbf{J}_{(m-1)} + \mathbf{I}_{(m-1)}) & \mathbf{0}_{(m-1)\times(m-1)} \\ \mathbf{0}_{m-1} & \mathbf{0}_{(m-1)\times(m-1)} & -(m\delta)^{-1}(\mathbf{J}_{(m-1)} + \mathbf{I}_{(m-1)}) \end{bmatrix},
\end{aligned}$$

where $\delta = -1/(m-1)$, $\mathbf{0}_k$ is a column vector of 0's, $\mathbf{J}_k$ is a $k \times k$ matrix of 1's, and $\mathbf{I}_k$ is the $k \times k$ identity matrix. The top left 2×2 submatrix in both cases is the equivalent information matrix and inverse under the first-order model. Now we can apply the checking condition (2.1.4) provided by the equivalence theorem and exploit the properties of the orthogonal polynomial contrasts (3.1.2).

$$\begin{aligned}
\boldsymbol{f}(\boldsymbol{b})^{\mathrm{T}}\boldsymbol{M}(\boldsymbol{\xi})^{-1}\boldsymbol{f}(\boldsymbol{b}) &= 1 - \frac{2}{m\delta}\sum_{k=1}^{m-1} p_1^2(b_k) - \frac{2}{m\delta}\sum_{k=1}^{m-2}\sum_{l>k}^{m-1} p_1(b_k)p_1(b_l) \\
&\quad - \frac{2}{m\delta}\sum_{k=1}^{m-1} p_2^2(b_k) - \frac{2}{m\delta}\sum_{k=1}^{m-2}\sum_{l>k}^{m-1} p_2(b_k)p_2(b_l).
\end{aligned}$$

Because $\boldsymbol{b} = (b_1, \ldots, b_m)$ is a permutation of $\{1, \ldots, m\}$, using (3.1.2) and some algebra, for $j = 1, 2$, we have

$$2\sum_{k=1}^{m-1} p_j^2(b_k) + 2\sum_{k=1}^{m-2}\sum_{l>k}^{m-1} p_j(b_k)p_j(b_l) = m.$$

Therefore,

$$\boldsymbol{f}(\boldsymbol{b})^{\mathrm{T}}\boldsymbol{M}(\boldsymbol{\xi})^{-1}\boldsymbol{f}(\boldsymbol{b}) = 1 - \frac{1}{\delta} - \frac{1}{\delta} = 1 + (m-1) + (m-1) = 2m-1.$$

As the quadratic model has $p = 2m - 1$ parameters, the equality in (2.1.4) holds for any $\boldsymbol{b} \in \mathcal{X}$. By the equivalence theorem, every COA(n, m) is D-optimal for the quadratic model. The proof of D-optimality for the first-order model is simpler. This completes the proof. $\square$

Proof of Theorem 3.3. For the full design $\mathbf{F}_m$ and the second-order position model (3.1.5), let $\boldsymbol{X}$ be the $n \times p$ model matrix and $\boldsymbol{M} = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}/n$ be the $p \times p$ information matrix with $n = m!$ and $p = (m-1)(m+2)/2$. Let $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}$ be the hat matrix. To prove the D-optimality, we need to show that the equality in (2.1.4) holds for any $\boldsymbol{b} \in \mathcal{X}$. For any $\boldsymbol{b} \in \mathcal{X}$, by the standard linear model theory, the variance of the fitted value when $\boldsymbol{x} = \boldsymbol{b}$ is $Var(\hat{y}(\boldsymbol{x})) = \sigma^2 \boldsymbol{f}(\boldsymbol{b})^{\mathrm{T}}(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{f}(\boldsymbol{b}) = n^{-1}\sigma^2\boldsymbol{f}(\boldsymbol{b})^{\mathrm{T}}\boldsymbol{M}^{-1}\boldsymbol{f}(\boldsymbol{b})$. Because every $\boldsymbol{b}$ is a row of the full design $\mathbf{F}_m$, it is sufficient to show that each of the diagonal elements of the hat matrix $\boldsymbol{H}$ is p/n.

To do this, we consider the extended second-order model

$$y = \beta_0 + \sum_{k=1}^{m} p_1(b_k)\beta_k + \sum_{k=1}^{m} p_2(b_k)\beta_{kk} + \sum_{1\le k<l\le m} p_1(b_k)p_1(b_l)\beta_{kl} + \varepsilon, \tag{A.0.1}$$

which includes all m first-order, m pure quadratic and $m(m-1)/2$ bilinear (or interaction) terms. The extended second-order model has $q = (m+1)(m+2)/2$ parameters. Let $\boldsymbol{Z}$ be the $n \times q$ model matrix for the full design $\mathbf{F}_m$. Due to the constraints on the orthogonal polynomials and the fact that each row is a permutation, $\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z}$ has rank p and its inverse does not exist, so we consider its Moore-Penrose generalized inverse $(\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z})^{-}$. By the standard linear model theory (Seber and Lee, 2003), the projection matrix $\boldsymbol{P} = \boldsymbol{Z}(\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z})^{-}\boldsymbol{Z}^{\mathrm{T}}$ of the extended second-order model (A.0.1) is identical to the hat matrix $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}$ of the second-order position model (3.1.5) because columns of $\boldsymbol{Z}$ and $\boldsymbol{X}$ span the same linear space. Under the extended model (A.0.1), all variables are exchangeable; therefore, the variances of the fitted values are the same for all rows of the full design. This is equivalent to saying that the diagonal elements of projection matrix $\boldsymbol{P}$ are the same. Since $\boldsymbol{P}$ is idempotent and has rank p, its trace is equal to its rank. Therefore, all of the diagonal elements of $\boldsymbol{P}$, and hence $\boldsymbol{H}$, are equal to p/n. This completes the proof. $\square$

Proof of Theorem 4.1. For a fixed m and q the normalized information matrix and its inverse under the full design $\mathbf{S}_{m,q}$ are given by

$$\boldsymbol{M}(\boldsymbol{S_{m,q}}) = \begin{bmatrix} 1 & \boldsymbol{a}^{\mathrm{T}} & \boldsymbol{a}^{\mathrm{T}} & \dots & \boldsymbol{a}^{\mathrm{T}} \\ \boldsymbol{a} & \boldsymbol{B} & \boldsymbol{C} & \dots & \boldsymbol{C} \\ \boldsymbol{a} & \boldsymbol{C} & \boldsymbol{B} & \dots & \boldsymbol{C} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{a} & \boldsymbol{C} & \boldsymbol{C} & \dots & \boldsymbol{B} \end{bmatrix} \qquad \boldsymbol{M}^{-1}(\boldsymbol{S_{m,q}}) = \begin{bmatrix} d & \boldsymbol{e}^{\mathrm{T}} & \boldsymbol{e}^{\mathrm{T}} & \dots & \boldsymbol{e}^{\mathrm{T}} \\ \boldsymbol{e} & \boldsymbol{F} & \boldsymbol{G} & \dots & \boldsymbol{G} \\ \boldsymbol{e} & \boldsymbol{G} & \boldsymbol{F} & \dots & \boldsymbol{G} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{e} & \boldsymbol{G} & \boldsymbol{G} & \dots & \boldsymbol{F} \end{bmatrix}$$

,

$$\begin{aligned} \boldsymbol{a} &= \frac{1}{m}\mathbf{1}_{(m-1)} & d &= \frac{m(1-2q+mq)}{m-q} \\ \boldsymbol{B} &= \frac{1}{m}\boldsymbol{I}_{(m-1)} & \boldsymbol{e} &= -\frac{m(m-1)}{(m-q)}\mathbf{1}_{m-1} \\ \boldsymbol{C} &= \alpha\boldsymbol{J}_{(m-1)} - \alpha\boldsymbol{I}_{(m-1)} & \boldsymbol{F} &= \delta\boldsymbol{J}_{(m-1)} + \delta\boldsymbol{I}_{(m-1)} \\ \alpha &= \frac{1}{m(m-1)}, \delta = \frac{(m-1)(m-q+1)}{m-q}, \zeta = \frac{m-1}{m-q} & \boldsymbol{G} &= \zeta\boldsymbol{J}_{(m-1)} + \zeta\boldsymbol{I}_{(m-1)}, \end{aligned}$$

where $\boldsymbol{I}_m$ is the $m \times m$ identity matrix, $\boldsymbol{J}_m$ is an $m \times m$ matrix of 1's, and $\mathbf{1}_m$ is a length m column vector of 1's.

Using the equivalence theorem, for row i of the model matrix:

$$\boldsymbol{X}_i\boldsymbol{M}^{-1}(\boldsymbol{S_{m,q}})\boldsymbol{X}_i^{\mathrm{T}} - ((m-1)q+1) = d + w_i\Big(-2m\zeta + \delta + (w_i - 1)\zeta\Big) - mq + q - 1,$$

where $w_i = q$ if run i does not include component m and $w_i = q - 1$ otherwise.

After some algebra this expression simplifies to the following:

$$\boldsymbol{X}_i\boldsymbol{M}^{-1}(\boldsymbol{S_{m,q}})\boldsymbol{X}_i^{\mathrm{T}} - ((m-1)q+1) = -mq + w_i m - 2mqw_i + 2qw_i - w_i + w_i^2 m - w_i^2 + mq^2 - q^2 + q$$

For both settings of w_i, this expression evaluates to 0, satisfying the equivalence theorem. $\square$

Proof of Theorem 4.2. For a fixed m and q the normalized information matrix and its inverse under the full design $\mathbf{S}_{m,q}$ are given by

$$\boldsymbol{M}(\boldsymbol{S_{m,q}}) = \begin{bmatrix} 1 & \mathbf{0}_{m-1}^{\mathrm{T}} & \mathbf{0}_{m-2}^{\mathrm{T}} & \cdots & \mathbf{0}_{2}^{\mathrm{T}} & 0 \\ \mathbf{0}_{m-1} & \boldsymbol{A}_{m-1} & \boldsymbol{B}_{m-2,m-1}^{\mathrm{T}} & \cdots & \boldsymbol{B}_{2,m-1}^{\mathrm{T}} & \boldsymbol{B}_{1,m-1}^{\mathrm{T}} \\ \mathbf{0}_{m-2} & \boldsymbol{B}_{m-2,m-1} & \boldsymbol{A}_{m-2} & \cdots & \boldsymbol{B}_{2,m-2}^{\mathrm{T}} & \boldsymbol{B}_{1,m-2}^{\mathrm{T}} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0}_{2} & \boldsymbol{B}_{2,m-1} & \boldsymbol{B}_{2,m-2} & \cdots & \boldsymbol{A}_{2} & \boldsymbol{B}_{1,2}^{\mathrm{T}} \\ 0 & \boldsymbol{B}_{1,m-1} & \boldsymbol{B}_{1,m-2} & \cdots & \boldsymbol{B}_{1,2} & \boldsymbol{A}_{1} \end{bmatrix}$$

$$\boldsymbol{M}^{-1}(\boldsymbol{S_{m,q}}) = \begin{bmatrix} 1 & \mathbf{0}_{m-1}^{\mathrm{T}} & \mathbf{0}_{m-2}^{\mathrm{T}} & \cdots & \mathbf{0}_{2}^{\mathrm{T}} & 0 \\ \mathbf{0}_{m-1} & \boldsymbol{C}_{m-1} & \boldsymbol{D}_{m-2,m-1}^{\mathrm{T}} & \cdots & \boldsymbol{D}_{2,m-1}^{\mathrm{T}} & \boldsymbol{D}_{1,m-1}^{\mathrm{T}} \\ \mathbf{0}_{m-2} & \boldsymbol{D}_{m-2,m-1} & \boldsymbol{C}_{m-2} & \cdots & \boldsymbol{D}_{2,m-2}^{\mathrm{T}} & \boldsymbol{D}_{1,m-2}^{\mathrm{T}} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0}_{2} & \boldsymbol{D}_{2,m-1} & \boldsymbol{D}_{2,m-2} & \cdots & \boldsymbol{C}_{2} & \boldsymbol{D}_{1,2}^{\mathrm{T}} \\ 0 & \boldsymbol{D}_{1,m-1} & \boldsymbol{D}_{1,m-2} & \cdots & \boldsymbol{D}_{1,2} & \boldsymbol{C}_{1} \end{bmatrix}$$

,

$$\begin{aligned} \boldsymbol{A}_i &= \alpha(\boldsymbol{J}_m - \boldsymbol{I}_m) + \delta\boldsymbol{I}_m & \boldsymbol{C}_i &= \zeta(\boldsymbol{J}_m - \boldsymbol{I}_m) + \eta\boldsymbol{I}_m \\ \boldsymbol{B}_{i,j} &= [\mathbf{0}_{(i\times j-i-1)} \vdots -\alpha\mathbf{1}_i \vdots \alpha\boldsymbol{I}_i] & \boldsymbol{D}_{i,j} &= [\mathbf{0}_{(i\times j-i-1)} \vdots -\zeta\mathbf{1}_i \vdots \zeta\boldsymbol{I}_i] \\ \alpha &= \frac{\sum\limits_{j=1}^{q-1} j(j-1)}{m(m-1)(m-2)} & \delta &= \frac{q(q-1)}{m(m-1)} \\ \zeta &= \frac{-\alpha}{(\delta-2\alpha)(\delta+(m-2)\alpha)} & \eta &= \frac{\delta+(m-4)\alpha}{(\delta-2\alpha)(\delta+(m-2)\alpha)}, \end{aligned}$$

where $\boldsymbol{I}_m$ is the $m \times m$ identity matrix, $\boldsymbol{J}_m$ is an $m \times m$ matrix of 1's, $\mathbf{0}_m$ is a length m column vector of 0's, $\mathbf{1}_m$ is a length m column vector of 1's and $\mathbf{0}_{i\times j}$ is an $i \times j$ matrix of 0's.

Using the equivalence theorem, for any row i of the model matrix:

$$\boldsymbol{X}_i\boldsymbol{M}^{-1}(\boldsymbol{S_{m,q}})\boldsymbol{X}_i^{\mathrm{T}} - \binom{m}{2} - 1 = 1 + \frac{1}{2}q(q-1)\eta + g\zeta - \binom{m}{2} - 1,$$

where $g = \sum_{j=1}^{q-1} j(j-1) = q(q-1)(q-2)/3$. After some algebra this expression simplifies to

$$\begin{aligned}
&= q(q-1)\Big(q(q-1) + \frac{(m-4)g}{m-2}\Big) - 2g\Big(\frac{g}{m-2}\Big) - \Big(q(q-1) - \frac{2g}{m-2}\Big)\Big(q(q-1)+g\Big) \\
&= \frac{q(q-1)(m-4)g}{m-2} - gq(q-1) + \frac{2gq(q-1)}{m-2} \\
&= gq(q-1)\Big(\frac{m-4}{m-2} - 1 + \frac{2}{m-2}\Big) \\
&= 0
\end{aligned}$$

□

Proof of Theorem 4.3. Let m be prime or a prime power, $1 < q < m$, and $n = \lambda m(m-1)$ where $\lambda \geq 1$. The design $\mathbf{S^c}_{n,m,q}$ inherits the properties of the COA(n,m) proposed by Yang et al. (2021). This includes the property that every two-column sub-array contains every level combination (i,j) with $i \neq j$ and $i,j = 1,2,\ldots,m$ the same number of times. From this property we know that before normalization the information matrix of the design $\mathbf{S^c}_{n,m,q}$ is given by

$$n\boldsymbol{M}(\boldsymbol{S^c_{n,m,q}}) = \begin{bmatrix} \lambda m(m-1) & \lambda(m-1)\mathbf{1}^{\mathrm{T}}_{\boldsymbol{(m-1)}} & \ldots & \lambda(m-1)\mathbf{1}^{\mathrm{T}}_{\boldsymbol{(m-1)}} \\ \lambda(m-1)\mathbf{1}_{\boldsymbol{(m-1)}} & \lambda(m-1)\boldsymbol{I}_{(m-1)} & \ldots & \lambda\big(\boldsymbol{J}_{(m-1)} - \boldsymbol{I}_{(m-1)}\big) \\ \lambda(m-1)\mathbf{1}_{\boldsymbol{(m-1)}} & \lambda\big(\boldsymbol{J}_{(m-1)} - \boldsymbol{I}_{(m-1)}\big) & \ldots & \lambda\big(\boldsymbol{J}_{(m-1)} - \boldsymbol{I}_{(m-1)}\big) \\ \vdots & \vdots & \ddots & \vdots \\ \lambda(m-1)\mathbf{1}_{\boldsymbol{(m-1)}} & \lambda\big(\boldsymbol{J}_{(m-1)} - \boldsymbol{I}_{(m-1)}\big) & \ldots & \lambda(m-1)\boldsymbol{I}_{(m-1)} \end{bmatrix}$$

Dividing this matrix by n yields the same information matrix as the full design $\mathbf{S}_{m,q}$ given in Theorem 4.1. □

Proof of Theorem 4.4. (i) Assume that there is some $n < 2\binom{m}{2}$ such that there is a design with n runs for which the normalized information matrix is equal to the one given in the proof of Theorem 4.1. For such a design the raw information matrix is given by

$$n\boldsymbol{M}(\boldsymbol{S_{m,2}}) = \begin{bmatrix} n & \mathbf{0}^{\mathrm{T}}_{\binom{m}{2}} \\ \mathbf{0}_{\binom{m}{2}} & \frac{2n}{m(m-1)}\boldsymbol{I}_{\binom{m}{2}} \end{bmatrix}.$$

In this matrix $n\delta = \frac{2n}{m(m-1)}$ represents the number of occurrences of each pair of components (i, j). This number must be even so that each pair can occur as i, j and j, i an equal number of times and produce a value of 0 in the first row and column of the matrix. Every pair must also occur an equal number of times to all other pairs. We know that there is at least one pair of components that appears fewer than 2 times. If a pair only appears once, then balance is not possible, and if it appears 0 times, then it is not possible to have equal values on the diagonal, so no such n exists.

(ii) Assume that there is some $n < 6\binom{m}{3}$ such that there is a design with n runs for which the normalized information matrix is equal to the one given in the proof of Theorem 4.1. Following the proof of (i) each pair of components must occur the same number of times, specifically $n\delta = \frac{6n}{m(m-1)}$ times, which must be even. Similarly, each pair of pairs with one overlapping component must occur $n\alpha = \frac{2n}{m(m-1)(m-2)}$ times, which must be an integer. The only value of n for which $n\alpha$ is an integer is $n = 3\binom{m}{3}$, yielding $n\alpha = 1$. However, this value of n yields $n\delta = 3(m-2)$. Since m is odd, $n\delta$ cannot be even.

(iii) Following the proof of (ii), in the case that m is even, then $n = 3\binom{m}{3}$ is the only $n < 6\binom{m}{3}$ which yields an integer value for $n\alpha$ and an even value for $n\delta = 3(m-2)$. □

Proof of Theorem 4.5. With $n = 3\binom{m}{3}$ and even m, and using the information matrix given in the proof of Theorem 4.1, the design $\mathbf{S^P}_{n,m,3}$ is optimal for the CPS model if each component occurs in each position the same number of times, specifically $(m-1)(m-2)/2$ times. Likewise, each pair of components (i, j) must occur in each 2-column sub-array equally often, specifically $(m-2)/2$ times.

The design $\mathbf{S^P}_{n,m,3}$ is made up of $\binom{m}{3}$, $\boldsymbol{D}_{ijk}$ matrices with $0 \le i < j < k \le m-1$, half of which have $i+j+k$ even and half with $i+j+k$ odd. Since each $\boldsymbol{D}_{ijk}$ is a Latin square, each component occurs in each position once per square, meaning that each component occurs in each position $\binom{m-1}{2} = (m-1)(m-2)/2$ times.

For a fixed ordered pair of components (x, y) with $x < y$, consider the ordered set of $m-2$ elements $\boldsymbol{z} = \{j = 0, \ldots, m-1; j \neq x, y\}$. Consider a pair of triplets $\{x, y, \boldsymbol{z}[j]\}$ and $\{x, y, \boldsymbol{z}[j+1]\}$. There are $(m-2)/2-1$ such distinct pairs with no overlapping triplets. Each triplet $\{x, y, \boldsymbol{z}[j]\}$ admits a matrix $\boldsymbol{D}_{\boldsymbol{z}[j]xy}$, $\boldsymbol{D}_{x\boldsymbol{z}[j]y}$ or $\boldsymbol{D}_{xy\boldsymbol{z}[j]}$. $x+y+\boldsymbol{z}[j]$ and $x+y+\boldsymbol{z}[j+1]$ have opposite parity if one of the following is true:

i) $\boldsymbol{z}[j] < x < y$ and $\boldsymbol{z}[j+1] < x < y$

ii) $x < \boldsymbol{z}[j] < y$ and $x < \boldsymbol{z}[j+1] < y$

iii) $x < y < \boldsymbol{z}[j]$ and $x < y < \boldsymbol{z}[j+1]$

iv) $\boldsymbol{z}[j] < x < y$ and $x < y < \boldsymbol{z}[j+1]$

Otherwise the parity of both triplets is the same if one of the following is true:

i) $\boldsymbol{z}[j] < x < y$ and $x < \boldsymbol{z}[j+1] < y$

ii) $x < \boldsymbol{z}[j] < y$ and $x < y < \boldsymbol{z}[j+1]$

In all cases, the six runs generated from concatenating the D matrices produced from the two triplets yields the following 7×7 partition of the raw information matrix that includes only the intercept and terms that feature component x or y.

$$\begin{array}{c|ccc|} & 1 & z_{x\cdot} & z_{y\cdot} \\ \hline 1 & 6 & \mathbf{2}_3^{\mathrm{T}} & \mathbf{2}_3^{\mathrm{T}} \\ z_{x\cdot} & \mathbf{2}_3 & 2\boldsymbol{I}_3 & \boldsymbol{J}_3 - \boldsymbol{I}_3 \\ z_{y\cdot} & \mathbf{2}_3 & \boldsymbol{J}_3 - \boldsymbol{I}_3 & 2\boldsymbol{I} \end{array}$$

Since this holds for each of the $(m-2)/2-1$ pairs of triplets and for all pairs of components (x,y) we conclude that each pair of components shows up in each pair of positions an equal number of times, specifically $(m-2)/2-1$.

For the PWOS model, optimality requires that each pair of components (x, y) occur as both (x, y) and (y, x) an equal number of times. In proving the optimality of the design for the CPS model we have already seen that this is true for each pair of positions and thus true for the design overall.

Additionally, we know that each pair of pairs of components with exactly one component overlapping is present in exactly 3 runs. For components $x < y < z$ this can take 3 forms: $(xy)(xz)$, $(xy)(yz)$ and $(xz)(yz)$. Regardless of the parity of $x + y + z$, the 3 runs of $\boldsymbol{D}_{xyz}$ admits the following partition of the raw information matrix that includes only the terms that involve the three components:

$$\begin{array}{c} \\ I_{xy} \\ I_{xz} \\ I_{yz} \end{array} \begin{array}{c} \begin{array}{ccc} I_{xy} & I_{xz} & I_{yz} \end{array} \\ \begin{bmatrix} 3 & 1 & -1 \\ 1 & 3 & 1 \\ -1 & 1 & 3 \end{bmatrix} \end{array}$$

Summing these information matrices across all triplets recovers the information matrix given in Theorem 4.2 for the specific case that $q = 3$, where $\alpha = 2/m(m-1)(m-2)$ and $\delta = 6/m(m-1)$.

$\square$

Proof of Theorem 4.6. For each pair of components $x < y$, we generate a set of $\binom{m-2}{q-2}$ OofA-OAs. Concatenating these designs preserves the properties of each pair of pseudo-factors given in Section 2.3, except that now the inner product of the factor with itself is $\pm \binom{m-2}{q-2}\ n$, where $n = 12\lceil(\binom{q}{2} +1)/12\rceil$.

For each triplet $x < y < z$, we generate a set of $\binom{m-3}{q-3}$ OofA-OAs. Concatenating these designs preserves the properties of each pair of pseudo-factors given in Section 2.3, except that now the inner product of the factors is $\pm \binom{m-3}{q-3}\ n/3$, where $n = 12\lceil(\binom{q}{2} +1)/12\rceil$.

Dividing each of these values by the run size of the whole design $\binom{m}{q}\ n$ yields the correct values for α and δ to reproduce the information matrix given in the proof of Theorem 4.2. $\square$

APPENDIX B

Additional Results

B.1 Additional Permutation Robustness Results

For each sample size n in Figure 3.2, up to five unique designs are necessary to obtain the largest D-efficiency under each model. Instead, we could consider a single design $\mathbf{F}^*_{n,m}$ for each combination of n and m that is derived from maximizing the geometric mean efficiency of the estimable models, akin to those presented in Table 3.6. The results of this analysis are presented in Figure B.1. Generally, we see that many of the efficiencies are on par with what we observed when maximizing the value for each model individually. A notable exception is the efficiency of the maximal geometric mean design under the PWO model, which for some combinations of n and m is slightly lower than the efficiency of the design that solely optimizes performance for the PWO model; see Figure 3.2.

While Figures 3.2 and B.1 substantiate our claim that our algorithm produces efficient designs, they do not inherently show how much there is to gain or lose by selection of permutations in Algorithm 3.1. They also do not consider the effect of level or $\boldsymbol{C}_i$ permutations. To remedy this, Table B.1 gives the maximal D_{PWO} and D_{SO} values obtained through a brute force search over all choices of the three permutations and the improvement relative to the values of the $\mathbf{F}_{n,m}$ designs in Table 3.6. We do not include the other efficiencies since many of the designs are optimal under the CP, first-order and quadratic models and show minimal improvement with column permutations. The notation $\mathbf{F}^+_{n,m}$ is used to represent the resulting designs. Δ_{PWO} and Δ_{SO} give the difference in D-efficiency under the specified

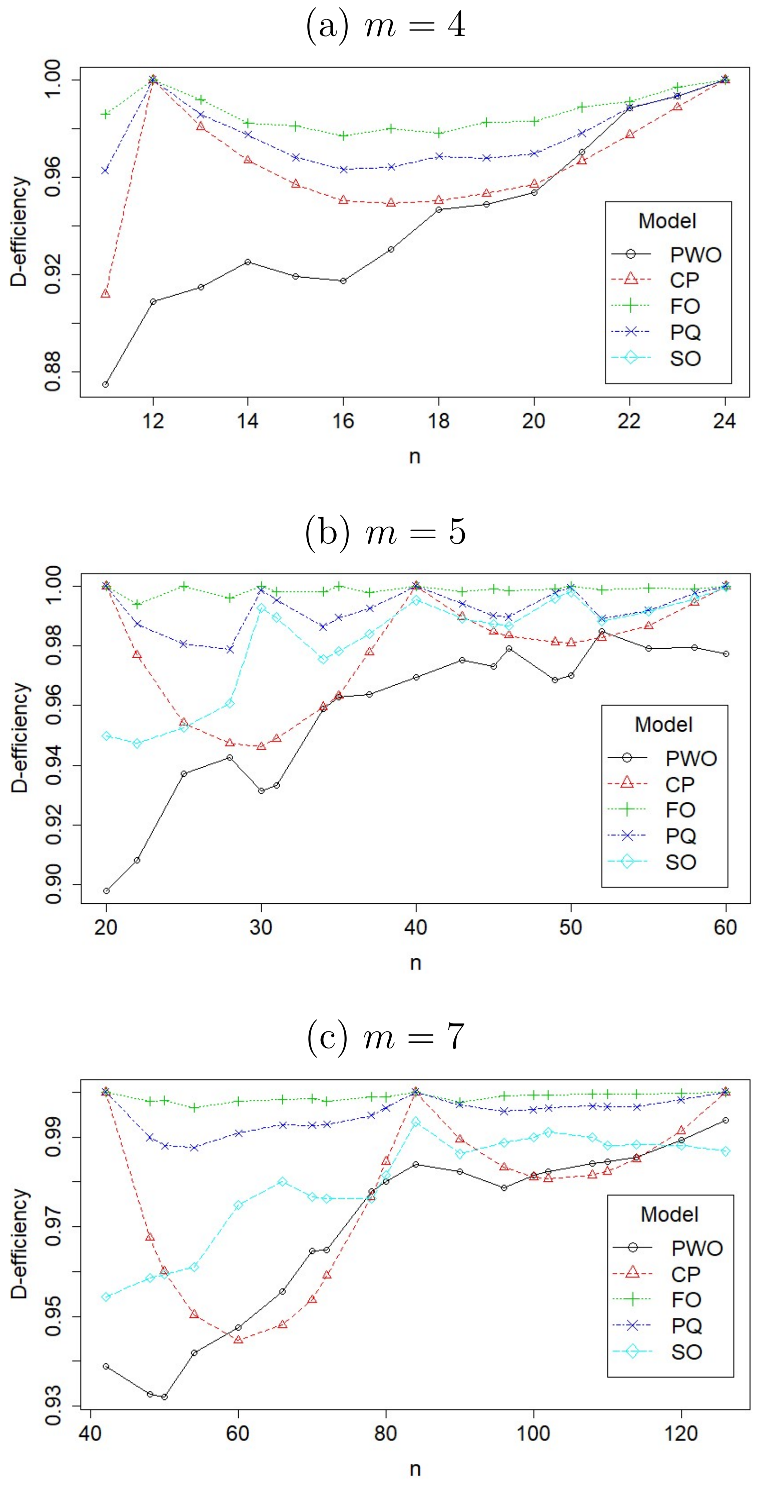


Figure B.1: The D-efficiency of $\mathbf{F}^*_{n,m}$, which maximizes the geometric mean efficiency for variable run sizes for (a) $m = 4$, (b) $m = 5$, and (c) $m = 7$.

model between the best permuted design and the design $\mathbf{F}_{n,m}$ given in Table 3.6. Cases for which the same set of permutations generate the best design for both models are indicated by †.

Table B.1: Maximal D-efficiencies of designs for the PWO and second-order models under permutation.

			D-efficiency		Change	
n	m	Design	D_{PWO}	D_{SO}	Δ_{PWO}	Δ_{SO}
†12	4	$\mathbf{F}^{+}_{12,4}$	0.909	1	0	0
†16	4	$\mathbf{F}^{+}_{16,4}$	0.917	0.953	0	0
†20	4	$\mathbf{F}^{+}_{20,4}$	0.954	0.961	0	0
20	5	$\mathbf{F}^{+}_{20,5}$	0.898	0.959	0.898	0
24	5	$\mathbf{F}^{+}_{24,5}$	0.926	0.961	0.381	0.012
40	5	$\mathbf{F}^{+}_{40,5}$	0.969	0.999	0.08	0
†60	5	$\mathbf{F}^{+}_{60,5}$	0.977	0.999	0	0.013

There are several interesting observations we can make from Table B.1. First, as expected, we see that the D_{PWO} values of our designs in situations that were previously troubling greatly improve with this manipulation, closing the gap between our designs and Voelkel's. We also see minor improvements in the D_{SO} values for some cases with $m = 5$. Most importantly, considering all three types of permutations did not lead to designs that substantially outperform the $\mathbf{F}^{*}_{n,m}$ designs that only considered column permutations.

We are also interested in knowing the worst efficiency attainable under our algorithm. Table B.2 summarizes this effect in the same manner as before, this time using $\mathbf{F}^{-}_{n,m}$ to denote the worst design. In this case we see that compared to the small gains in D_{SO} of Table B.1, the loss of efficiency due to poor selection of permutations is relatively large when $m = 5$. On the other hand, the minimal value of D_{PWO} is often not a substantial decrease from the values found without permutations. For larger m, the brute force approach is limited by the same combinatorial explosion that motivates order-of-addition designs.

Table B.2: Minimal D-efficiencies of designs for the PWO and second-order models under permutation.

			D-efficiency		Change	
n	m	Design	D_{PWO}	D_{SO}	Δ_{PWO}	Δ_{SO}
†12	4	$\mathbf{F}^{-}_{12,4}$	0.909	1	0	0
†16	4	$\mathbf{F}^{-}_{16,4}$	0.917	0.953	0	0
†20	4	$\mathbf{F}^{-}_{20,4}$	0.954	0.961	0	0
20	5	$\mathbf{F}^{-}_{20,5}$	0	0.428	0	−0.531
24	5	$\mathbf{F}^{-}_{24,5}$	0.545	0.581	0	−0.368
†40	5	$\mathbf{F}^{-}_{40,5}$	0.784	0.735	−0.105	−0.264
†60	5	$\mathbf{F}^{-}_{60,5}$	0.861	0.875	−0.116	−0.111

B.2 Additional Model Misspecification Results

We fix $(n, m) = (12, 4)$ and consider the PWO and CP models. We define our confidence that the PWO model is indeed the true model as $\alpha \in [0, 1]$ and similarly our confidence in the CP model as $1 - \alpha$. We then use Differential Evolution (Storn, 1996) to find 12-run designs for various values of α that maximize the desirability function given by

$$\bar{G}^{(\alpha)}(\boldsymbol{\xi}) = D^{\alpha}_{PWO}(\boldsymbol{\xi}) D^{1-\alpha}_{CP}(\boldsymbol{\xi}). \tag{B.2.1}$$

Differential Evolution is inspired by principles of natural selection, mutation and genetic crossover and has been shown to work well for finding optimal designs while only depending on the choice of a few parameters (Price et al., 2005). A full description of Differential Evolution is given in Section 5.1. After tuning the parameters, we are able to use this algorithm to quickly locate the maximum for all $\alpha = 0, 0.1, \ldots, 1$.

Figure B.2 shows the unweighted efficiencies of the designs found by the search. In this plot, α gives our confidence in the PWO model. When $\alpha = 0$, we assume that the data follow the CP model with high confidence ($1 - \alpha = 1$). In this case the algorithm finds a design that is isomorphic to $\mathbf{F}_{12,4}$ from Algorithm 3.1, with the efficiencies matching our results from Table 3.6. Designs with this property are represented in the plot by the "F" symbol. As we then increase α and split our confidence between this model and the PWO model, $\mathbf{F}_{12,4}$ continues to have maximal $\bar{G}^{(\alpha)}$. In fact, it is not until we increase α from 0.7 to 0.8 that this changes. For $\alpha \geq 0.8$ the algorithm finds a design equivalent to Voelkel.12a. These designs are denoted with the "V" symbol.

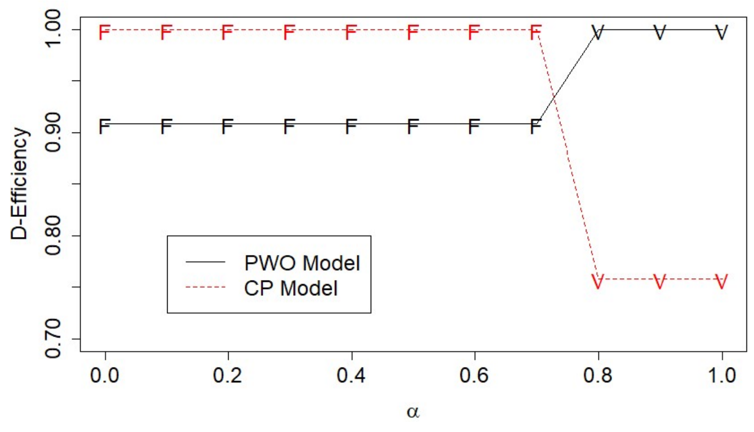


Figure B.2: D-efficiencies of designs that maximize (B.2.1).

This result demonstrates that our designs are indeed robust to model misspecification in this case. In addition to the model's form, there is also the underlying assumption made by each of these models as to whether the relative positions or absolute positions are important in determining the response. By demonstrating that our designs are robust to model misspecification under this pair of models, we have also shown that they are robust to this assumption.

Bibliography

Ailon, N. (2012). An Active Learning Algorithm for Ranking from Pairwise Preferences with an Almost Optimal Query Complexity. *Journal of Machine Learning Research* **13**, 137–164.

Anderson-Cook, C. M. and Lu, L. (2019). Discussion on "Order-of-Addition Experiments: A review and some new thoughts". *Quality Engineering* **31**, 64–68.

Atkinson, A. (1996). The Usefulness of Optimum Experimental Designs. *Journal of the Royal Statistical Society Series B* **58**(1), 59–76.

Baptista, R. and Poloczek, M. (2018). Bayesian Optimization of Combinatorial Structures.*Proceedings of the 35th International Conference on Machine Learning* **80**, 462–471.

Barker, H. A. (1986). Sum and product tables for Galois fields. *International Journal of Mathematical Education in Science and Technology* **17**, 473–485.

Basu, D. (1980) Randomization Analysis of Experimental Data: The Fisher Randomization Test. *Journal of the American Statistical Association* **75**(371), 575–582.

Birattari, M. (2009). *Tuning Metaheuristics*. Springer.

Blum, C. and Roli, A. (2001). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys* **35**, 268–308.

Bose, M. and Dey, A. (2015). Crossover designs. In *Handbook of Design and Analysis of Experiments*, Chapman and Hall/CRC, 159–195.

Box, G. E. P. and Draper, N. R. (2007). *Response Surfaces, Mixtures, and Ridge Analyses.* John Wiley & Sons, 2nd edition.

Buchheim, C. and Jünger, M. (2005). Linear Optimization over Permutation Groups. *Discrete Optimization* **2**(4), 308–319.

Chakraborty, U. K. (2008). *Advances in Differential Evolution*. Springer.

Chen, M. R., Li, X., Zhang, X. and Lu, Y. Z. (2010). A novel particle swarm optimizer hybridized with extremal optimization. *Applied Soft Computing* **10**(2), 367–373.

Chen, J., Mukerjee, R. and Lin, D. K. J. (2020). Construction of optimal fractional Order-of-Addition designs via block designs. *Statistics and Probability Letters* **161**(2020), 108728.

Chen, R. B., Wang, W. C., Chang, S. P. and Wong, W. K. (2015). Minimax Optimal Designs via Particle Swarm Optimization Methods. *Statistics and Computing* **25**, 975–988.

Cheng, C.-S., Deng, L.-Y., and Tang, B. (2002). Generalized minimum aberration and design efficiency for nonregular fractional factorial designs. *Statistica Sinica* **12**, 991–1000.

Cheng, C. S., Martin, R. J. and Tang, B. (1998). Two-level factorial designs with extreme numbers of level changes. *The Annals of Statistics* **26**(4), 1522–1539.

Cornell, J. A. (2011). *Experiments with mixtures: designs, models, and the analysis of mixture data*. John Wiley & Sons, 3rd edition.

Dawar. D. and Ludwig, S. A. (2014). Differential evolution with dither and annealed scale factor. *2014 IEEE Symposium on Differential Evolution*,1–8. Orlando, Florida.

Ding, X., Matsuo, K., Xu, L., Yang, J. and Zheng, L. (2015). Optimized combinations of bortezomib, camptothecin, and doxorubicin show increased efficacy and reduced toxicity in treating oral cancer. *Anti-Cancer Drugs* **26**, 547–554.

Ding, X., Xu, H., Hopper, C., Yang, J., and Ho, C.-M. (2013). Use of fractional factorial designs in antiviral drug studies. *Quality and Reliability Engineering International* **29**, 299–304.

Eberhart, R. C. and Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. *Proceedings of the 2000 Congress on evolutionary computation* **1**, 84–88.

Elbes, M., Alzubi, S., Kanan, T., Al-Fuqaha, A. and Hawashin, B. (2019). A survey on particle swarm optimization with emphasis on engineering and network applications. *Evolutionary Intelligence* **12**, 113-129.

Emami, P. and Ranka, S. (2018). Learning Permutations with Sinkhorn Policy Gradient. *ArXiv*, abs/1805.07010.

Fang, K.-T., Li, R. and Sudjianto, A. (2006). *Design and Modeling for Computer Experiments*, New York: Chapman and Hall/CRC.

Fedorov, V. V. (1972). *Theory of Optimal Designs*, Academic Press.

Feoktistov, V., Pietravalle, S. and Heslot, N. (2017). Optimal experimental design of field trials using Differential Evolution. *2017 IEEE Congress on Evolutionary Computation*, 1690–1696. San Sebastian.

Fisher, R. A. *The Design of Experiments*, 9th edition. Macmillan Publishing Company, New York.

Fries, A. and Hunter, W. G. (1980). Minimum aberration 2^{k-p} designs. *Technometrics* **22**, 601–608.

Garey, M.R., Johnson, D.S., and Sethi, R. (1976). The complexity of jobshop and flowshop scheduling. *Mathematics of Operations Research* **1**(2), 117–129.

Gramacy, R. B. (2021). *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*, Chapman Hall/CRC.

Grant, D. (2017). The ballot order effect is huge: evidence from Texas. *Public Choice* **172**, 421–442.

Groemping, U., Amarov, B., and Xu, H. (2014). DoE.base: Full factorials, orthogonal arrays and base utilities for DoE packages. Package version 0.25-3.

He, Y., Ma, W. J., and Zhang, J. P. (2016). The Parameters Selection of PSO Algorithm influencing On performance of Fault Diagnosis. *MATEC Web of Conferences* **63**, 02019.

Huang, H. (2021). Construction of component orthogonal arrays with any number of components. *Journal of Statistical Planning and Inference* **213**, 72–79.

Hyun, S. W. and Wong W. K. (2015). Multiple-Objective Optimal Designs for Studying the Dose Response Function and Interesting Dose Levels.*International Journal of Biostatistics* **11**, 253–271.

Jaynes, J., Ding, X., Xu, H., Wong, W. K., and Ho, C.-M. (2013). Application of fractional factorial designs to study drug combinations. *Statistics in Medicine* **32**, 307–318.

Johnson, M. E., Moore, L. M. and Ylvisaker, D. (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference* **26**, 131–148.

Jones, B. and Kenward, M. G. (2015). *Design and Analysis of Cross-Over Trials*, 3rd ed. New York: Chapman & Hall/CRC.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions, *Journal of Global optimization* **13**, 455–492.

Jourdain, L. S., Schmitt, C., Leser, M. E., Murray, B. S. and Dickinson, E. (2009). Mixed Layers of Sodium Caseinate + Dextran Sulfate: Influence of Order of Addition to OilWater Interface. *Langmuir* **25**(17), 10026-10037.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. *4th Proceedings of International Conference on Neural Networks*, 1942–1948. Perth, WA, Australia.

Kiefer, J. C. (1959). Optimum experimental designs. *Journal of the Royal Statistical Society Series B* **21**, 272–319.

Kiefer, J. C. (1961). Optimum designs in regression problems, II. *Annals of Mathematical Statistics* **32**, 298–325.

Kumar, S. and Jadon, P. (2014). A Novel Hybrid Algorithm for Permutation Flow Shop Scheduling. *International Journal of Computer Science and Information Technologies* **5**(4), 5057–5061.

Lin, D. K. J. and Peng, J. (2019). Order-of-addition experiments: A review and some new thoughts. *Quality Engineering* **31**(1), 49–59.

Lukemire, J., Mandal, A. and Wong, W. K. (2019). d- QPSO: A Quantum-Behaved Particle Swarm Technique for Finding D-Optimal Designs With Discrete and Continuous Factors and a Binary Response. *Technometrics* **61**(1), 77–87.

MacBeath, G., and Yaffe, M. B. (2012). Sequential application of anticancer drugs enhances cell death by rewiring apoptotic signaling networks. *Cell* **149**(4), 780–794.

Mandal, A. and Mukerjee, R. (2005). Design efficiency under model uncertainty for nonregular fractions of general factorials.*Statistica Sinica* **15**, 697–707.

Mee, R. W. (2020). Order of addition modeling. *Statistica Sinica* **30**, 1543–1559.

Mena, G., Belanger, D., Linderman, S. and Snoek, J. (2018). Learning Latent Permutations with Gumbel-Sinkhorn Networks. *Int. Conference on Learning Representations* **6**.

Mitliagkas, I., Gopalan, A., Caramanis, C. and Vishwanath, S. (2011). User rankings from comparisons: Learning permutations in high dimensions. *49th Annual Allerton Conference on Communication, Control, and Computing*, 1143–1150. Monticello, IL.

Olsen, G. J., Matsuda, H., Hagstrom, R., Overbeek R. (1994). fastDNAmL: a tool for construction of phylogenetic trees of DNA sequences using maximum likelihood.*Computer Applications in the Biosciences* **10**(1), 41-48.

Paredes-García, W.J. and Castaño-Tostado, E. (2017). Differential evolutionary algorithm in the construction process of optimal experimental designs, *Communications in Statistics - Simulation and Computation* **46**(10), 7733–7743.

Peng, J. Y., Mukerjee, R. and Lin, D. K. J. (2019). Design of order-of-addition experiments. *Biometrika* **106**(3), 683–694.

Phoa, F. K. H., Chen, R. B., Wang, W. C. and Wong, W. K. (2016). Optimizing Two-level Supersaturated Designs using Swarm Intelligence Techniques. *Technometrics* **58**(1), 43–49.

Piepho, H.-P. and Williams, E. R. (2021). Regression Models for Order-of-Addition Experiments. Preprint.

Price, K., Storn, R. M., and Lampinen, J. A. (2005). Differential Evolution: A Practical Approach to Global Optimization. Springer.

Poli, R. (2008). Analysis of the Publications on the Applications of Particle Swarm Optimisation. *Journal of Artificial Evolution and Applications* **2008**, 1–10.

Qin, A. K., Huang, V. L. and Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Trans. Evol.Comput.* **12**(1), 398–417.

Quinlan, K. R. and Lin, D. K. J. (2015). Run order considerations for Plackett and Burman designs. *Journal of Statistical Planning and Inference* **165**, 56–62.

Rao, C. R. (1961). Combinatorial arrangements analogous to orthogonal arrays. *Sankhya: The Indian Journal of Statistics, Series A* **23**, 283–286.

Rego, C., Gamboa, D., Glover, F. and Osterman, C. (2011). Traveling salesman problem heuristics: leading methods, implementations and latest advances. *European Journal of Operational Research* **211**(3), 427–441.

Ryberg, P. (2008) Development of a mild and robust method for large-scale palladiumcatalysed cyanation of aryl bromides: Importance of the order of addition. *Organic Process Research & Development* **12**(3), 540–543.

Santner, T. J., Williams, B. J. and Notz, W. I. (2013). *The Design and Analysis of Computer Experiments*, New York: Springer.

Scheffé, H. (1958). Experiments with mixtures.*Journal of the Royal Statistical Society, Series B* **20**, 344–360.

Schoen, E. D. and Mee, R. W. (2021). Order-of-addition orthogonal arrays to elucidate the sequence effects of treatments. Preprint.

Seber, G. A. F. and Lee, A. J. (2003). *Linear Regression Analysis*, 3rd edition, Wiley.

Shi, Y. and Eberhart, R. C. (1998). A modified particle swarm optimizer. *Proceedings of IEEE international Conference of Evolutionary Computation*, 69–73.

Silva, A., Lee, B. Y., Clemens, D. L., Kee, T., Ding, X., Ho, C. M. and Horwitz, M. A. (2016). Output-driven feedback system control platform optimizes combinatorial therapy of tuberculosis using a macrophage cell culture model. *Proceedings of the National Academy of Sciences* **113**, E2172–E2179.

Silvey, S. D. (1980). *Optimal Design*, Chapman and Hall.

Stokes, Z., Mandal, A. and Wong, W. K. (2020), Using Differential Evolution to Design Optimal Experiments, *Chemometrics and Intelligent Laboratory Systems* **199**, 103955.

Stokes, Z. and Xu, H. (2020), A Position-Based Approach for Design and Analysis of Order-of-Addition Experiments, *Statistica Sinica*. To Appear.

Storn, R. (1996). On the usage of differential evolution for function optimization. *Biennial Conference of the North American Fuzzy Information Processing Society* **5**, 519–523.

Storn, R. and Price, K. (1997). Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* **11**, 341–359.

Sun, F., Wang, Y. and Xu, H. (2019). Uniform Projection Designs. *Annals of Statistics* **47**, 641–661.

Sun, J., Xu, W., and Feng, B. (2004). A global search strategy of quantum-behaved particle swarm optimization.*Proceedings of IEEE Conference on Cybernetics and Intelligent Systems*, 111–116.

Tang, B. and Deng, L. Y. (1999). Minimum G_2-aberration for non-regular fractional factorial designs. *Ann. Statist.* **27**, 1914–1926.

Townsend, W. (1978). The single machine problem with quadratic penalty function of completion times: A branch-and-bound solution, *Management Science* **24**, 167-183.

Van Nostrand, R. C. (1995). Design of experiments where the order of addition is important. *ASA Proceeding of the Section on Physical and Engineering Sciences*, 155–160. American Statistical Association, Alexandria, Virginia.

Voelkel, J. G. (2019). The design of order-of-addition experiments. *Journal of Quality Technology* **51**(3), 230–241.

Voelkel, J. G. and Gallagher, K. P. (2019). The design and analysis of order-of-addition experiments: An introduction and case study. *Quality Engineering* **31**, 627–638.

Voutchkov, I.,Keane, A. J., Bhaskar, A. and Olsen, M. (2005). Weld sequence optimization: The use of surrogate models for solving sequential combinatorial problems. *Comput. Methods Appl. Mech. Engrg.* **194**, 3535-3551.

Wang, L., Xiao, Q. and Xu, H. (2018). Optimal Maximin L1-distance Latin Hypercube Designs Based on Good Lattice Point Designs. *Annals of Statistics* **46**(6B), 3741–3766.

Wang, A., Xu, H. and Ding, X. (2020). Simultaneous optimization of drug combination dose-ratio-sequence with innovative design and active learning. *Advanced Therapeutics* **3**, 1900135.

Whitacre, J. M. (2011a). Recent trends indicate rapid growth of nature-inspired optimization in academia and industry. *Computing* **93**, 121–133.

Whitacre, J. M. (2011b). Survival of the flexible: explaining the recent dominance of nature-inspired optimization within a rapidly evolving world. *Computing* **93**, 135–146.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* **1**(1), 67-82.

Wong, W. K, Chen, R. B., Huang, C. C. and Wang, W. C. (2015). A Modified Particle Swarm Optimization Technique for Finding Optimal Designs for Mixture Models. *PLoS ONE* **10**(6), e0124720.

Wu, C. F. J. and Hamada, M. S. (2009). *Experiments: Planning, Analysis and Optimization*, 2nd edition, New York: Wiley.

Xiao, Q., Wang, L., and Xu, H. (2019). Application of Kriging models for a drug combination experiment on lung cancer. *Statistics in Medicine* **38**, 236–246.

Xiao, Q., Wang, Y., Mandal, A. and Deng, X. (2020). Modeling and active learning for experiments with quantitative-sequence factors. Preprint.

Xiao, Q. and Xu, H. (2017). Construction of Maximin Distance Latin Squares and Related Latin Hypercube Designs. *Biometrika* **104**, 455–464.

Xiao, Q. and Xu, H. (2021). A mapping-based universal Kriging model for order-of-addition experiments in drug combination studies. *Computational Statistics and Data Analysis* **157**, 107155.

Xu, H. (2003). Minimum moment aberration for nonregular designs and supersaturated designs. *Statistica Sinica* **13**, 691–708.

Xu, H., Phoa, F. K. H., and Wong, W. K. (2009). Recent developments in nonregular fractional factorial designs.*Statistics Surveys* **3**, 18–46.

Xu, H. and Wu, C. F. J. (2001). Generalized minimum aberration for asymmetrical fractional factorial designs. *Annals of Statistics* **29**, 1066–1077.

Xu, W., Wong, W. K., Tan, K. C. and Xu, J. X. (2019). Finding High-Dimensional D-Optimal Designs for Logistic Models via Differential Evolution. *IEEE Access* **7**, 7133–7146.

Yang, J., Sun, F., and Xu, H. (2021). A component-position model, analysis and design for order-of-addition experiments, *Technometrics* **63**, 212–224.

Yang, M., Biedermann S. and Tang, E. (2013). On optimal designs for nonlinear models: a general and efficient algorithm. *Journal of the American Statistical Association* **108**(504), 1411–1420.

Zaharie, D. (2002). Critical values for the control parameters of differential evolution algorithms. *Proceedings of MENDEL 2002, 8th International Conference on Soft Computing*, 62–67.

Zhang, J.Q. and Sanderson, A.C. (2009). JADE: adaptive differential evolution with optional external archive. *IEEE Trans. Evol. Comput.* **13**(5), 945–958.

Zhao, Y. N., Lin, D. K. J., and Liu, M. Q. (2020). Designs for order-of-addition experiments. *Journal of Applied Statistics*, DOI: 10.1080/02664763.2020.1801607.

Zhao, Y., Li, Z., and Zhao, S. (2020). A new method of finding component orthogonal arrays for order-of-addition experiments. *Metrika*, 1–20.