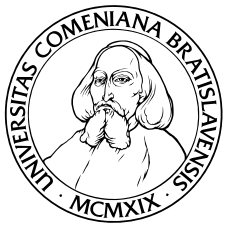


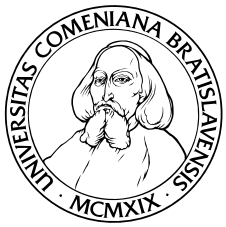
Faculty of Mathematics, Physics and Informatics
Comenius University in Bratislava

Algorithms for computing optimal designs of experiments under constraints

Dissertation thesis

Mgr. Eva Benková

Bratislava, 2022



Faculty of Mathematics, Physics and Informatics
Comenius University in Bratislava

Algorithms for computing optimal designs of experiments under constraints

Dissertation thesis

Mgr. Eva Benková

Applied Mathematics

1114 Applied Mathematics

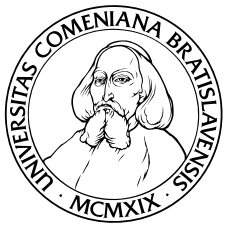
Department of Applied Mathematics and Statistics

Division of Statistics and Insurance Mathematics

Supervisor: doc. Mgr. Radoslav Harman, PhD.

Consultant: Univ.-Prof. Mag. Werner Müller, Dr.

Bratislava, 2022



Fakulta matematiky, fyziky a informatiky
Univerzita Komenského v Bratislave

Algoritmy výpočtu optimálnych návrhov experimentov s ohraničeniami

Dizertačná práca

Mgr. Eva Benková

Aplikovaná matematika

1114 Aplikovaná matematika

Katedra aplikovanej matematiky a štatistiky

Oddelenie štatistiky a poistnej matematiky

Vedúci práce: doc. Mgr. Radoslav Harman, PhD.

Konzultant: Univ.-Prof. Mag. Werner Müller, Dr.

Bratislava, 2022

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Mgr. Eva Benková
Study programme:	Applied Mathematics (Single degree study, Ph.D. III. deg., full time form)
Field of Study:	Mathematics
Type of Thesis:	Dissertation thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Algorithms for computing optimal designs of experiments under constraints

Annotation: The so-called optimal experimental design is a methodology of planning an experiment in a way that provides efficient estimation of the parameters of an underlying statistical model. In the dissertation thesis, we will study the mathematical and algorithmic aspects of the optimal experimental design for the situations with specific restrictions. Such restrictions follow from requirements that are either practical (e.g., to fit into a limited experimental budget) or theoretical (e.g., to guarantee that the distance between the "design points" is larger than a given threshold, which leads to the low correlations of observations).

Aim: The aim of the thesis is to analyse the mathematical properties and develop algorithmic solutions of selected problems of optimal experimental design under various constraints. More specifically, the focus will be on the resource constraints available for the experiment, as well as on the constraints determined by specific geometric requirements on the set of design points.

Tutor:	doc. Mgr. Radoslav Harman, PhD.
Consultant:	Univ.-Prof. Mag. Werner Müller, Dr.
Department:	FMFI.KAMŠ - Department of Applied Mathematics and Statistics
Head of department:	prof. RNDr. Marek Fila, DrSc.
Assigned:	11.02.2013
Approved:	28.02.2013

prof. RNDr. Marek Fila, DrSc.
Guarantor of Study Programme

.....
Student

.....
Tutor

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Mgr. Eva Benková
Študijný program: aplikovaná matematika (Jednoodborové štúdium, doktorandské III. st., denná forma)
Študijný odbor: matematika
Typ záverečnej práce: dizertačná
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Algorithms for computing optimal designs of experiments under constraints
Algoritmy výpočtu optimálnych návrhov experimentov s ohraničeniami

Anotácia: Takzvané optimálne navrhovanie experimentov je metodológia plánovania experimentu, s cieľom poskytnúť efektívny odhad parametrov použitého štatistického modelu. V dizertačnej práci budeme študovať matematické a algoritmické aspekty optimálneho navrhovania experimentov pre situácie so špecifickými ohraničeniami. Tieto ohraničenia plynú z požiadaviek, ktoré sú praktického charakteru (napríklad potreba zohľadniť obmedzené finančné zdroje, ktoré máme k dispozícii na vykonanie experimentu), alebo aj teoretického charakteru (napríklad zaručiť, že vzdialenosť medzi "bodmi experimentu" bude väčšia ako zadaný prah, čo vedie k nízkym koreláciám pozorovaní).

Cieľ: Cieľom dizertačnej práce je analyzovať matematické vlastnosti a vypracovať algoritmické riešenia vybraných problémov optimálneho navrhovania experimentov s rôznymi typmi ohraničení. Konkrétne ide o ohraničenia na zdroje, ktoré sú dostupné pre realizáciu experimentu, ako aj o špecifické ohraničenia určené geometrickými požiadavkami na množinu bodov návrhu experimentu.

Školiteľ: doc. Mgr. Radoslav Harman, PhD.
Konzultant: Univ.-Prof. Mag. Werner Müller, Dr.
Katedra: FMFI.KAMŠ - Katedra aplikovanej matematiky a štatistiky
Vedúci katedry: prof. RNDr. Marek Fila, DrSc.

Dátum zadania: 11.02.2013

Dátum schválenia: 28.02.2013

prof. RNDr. Marek Fila, DrSc.
garant študijného programu

.....
študent

.....
školiteľ

In the first place, I would like to express my gratitude to my supervisor doc. Mgr. Radoslav Harman, PhD. for his guidance and scientific help, as well as for his patience with my nine-year-long studies interrupted by the births of my children. I would also like to thank Univ.-Prof. Mag. Werner Müller, Dr., who kindly agreed to co-supervise my thesis and offered me his help during my stay at Johannes Kepler University Linz. I would also like to thank my family and friends, who supported me throughout my studies, especially my husband Matúš, who may be the only person who has never doubted the successful completion of my doctoral studies.

Mgr. Eva Benková

Abstrakt

BENKOVÁ, Eva. *Algoritmy výpočtu optimálnych návrhov experimentov s ohraničeniami* [dizertačná práca]. Univerzita Komenského v Bratislave. Fakulta matematiky, fyziky a informatiky; Katedra aplikovanej matematiky a štatistiky. Vedúci práce: doc. Mgr. Radoslav Harman, PhD. Bratislava 2022, 111 strán.

V tejto dizertačnej práci skúmame viacero problémov hľadania optimálnych návrhov experimentov za prítomnosti rôznych ohraničení. Ako prvý uvádzame multiplikatívny algoritmus pre konštrukciu D-optimálnych aproximatívnych návrhov ohraničených počtom pokusov a súčasne celkovou cenou experimentu. Navrhnutá metóda je netriviálnou špecifikáciou „barycentrického“ algoritmu predstaveného v [31], ktorý zahŕňa aj zmenšovanie množiny všetkých pokusov odstraňovaním nepotrebných bodov návrhu. Analyticky dokážeme konvergenciu tejto metódy a porovnáme jej výkon s inými konkurenčnými metódami.

Iným typom lineárne ohraničených návrhov sú priestor-vypĺňajúce návrhy často používané v počítačových experimentoch. V práci navrhujeme všeobecnú schému heuristiky výmenného typu pre výpočet exaktných návrhov, ktorá je založená na pojme „exkluzívnych množín“ a dá sa adaptovať pre rôzne priestor-vypĺňajúce ohraničenia. Naša špecifikácia tejto schémy pre takzvané Bridge návrhy výrazne prekonáva zavedenú metódu. Ďalej sme túto všeobecnú schému implementovali pre takzvané Minimum-distance návrhy, pričom sme využili Voronoiove diagramy a ich zovšeobecnenia - mocninné diagramy. Úlohu týchto diagramov pri konštruovaní efektívnych Minimum-distance návrhov vysvetľujeme odvodením niektorých ich kľúčových teoretických vlastností. Výkon tejto heuristiky v oboch prípadoch ohraničení vyhodnocujeme vzhľadom na konkurenčné algoritmy v numerických štúdiách.

Kľúčové slová: navrhovanie experimentov, optimálny návrh, lineárne ohraničený návrh, kritérium D-optimality, multiplikatívny algoritmus, výmenný algoritmus, Voronoiov diagram.

Abstract

BENKOVÁ, Eva. *Algorithms for computing optimal designs of experiments under constraints* [dissertation thesis]. Comenius University in Bratislava. Faculty of Mathematics, Physics and Informatics; Department of Applied Mathematics and Statistics. Supervisor: doc. Mgr. Radoslav Harman, PhD. Bratislava 2022, 111 pages.

In this dissertation thesis, we study several problems of finding the optimal experimental design under various design constraints. First, we introduce a multiplicative algorithm for constructing D-optimal approximate size- and cost-constrained designs, that is, designs restricted on the number of trials and, simultaneously, on the total cost of the experiment. The proposed method is a non-trivial specification of the "barycentric" algorithm introduced in [31], which includes also reducing the design space by removal of unnecessary design points. We analytically prove its convergence and demonstrate its performance in comparison to the competing methods.

A different type of linearly constrained designs are space-filling designs frequently used in computer experiments. We propose a general exchange-type heuristic framework for computing exact designs, which is based on the notion of "privacy sets" and can be adapted for various space-filling restrictions. Our specification of the framework for the so-called Bridge designs significantly outperforms the state-of-the-art method. Moreover, we implemented the general framework for the so-called Minimum-distance designs by the use of Voronoi diagrams and their generalizations - power diagrams. We explain the role of these diagrams in constructing efficient Minimum-distance designs by deriving some crucial theoretical properties. Performance of the heuristic for both kinds of constraints is compared against competing algorithms in numerical studies.

Key words: design of experiments, optimal design, linearly constrained design, D-optimality criterion, multiplicative algorithm, exchange algorithm, Voronoi diagram.

Contents

Chapter I

Introduction

1 Design of experiments

One of the most common tasks of statistics as a scientific discipline is to explain dependence of the so-called output (dependent) variables on the set of input (independent) variables. The aim of *design of experiments* (or *experimental design*) is to determine the values of the input variables, in case these are not already given and can be regulated by the experimenter. The experiment itself consists of a series of trials (called also runs or measurements), with each trial representing a certain combination of the input parameter settings. The main interest may vary - from estimating unknown parameters as precisely as possible, through predicting the output value for a given input, to finding a minimum/maximum of the output function - which leads to various approaches.

Let y be an observed output value of a single trial of an experiment believed to depend upon d explanatory, input variables $x_1, \ldots, x_d$, called also the *factors*. The individual factors typically take values in the bounded intervals whose boundaries may be given for example by the physical limitations of the system or by the interests of the experimenter. For instance, for safety reasons, participants in clinical trials may not be given doses of the medicaments that might be considered toxic. It may be convenient to scale the individual intervals of permissible values into the interval $[0, 1]$, or $[-1, 1]$.

The resulting set of permissible combinations of factor values - called also a *design region* or a *design space* - is then the d-dimensional standardized hypercube $[0, 1]^d$ (or $[-1, 1]^d$). In implementations it is typical to *discretize* a cubical design space into an

equally-spaced grid of size L^d, where $L \in \mathbb{N}$ is a suitably chosen constant. This results in finitely many points of the design space, which is a necessary condition for many algorithms for computing designs.

Sometimes it is necessary to restrict the design space by some additional constraints, leading to the design spaces of a non-cubic shape. This thesis deals with the so-called *constrained experimental design*, which should not be mistaken with the constrained design regions. In fact, it is not uncommon that both kinds of restrictions are present at the same time, see, e.g., Figures 3.6 and 3.7 of Chapter III.

Principles of experimental design were established by Fisher in 1935 [26]. Although the great part of this pioneering work was focused on the use of statistical methods in agriculture, nowadays the design of experiments has its applications across the scientific disciplines - mostly in natural and social sciences, engineering, marketing and policy making.

One of the principles of [26] is a so-called *randomization*, which means, for example, a random assignment of treatments to individuals from a population of interest in clinical trials. In other words, the group receiving one kind of a treatment has to be a random sample from a larger population, so that the results of the experiment may be extended to that population. Moreover, randomization helps to reduce probability of bias of the estimated parameters in case some unobservable and uncontrollable covariates are present.

Another tool that can be useful is *blocking*: In some experiments, it is possible to identify different groups of experimental units - "blocks", such that the values of a covariate z are similar within the blocks, but different for different blocks. The block effects are typically nuisance parameters and their estimation is not of interest to the experimenter. However, if the experimental units are correctly assigned to the blocks and block effects are included in the statistical model, the model parameters are estimated with greater precision.

By the term *experimental design* we intuitively understand composing the experiment by choosing some points from the design space. Although the order of the selected points may sometimes be of importance, in the thesis, we always consider cases where this

order plays no role. There are various admissible design representations: In practice, the experimenter is allowed to perform only a certain number N of carefully selected trials. This can be represented by an *exact design*, which is a selection of points of the design space in which the trials are performed. Nonetheless, for a large N, one can provide only a certain percentage of the measurements in a particular point, rather than specify the exact number. This approach leads to the notion of an *approximate design*, which is in fact a probability measure on the design space.

Since this thesis covers multiple design problems of quite heterogeneous character, multiple definitions of an experimental design are required. We provide these definitions together with more detailed introductions at the beginnings of Chapters II and III. For a deeper insight into (optimal) experimental design, see, e.g., [1], [54], [24], [61].

This chapter provides a very brief introduction to design of experiments, with focus on optimal experimental design theory, especially under various design constraints. Note that every design is already restricted by a constraint: an exact design by the total number of trials N, which cannot be exceeded; an approximate design by the total sum of 100%, which are to be distributed among the points of the design space. Such designs with one restriction are not considered as constrained. However, if more limitations are present simultaneously (which can easily happen in practice), the standard techniques of constructing designs have to be replaced by more appropriate methods.

Computing these *constrained experimental designs* is the main focus of the thesis. The formal definitions of a constrained design can be found in the introductions to the individual chapters, since the experimental design itself is defined in two different ways. For now let us perceive these constraints simply as some additional restrictions posed on the design.

2 Standard designs

This thesis focuses mainly on the topics of *optimal* experimental design. However, there is a wide range of designs used throughout history, many of them not dealing with the optimality criteria at all. Some of these "standard" or "classical" designs that have

Trial number	x_1	x_2	x_3	Treatment combination
1	-1	-1	-1	(1)
2	+1	-1	-1	a
3	-1	+1	-1	b
4	-1	-1	+1	c
5	+1	+1	-1	ab
6	+1	-1	+1	ac
7	-1	+1	+1	bc
8	+1	+1	+1	abc

Table I.1: Full factorial design for $d = 3$. In the last column, the trials are labelled by a letter-combination, where the letters a, b, c signify $x_1 = +1, x_2 = +1, x_3 = +1$, respectively.

been widely used by practitioners with good results are presented in this section. As for literature, the information contained in this section is taken mostly from [1].

Let us have d factors that span the interval $[-1, 1]$ (after scaling). The *full factorial design* typically consists of all combinations of the two extreme levels $-1, 1$ for each factor, resulting in 2^d trials in total. An example of such a design for $d = 3$ is given in Table I.1 where the individual runs are ordered in the standard way and should be randomized when performing an actual experiment.

Fisher's work [26] argued that factorial designs offer several advantages compared to the one-factor-at-a-time (OFAT) experiments, which vary only one factor at a time while keeping the other factors constant. Factorial designs are more efficient in the sense of providing more information at the same cost. Moreover, in contrast to OFAT, they are able to detect interactions between the factors.

Naturally, the factorial designs varying two levels for each factor are useful if the relationship between the factors $x_1, \ldots, x_d$ and the response y is *linear*. In this case, the factorial designs are *orthogonal*, which means that the individual parameters can be estimated without any correlation. That is, the covariance matrix of the parameter

estimates is diagonal, and so is the information matrix of the design (see definitions (2.3) in Chapter II and (3.3) in Chapter III). This is advantageous, e.g., when there are non-significant terms in the model - the parameters do not have to be re-estimated after the deletion.

In case the dependence of the output variables on the input factors is not linear, the model can be augmented by quadratic terms. In order to check whether such terms are needed in the model or not, there are often a few (three or four) "centre points" added to the design. These points take values $x_j = 0$ for one or more $j \in \{1, \ldots, d\}$. Some methods alternative to the replication of the centre point are described for example in Sections 19 and 20 of [1].

A disadvantage of full factorial designs is their size (the number of trials), which grows rapidly with higher dimensions. Their large size allows them to quite precisely estimate all higher-order interactions. However, if these interactions are known to be negligible, it is enough to run only a fraction of the complete 2^d trials without losing much information.

Fractional factorial designs enable the significant reduction of the number of the total runs: 2^d original design points are divided into 2^f blocks of size 2^{d-f} and only one of the blocks is run during the experiment. As a result, the same linear combinations of the observations estimate multiple (specifically 2^f) parameters of the original model. In order to decrease the sample size, the information about the higher-order interactions is sacrificed - it is *confounded* with the blocks.

For an example of a fractional factorial design see Table I.2. The 8 treatments of the full factorial design displayed in Table I.1 are divided into 2 blocks of 4 treatments each. Since by running one of the blocks we obtain only 4 observations to 8 unknown parameters, there are always two parameters that are estimated by the same linear combination of the 4 responses (eventually multiplied by (-1) - depending on the chosen block). For more details see, e.g., Section 7.4 in [1].

A disadvantage of the fractional factorial designs is that they are quite restrictive - the number of runs must be the power of 2. In [56], orthogonal designs of sizes that are multiples of 4 are proposed, up to the sample size of 100 (with the exception of 92). The

Block number	Treatment combinations			
1	a	b	c	abc
2	(1)	ab	ac	bc

Table I.2: 2^3 treatment combinations from Table I.1 divided into two blocks.

idea of these *Plackett-Burman designs* is that for any two factors, all 4 combinations of the two levels $-1, +1$ appear the same number of times, see Table I.3 for illustration. If the sample size is a power of 2, Plackett-Burman design is identical with the (fractional) factorial design.

Trial number	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
1	+	+	-	+	+	+	-	-	-	+	-
2	-	+	+	-	+	+	+	-	-	-	+
3	+	-	+	+	-	+	+	+	-	-	-
4	-	+	-	+	+	-	+	+	+	-	-
5	-	-	+	-	+	+	-	+	+	+	-
6	-	-	-	+	-	+	+	-	+	+	+
7	+	-	-	-	+	-	+	+	-	+	+
8	+	+	-	-	-	+	-	+	+	-	+
9	+	+	+	-	-	-	+	-	+	+	-
10	-	+	+	+	-	-	-	+	-	+	+
11	+	-	+	+	+	-	-	-	+	-	+
12	-	-	-	-	-	-	-	-	-	-	-

Table I.3: Plackett-Burman design for 12 runs and up to $d = 11$ factors. The signs "$-, +$" stand for the lower and the upper boundaries $-1, +1$, respectively. In case less than 11 factors are present, the corresponding subset of the columns of the table is run.

If a second-order polynomial model is to be fitted, more than just two levels for each factor have to be considered. One solution would be to use a 3^d factorial design, with

each factor varying over -1,0 and 1. This method, however, produces an excessive number of trials for larger d. So-called *composite designs* can be used instead and consist of 2^{d-f} fractional factorial design points ($f \geq 0$) augmented by $2d$ "star" points. Star points have $d-1$ zero coordinates and one coordinate with 1 or -1 value. If a few centre points are included in the design as well, we talk about *central* composite designs.

3 Optimal designs

Clearly, the standard experimental designs described in the previous section provide several advantages: They are simple enough to be directly applied, yet still yielding satisfactory results in many cases. On the other hand, there is little that we know about their actual efficiency, i.e., how "well" they perform in comparison to other designs. The quality of a "good" design is discussed below.

Let us now formalize the dependence of y on x by the *linear regression model*

$$y(x) = \mathbf{f}(x)^\top \beta + \varepsilon(x), \tag{1.1}$$

where

- $y(x)$ is an observation in design point $x \in \mathcal{X}$,
- $\mathbf{f}(x) \in \mathbb{R}^m$ is the value of a known regressor in design point x,
- $\beta \in \mathbb{R}^m$ is a vector of unknown parameters.

We assume that the errors $\varepsilon(x), x \in \mathcal{X}$, have mean value equal to 0, variance equal to some positive constant $\sigma^2 < \infty$, and trials are performed independently. Moreover, for simplicity we require that $\text{span}\{\mathbf{f}(x) : x \in \mathcal{X}\} = \mathbb{R}^m$ and $\mathbf{f}(x) \neq \mathbf{0}_m$ for all $x \in \mathcal{X}$.

Assume now the linear regression model with one unknown parameter, that is, $m = 1$. It is a known result of the Gauss-Markov theorem, that the least squares estimator is the best linear unbiased estimator (BLUE), in the sense of having the minimum variance. The reciprocal of this variance is called also the "Fisher information", which suggests that minimizing the variance equals maximizing the information. In optimal experimental design, a design is considered to be *good* if the variance of the parameter estimates is

small, that is, if the information gained by performing the experiment is large (in the best case of the minimal variance and the maximal information, the design is *optimal*).

In the vast majority of real-life examples, however, there are several parameters in the linear regression model ($m \geq 2$). The corresponding variance of this vector of parameters is a matrix and "minimizing a matrix" is an intricate problem that can be handled in multiple ways. The diversity of approaches is mirrored in the diverse objective functions (called also the criteria of optimality). In all of these cases, an optimal experimental design solves an optimization problem of "minimizing" the covariance matrix of the parameters, or, equivalently, of "maximizing" its inverse - the information matrix (see (2.3) in Chapter II and (3.3) in Chapter III for formal definitions).

We note that what we use as a measure of the quality of a design is one of many possible assessments of a design - the one based on the information matrix of a design. The information matrix describes the amount of information on the unknown parameters of the statistical model 1.1, therefore we also use term *model-based designs*.

The theory of optimal design offers a new insight into the popular standard designs discussed in the previous section: Some designs such as factorial designs or Plackett-Burman designs are in fact D-optimal (see definitions 2.6 in Chapter II and 3.4 in Chapter III) for the regression model with first-order terms. Under more complicated conditions, such as a constrained design region or a constrained design itself, the standard designs might be impermissible and appropriate algorithms need to be employed to find the optimal (or nearly optimal) designs.

3.1 State of the art

The pioneering paper on optimal experimental design was [68], in which Danish statistician Kirstine Smith proposed optimal designs for one-factor polynomial models. Since then, this field of statistics has progressed in many directions. This thesis is aimed mainly at the algorithms for the computation of optimal designs, since these are an inevitable part of incorporating the optimal design theory into practice.

The methods of computing exact designs (allocating N design points in a design space $\mathcal{X}$) and approximate designs (assigning the points in $\mathcal{X}$ percentages instead of the concrete

quantities) are typically fundamentally different. In below, we list some of the methods of constructing approximate and exact designs both with focus on the constrained designs, in the respective order.

Finding an optimal approximate design is in fact for many criteria a special case of convex optimization. Therefore, naturally, many approximate design algorithms incorporate ideas of this research field. The classical example is the so-called *vertex-direction algorithm* (a special case of the so-called Frank-Wolfe algorithm), which is based on the steepest ascent method and was first introduced in [24] and in [84] (for D-optimal designs). In [83], its convergence for more general design criteria is proven. This first-order iterative method can be modified for any linearly constrained designs as well (see [14], [23]) and can be applied to various optimality criteria.

Another approach to computing approximate optimal designs under multiple constraints is to use a standard algorithm applied to the primary criterion modified by a penalty term (see [7]). An advantage of this approach is that it can be applied to a general criterion and non-linear convex constraints. However, the designs which the algorithm produces are guaranteed to be permissible only asymptotically, with increasing magnitude of the penalty term. Moreover, the variables of the algorithm cannot be directly used for statistically interpretable stopping rules.

Another class of methods is motivated by the fact that an optimal design under convex constraints can be found by maximizing a compound criterion or by the closely related analytic technique of Lagrange multipliers (e.g., [47], [16], [14], [71], [43]). The most problematic aspect of these methods is that the parameters which define the compound criterion, or the Lagrange multipliers, are not known in advance, and they must be determined by a separate method, which is usually iterative. The advantages and disadvantages of this approach are similar to the penalty approach.

The so-called *barycentric multiplicative algorithm* proposed in [31] has favourable properties similar to standard multiplicative algorithms ([66], [70], [72], [17], [73], [85]) - easy implementation, low memory requirements. This method is designed to deal with specific design constraints, yielding for example stratified designs, as demonstrated in [31]. Under mild technical conditions, its monotonic convergence to the optimum can

be proved. Moreover, the barycentric algorithm can be seamlessly combined with stopping rules based on statistically natural efficiency as well as with "deletion rules" for the removal of redundant design points, which greatly enhances the speed of the computation (cf. [35]). The barycentric multiplicative algorithm computing so-called size- and cost-constrained designs is the main object of Chapter II and was proposed in [33].

Another choice are modern methods of mathematical programming, such as maxdet programming ([77]), semidefinite programming (SDP, cf. [5]) or second-order cone programming ([64]). Although these algorithms are very versatile, their time and memory requirements grow steeply with increasing size of the problem. Furthermore, they require advanced software tools and their practical application can be technically challenging. Also, it is not straightforward to combine these methods with statistically meaningful stopping rules.

In real-life problems, only exact designs with a given number of trials N can be used for an actual experiment. In order to find a "good" exact design, one can use an optimal approximate design, which allots a fraction (percentage) of all performed measurements to each point. For a given N, however, distributing measurements according to an approximate design may yield a non-integer number of trials in one or more points of the design space. This can be solved by employing a *rounding method* (see [41], [62]), which "rounds" an approximate design to the *closest* exact design of a given size N, where various ways of interpreting the "closeness" are possible. However, if the number of trials N is rather small, this approximation is not very efficient and we should focus directly on exact designs instead.

In case of the restricted designs, the rounding becomes even more difficult with no appropriate universal method available. In situations such as, e.g., the case with no replicated observations in the same design point (see Chapter III), rounding an optimal approximate design can be very inefficient and it is better to use one of the methods of computing optimal exact designs of size N (with respect to the constraints). Some of the adequate methods are listed in the following.

Finding an optimal exact design is in general a complex problem with many local optima, unless n and N are very small numbers, which permits the complete enumeration of all exact designs. For finite design spaces of size n, one could use for instance an

"intelligent" enumeration method, such as branch-and-bound or branch-and-cut, that guarantees a globally optimal solution (see [79], [64], [34]). Nevertheless, these methods tend to get extremely slow for larger design problems.

From a practical point of view, it is often worth to search for a design with high efficiency rather than for an optimal exact design. Various heuristics and nature-inspired algorithms can be used to find a "good-enough" design with much less computational effort (see, e.g., [24], [48], [30], [46]).

Many of these heuristics are based on the idea of adding and deleting points so that the current design is improving. For example, DETMAX algorithm ([48]) performs "excursions" of various sizes in which a chosen number of points is sequentially added and then deleted. In [80], its modification is used to solve replication-free design problems.

Nowadays, one of the most popular methods for solving the standard optimal design problems are the local-search exchange algorithms (see Chapter 12 in [1] and [28]). They extend the idea of DETMAX procedure by considering the operations of addition and deletion at the same time, see, e.g., KL-exchange algorithm described in Section 12.6 in [1]. However, the application of exchange methods on constrained design problems is in general rather limited. The main difficulty is that performing an exchange on a permissible design can easily yield an impermissible design. One should thus try to exchange the points wisely, taking the specific restrictions into account.

One of the most broadly applicable algorithms for constrained exact designs is proposed in [32]. This heuristic method assumes a finite design space of a "reasonable" size n. This so-called resource constraints heuristic is described in more detail in Section 4.7 of Chapter III, where its performance is compared to our own method for a specific type of linear constraints.

A great part of the thesis is devoted to the search for so-called *space-filling designs*. The usual space-filling approach, however, does not include optimizing a statistical criterion or posing restrictions on design. Hence, there are not many methods for computing constrained optimal designs in this research area. One exception is the algorithm proposed in [39], which deals with the so-called Bridge designs and the criterion of D-optimality.

The cited paper is in some sense the starting point for our own algorithms presented in Chapter III.

4 Goals of the thesis

In the past, the research related to the computation of optimal experimental designs focused mostly on situations with a single linear constraint that can be interpreted as a limit on the "size" of the experiment. However, experimental situations with multiple constraints do occur in practice; they can represent restrictions on the availability of various resources consumed by the experiment. Although useful for applications, efficient numerical methods for computing optimal designs under non-standard constraints receive only sporadic attention (apart from the literature cited above, see also [69], [52], [82]). The reason is that multiple constraints on experimental design pose challenging problems of theoretical as well as computational nature.

One of the aims of the thesis can be formulated as follows:

- Propose an algorithm of the multiplicative type for computing approximate D-optimal designs under a pair of linear restrictions, which can be interpreted as simultaneous constraints on the size and on the total cost of the experiment. This can be viewed as the simplest problem in optimal design beyond the standard case of a single linear constraint. Base the proposed method on the barycentric algorithm introduced in [31] and prove its convergence for this specific case.

Many of the classic problems of design of experiments can be formulated in terms of the constrained designs, even though it is not the usual approach in the particular design field. For instance, most of the literature on space-filling designs attempts to achieve its aim by optimizing a prescribed objective measuring a degree of space-fillingness, such as maximin, minimax, etc., sometimes combined with an estimation or prediction oriented criterion (like suggested in [49]). Let us label those as *soft* space-filling methods. In contrast, *hard* space-filling methods ensure desirable properties by enforcing constraints on the designs, such that a secondary criterion can be used for optimization (e.g., D-optimality in case of the Bridge designs of [39]). This leads us to the following objective:

- Propose a general framework for a hard space-filling method of an exchange type, based on the central notion of so-called *privacy sets*. Privacy sets represent a wide range of different, mostly space-filling, design constraints and allow for an efficient and elegant formulation of the algorithms.

- Specify the proposed framework for the class of Bridge designs and compare the proposed algorithm to the state-of-the-art method ([39]).

The idea of space-filling itself suggests spreading the design points apart from each other. Since, as discussed above, the usual space-filling approach is focused primarily on the target function to be optimized, the exact formulation of the appropriate design constraints is missing. Thus, we establish the following goal:

- Define a new class of so-called Minimum-distance designs and specify the proposed framework accordingly. In order to propose an efficient functional algorithm, utilize the so-called Voronoi diagrams and their weighted generalizations called power diagrams and provide theoretical justification for their implementation in the method. Finally, evaluate the performance of the proposed algorithm against the other relevant methods.

5 Outline

Apart from the introductory Chapter I, this thesis consists of two main chapters.

Chapter II deals with approximate designs under simultaneous size- and cost-restrictions. We start the chapter with the formal definition of an experimental design that is worked with throughout this chapter. Section 1 unifies necessary preliminaries by introducing the linearly constrained designs, providing fundamentals of optimal experimental design and finally describing the general barycentric multiplicative algorithm that we elaborate on in the sequel. In Section, 2 we justify the notion of a D-optimal size- and cost-constrained design and formulate the key optimization problem (2.16). In Section 3, we provide selected theoretical results for the problem (2.16), including the so-called equivalence theorem and the deletion rules. Section 4 describes a barycentric algorithm for computing designs solving (2.16), called S&C algorithm. Section 5 gives a more complex example of a D-optimal size- and cost-constrained design, provides a numerical study

exploring the behaviour of the proposed S&C algorithm and compares its performance to the selected competing methods. In Section 6, we provide miscellaneous short remarks related to the constrained optimal design problems (2.13) and (2.16).

Chapter III is oriented on exact space-filling designs restricted by privacy sets of the individual design points. At the beginning of the chapter, we provide the definition of an exact design with no replications that will be used throughout this chapter. We continue with Section 1, in which we specify the linear constraints as well as some of the fundamental optimal design notions for that particular definition of a design. Short but crucial Section 2 includes the definition of privacy sets, which are the key concept in the so-called Privacy Sets Algorithm (PSA) - a general framework dealing with a wide range of space-filling constraints. In Section 3, we propose a specific version of PSA for the class of Bridge designs and apply the method to examples of various kind. Finally, in Section 4, we specify PSA for another type of space-filling designs, the so-called Minimum-distance designs. We base the algorithm on Voronoi diagrams, theoretically justify their use and evaluate the algorithm against other available methods.

Some of the results of the thesis have already been published: Chapter II has been addressed in [33]. Sections 2 and 3 of Chapter III have been published in [6] and in [50].

Chapter II

Size- and cost-constrained designs

All the results presented in this chapter are based on the paper [33]. We refer the reader to this paper for the proofs of the provided theorems.

The discussion in the introductory Chapter I justifies various design representations as relevant. Let us have a design space $\mathcal{X}$ and a required number of measurements $N \in \mathbb{N}$. For N large enough it is possible to apply the so-called *approximate design* theory, viewing a design as a probability measure on $\mathcal{X}$ with a finite support, which leads to a simpler problem of convex optimization. In other words, there are finitely many points in $\mathcal{X}$ with a positive design "weight" assigned, and the overall sum of the weights is 1. Negative weights are not permitted.

If $|\mathcal{X}| = n$, we can identify this probability measure with an n-dimensional vector $\mathbf{w}$ with its coordinate $w_x \geq 0$ denoting a weight of design point $x \in \mathcal{X}$, that is,

$$\sum_{x \in \mathcal{X}} w_x = 1, \quad w_x \geq 0 \quad \text{for all } x \in \mathcal{X}. \tag{2.1}$$

In this relaxation of the original problem with N trials, weight w_x represents an approximate ratio of trials performed in the point x to the total number of trials. From a geometric point of view, the set of all approximate designs on a finite $\mathcal{X}$ forms a unit probability simplex in $\mathbb{R}^n$. We denote this simplex by $\mathbb{P}^n$. In this chapter, we always assume a finite design space $\mathcal{X}$ and represent a design by an n-dimensional vector $\mathbf{w}$.

In contrast to approximate designs, an *exact design* of size $N \in \mathbb{N}$ has to satisfy 2.1 and

$$w_x = N_x/N, \quad N_x \in \{0, 1, \ldots, N\} \quad \text{for all } x \in \mathcal{X}, \quad \text{and } \sum_{x \in \mathcal{X}} N_x = N,$$

where N_x represents the number of trials performed in $x \in \mathcal{X}$. However, since this chapter deals with a special problem of approximate designs, $\mathbf{w}$ always denotes an approximate design rather than an exact design. Geometrically, the set of all permissible exact designs of size N forms a discrete subset of $\mathbb{P}^n$.

1 Preliminaries

In this section, we provide some general information on the (constrained) experimental design, as well as more recent results that are key for our work. We do not provide our own results in this section.

1.1 Linear constraints

When performing an experiment, it often happens that one is limited by some restrictions additional to the maximum allowed number of measurements, i.e., multiple constraints are present. Design problems with multiple constraints are in general difficult, especially for a large design space, even when the constraints are linear ([14]). For instance, even finding a single permissible exact design (or proving that there is no such a design) may be very challenging.

Design $\mathbf{w}$ is called *linearly constrained*, if it satisfies (2.1) and

$$\sum_{x \in \mathcal{X}} a_{xj} w_x \leq b_j, \quad \text{for } j = 1, \ldots, l, \tag{2.2}$$

where $l \in \mathbb{N}$ and $a_{xj}, b_j \in \mathbb{R}$ are given constants.

For a finite $\mathcal{X}$ of size n, the set of all permissible approximate designs forms a convex polytope $\mathbb{Q}^n \subseteq \mathbb{P}^n$ (we will use this universal notation for all kinds of linearly constrained designs).

Let us provide a brief overview of some of the most common linear restrictions. All of the below-mentioned constraints do occur in practise, and there are still many that could be added on the list. We introduce mostly those constraints that receive particular attention in the thesis, some of them more than the others. In addition, there is a group of so-called space-filling designs defined in Chapter III - these are typically exact

designs whose definition in terms of the approximate design vector $\mathbf{w}$ would be needlessly complicated and somewhat unnatural.

First note that the original restriction on the number of trials (2.1) is clearly a special case of (2.2) for $l = 1$, $b_1 = 1$ and $a_{x1} = 1$ for all $x \in \mathcal{X}$ with a sign of equality instead of inequality.

Assume now there are positive unequal costs c_x corresponding to trials in points $x \in \mathcal{X}$, and our budget is limited (see, e.g., Section 6 in [22]). This restriction can be formalized by one constraint (2.2), such that values $a_{x1} = c_x$ represent possibly unequal normalized costs of trials in individual design points and $b_1 = 1$. Any permissible design must then satisfy two constraints: the "size" constraint and the "cost" constraint. This situation is in detail discussed in this chapter, where an efficient multiplicative algorithm computing optimal approximate *size- and cost-constrained* designs is proposed.

A natural restriction is to limit the design weights in points x by some positive constants s_x, for all $x \in \mathcal{X}$. These *direct constraints* (cf. [25], [80]) can be written in terms of (2.2) by setting $l = n$, $a_{xy} = \delta_{xy}$ (the Kronecker delta) and $b_y = s_y$ for all $x, y \in \mathcal{X}$. For exact designs of size N, it is also usual to set $s_x = 1/N$ for all $x \in \mathcal{X}$, yielding the so-called *replication-free designs*. Note that in Chapter III, by defining a design as a set (definition (3.1)), we automatically assume it to be replication-free.

Another class of constraints corresponds to the *marginal restrictions* (see, e.g., [15], [42]) or, more generally, *strata restrictions* ([31]). In case of the latter, the design space $\mathcal{X}$ is partitioned into non-overlapping sets $\mathcal{X}_1, \ldots, \mathcal{X}_k$ (called also strata or partitions) and any design $\mathbf{w}$ must satisfy $\sum_{x \in \mathcal{X}_j} w_x \leq s_j$ for all $j = 1, \ldots, k$, where $s_1, \ldots, s_k$ are given positive weights. Here, we obtain (2.2) by setting $l = k$, $a_{xj} = I[x \in \mathcal{X}_j]$ (the indicator function) and $b_j = s_j$ for all $j = 1, \ldots, k$.

1.2 Optimal design theory

We can now explicitly define the optimization problem outlined in Section 3 of Chapter I. The aim of optimal experimental design as a statistical discipline is to solve this problem, in order to ensure maximal informational yield from the experiment. First, we describe the

standard situation with one restriction (2.1) only, and put it into the context of linearly constrained designs afterwards.

The aim of an experimenter is to obtain as much information about unknown parameters of the model 1.1 as possible. This means that we deal with an optimization problem where the objective function is defined on the set of *information matrices*, where the information matrix of design $\mathbf{w}$ is given by

$$\mathbf{M}(\mathbf{w}) = \sum_{x \in \mathcal{X}} w_x \mathbf{M}_x, \tag{2.3}$$

where $\mathbf{M}_x = \mathbf{f}(x)\mathbf{f}(x)^\top$ is a matrix of rank 1, representing information observed from a single point $x \in \mathcal{X}$. Note that for a non-singular information matrix, its inverse is proportional to the covariance matrix of the best linear unbiased estimator (BLUE) of β. Let $\mathcal{M}$ denote the set of all information matrices for a given regression function $\mathbf{f}$.

In the sequel, design $\mathbf{w}$ is called *regular* if its information matrix $\mathbf{M}(\mathbf{w})$ is non-singular. The objective function $\phi : \mathcal{M} \to \mathbb{R}$ to be maximized is called the *optimality criterion* and has to be monotonic in the sense that

$$\mathbf{M}(\mathbf{w}_1) \preceq \mathbf{M}(\mathbf{w}_2) \Rightarrow \phi(\mathbf{M}(\mathbf{w}_1)) \leq \phi(\mathbf{M}(\mathbf{w}_2)), \quad \mathbf{w}_1, \mathbf{w}_2 \in \mathbb{P}^n,$$

where the symbol "$\preceq$" denotes Loewner ordering, i.e., $\mathbf{M}(\mathbf{w}_2) - \mathbf{M}(\mathbf{w}_1)$ is a positive semidefinite matrix. We most frequently write $\phi(\mathbf{w})$ instead of $\phi(\mathbf{M}(\mathbf{w}))$, i.e., we omit the symbol of information matrix from the input of the optimality criterion, if it is not necessary. However, the meaning remains the same.

The *efficiency* of a design $\mathbf{w}_1$ relative to a design $\mathbf{w}_2$ is given by

$$\mathrm{eff}_\phi(\mathbf{w}_1|\mathbf{w}_2) = \frac{\phi(\mathbf{w}_1)}{\phi(\mathbf{w}_2)}. \tag{2.4}$$

As definition (2.4) implies, the efficiency represents the statistical "quality" of design $\mathbf{w}_1$ compared to design $\mathbf{w}_2$. This is meaningful if the optimality criterion is *positively homogeneous*: For every $\alpha > 0$ and for every $\mathbf{w} \in \mathbb{P}^n$ it holds that $\phi(\alpha\mathbf{M}(\mathbf{w})) = \alpha\phi(\mathbf{M}(\mathbf{w}))$. In that case, the inverse value $\mathrm{eff}_\phi(\mathbf{w}_1|\mathbf{w}_2)^{-1}$ says how many times we need to perform the experiment conducted by $\mathbf{w}_1$ in order to gain the same amount of information as from the experiment conducted by $\mathbf{w}_2$.

Usually, it is our concern to compare a design $\mathbf{w}$ to an *optimal design* $\mathbf{w}^*$ that maximizes the optimality criterion ϕ:

$$\mathbf{w}^* \in \arg\max_{\mathbf{w}\in\mathbb{P}^n} \phi(\mathbf{w}).$$

In this case, we have $\mathrm{eff}_\phi(\mathbf{w}|\mathbf{w}^*) \leq 1$.

For linearly constrained designs, the optimization is restricted to the convex polytope $\mathbb{Q}^n$ leading to

$$\mathbf{w}^* \in \arg\max_{\mathbf{w}\in\mathbb{Q}^n} \phi(\mathbf{w}). \tag{2.5}$$

It is sensible to set the linear restrictions so that there exists a regular design $\mathbf{w} \in \mathbb{Q}^n$. For some models the optimal solution of (2.5) is not unique, but the set of all optimal solutions is convex.

In this work, we focus mainly on the most common criterion called *D-optimality* criterion. It can be in principle defined in several ways, all of them being increasing functions of the determinant of the information matrix. We prefer the following definition (2.6) which ensures the positive homogeneity of D-optimality criterion:

$$\phi_D(\mathbf{w}) = \det{}^{1/m}(\mathbf{M}(\mathbf{w})), \quad \mathbf{w} \in \mathbb{P}^n. \tag{2.6}$$

It is possible to show that the D-optimality criterion is concave and continuous (see, e.g., [53]). The statistical interpretation is that any D-optimal design minimizes the volume of the confidence ellipsoid of the vector of parameters β. In case there exists a regular design $\mathbf{w} \in \mathbb{Q}^n$, the optimal criterial value $\phi_D(\mathbf{w}^*)$ is strictly positive.

Another popular criterion is A-optimality defined for a regular design $\mathbf{w}$ by

$$\phi_A(\mathbf{w}) = 1/\mathrm{tr}(\mathbf{M}(\mathbf{w})^{-1}), \quad \mathbf{w} \in \mathbb{P}^n.$$

For a singular design $\mathbf{w}$, we set $\phi_A(\mathbf{w}) = 0$. An A-optimal design minimizes the average variance of the parameter estimates.

1.2.1 The equivalence theorem

The general equivalence theorem is a fundamental theorem in the theory of approximate experimental designs. It provides necessary and sufficient conditions for a design to be

optimal. Moreover, it also enables one to develop algorithms converging to the global optimum.

For any design $\mathbf{w} \in \mathbb{P}^n$, let $\nabla\phi(\mathbf{M}(\mathbf{w}))$ be the *gradient* of criterion ϕ in $\mathbf{M}(\mathbf{w})$ given by

$$\nabla\phi_{ij}(\mathbf{M}(\mathbf{w})) = \frac{\partial\phi(\mathbf{M}(\mathbf{w}))}{\partial \mathrm{M}_{ij}(\mathbf{w})}, \quad i,j = 1,\ldots,m.$$

Let $\psi(\mathbf{w},\bar{\mathbf{w}})$ be the directional derivative of function ϕ at approximate design vector $\mathbf{w} \in \mathbb{P}^n$ in the direction of design $\bar{\mathbf{w}} \in \mathbb{P}^n$, that is

$$\psi(\mathbf{w},\bar{\mathbf{w}}) = \lim_{\alpha\to 0^+} \frac{1}{\alpha}\left[\phi\left((1-\alpha)\mathbf{M}(\mathbf{w}) + \alpha\mathbf{M}(\bar{\mathbf{w}})\right) - \phi(\mathbf{M}(\mathbf{w}))\right].$$

Now put $\bar{\mathbf{w}} = \mathbf{e}^{(x)}$ for some $x \in \mathcal{X}$, i.e., $\mathbf{e}^{(x)}$ is the standard unit vector with its coordinate 1 corresponding to x. In this case we will use notation $\psi(\mathbf{w},x) = \psi(\mathbf{w},\mathbf{e}^{(x)})$.

It can be shown that if $\nabla\phi(\mathbf{M}(\mathbf{w}))$ exists then the directional derivative $\psi(\mathbf{w},x)$ can be computed as

$$\psi(\mathbf{w},x) = \mathbf{f}^\top(x)\nabla\phi(\mathbf{M}(\mathbf{w}))\mathbf{f}(x) - \mathrm{tr}\{\mathbf{M}(\mathbf{w})\nabla\phi(\mathbf{M}(\mathbf{w}))\}.$$

The following theorem is a general form of the so-called equivalence theorem for any concave optimality criterion and it can be proved for instance by using Karush-Kuhn-Tucker conditions (see [10]).

Theorem 1.1 *Let ϕ be a concave optimality criterion and let $\nabla\phi(\mathbf{M}(\mathbf{w}))$ exist for every $\mathbf{w} \in \mathbb{P}^n, x \in \mathcal{X}$. Then, approximate design $\mathbf{w}^*$ is optimal in $\mathbb{P}^n$ iff*

$$\max_{x\in\mathcal{X}} \psi(\mathbf{w}^*,x) \le 0.$$

It can be further shown that in case of an optimal design $\mathbf{w}^*$, there is the equality sign in 1.1, that is, $\max_{x\in\mathcal{X}} \psi(\mathbf{w}^*,x) = 0$, and the maximum occurs at the support points of design $\mathbf{w}^*$. For any design $\mathbf{w}$ that is non-optimal, it holds that $\max_{x\in\mathcal{X}} \psi(\mathbf{w},x) > 0$.

In this chapter we assume finite design spaces $|\mathcal{X}| = n$. However, this assumption is not inevitable for the equivalence theorem; in fact, equivalence theorem can be proved for any design space $\mathcal{X}$ that is compact.

In the specific case of the D-optimality criterion, for every $\mathbf{w} \in \mathbb{P}^n$ the gradient can be computed as $\nabla\phi_D(\mathbf{M}(\mathbf{w})) = 1/m \det^{1/m}(\mathbf{M}(\mathbf{w}))\mathbf{M}^{-1}(\mathbf{w})$, which implies the following form of Theorem 1.1.

Theorem 1.2 *Approximate design* $\mathbf{w}^*$ *is D-optimal in* $\mathbb{P}^n$ *iff*

$$\max_{x \in \mathcal{X}} d_x(\mathbf{w}^*) \leq m,$$

where

$$d_x(\mathbf{w}) = \mathbf{f}^\top(x)\mathbf{M}^{-1}(\mathbf{w})\mathbf{f}(x) \tag{2.7}$$

denotes the so-called *variance function* in design point x. In our work, we also use $\mathbf{d}(\mathbf{w})$ to denote the n-dimensional vector with components $d_x(\mathbf{w})$. In the standard linear regression model with regressors $\mathbf{f}(x)$, $x \in \mathcal{X}$, and homoscedastic uncorrelated errors, value $d_x(\mathbf{w})$ is proportional to the variance of the predicted response at point x under design $\mathbf{w}$, see, e.g., [23] or [1].

The criterion of D-optimality can be proven to be strictly concave on the set of regular information matrices. This implies that the optimal information matrix is unique, in contrast to a D-optimal design, which is not unique in general, but the set of D-optimal designs is convex. Additionally, it holds that there exists a D-optimal design containing at most $m(m+1)/2$ supporting points (see, e.g., [14]).

Theorem 1.2 justifies the use of the following vertex-direction procedure to find the approximate D-optimal design. This algorithm can be employed in the standard case as well as in case of the linearly restricted designs. We focus on the latter and thus optimize on the set of constrained designs $\mathbb{Q}^n$. We remark that we describe this method (see [14]) for its straightforwardness and simplicity; there are other more complex and more efficient methods listed in Section 3.1 of Chapter I.

Assume there exists a regular design $\mathbf{w} \in \mathbb{Q}^n$. Let $\{\alpha_t\}_{t=0}^{\infty}$ be a sequence of positive constants such that $\lim_{t\to\infty} \alpha_t = 0$ and $\sum_{t=0}^{\infty} \alpha_t = \infty$. Now, starting in a permissible design $\mathbf{w}^{(0)}$, we can define an iterative procedure

$$\mathbf{w}^{(t+1)} = (1-\alpha_t)\mathbf{w}^{(t)} + \alpha_t \mathbf{w}_t^*, \quad t = 0, 1, 2, \ldots,$$

where

$$\mathbf{w}_t^* \in \arg\max_{\mathbf{w} \in \mathbb{Q}^n} \sum_{x \in \mathcal{X}} w_x d_x(\mathbf{w}^{(t)}), \tag{2.8}$$

where $d_x(.)$ is the variance function defined by (2.7).

In every iteration of the vertex-direction algorithm, we need to solve the linear programming subproblem (2.8). This can be done for example using the simplex method. Although the problem of linear programming is not difficult, it can still significantly slow down the algorithm (compared to the standard case with no restrictions). Moreover, all vertex-direction methods are considered to be rather slow and can be easily outperformed by some of the competing methods, such as multiplicative algorithms or semidefinite programming. Their advantage is that they are universal, easy to implement and their convergence to the global optimum can be proved.

1.3 Barycentric algorithm for linearly constrained designs

For an overview of accessible algorithms for linearly constrained designs, we refer the reader to Section 3.1 of Chapter I. In this chapter, we focus on the barycentric multiplicative algorithm as proposed in [31], which represents a starting point for the results derived in the sequel. It concentrates on the criterion of D-optimality as defined in (2.6). Let us now discuss the main idea and some of the properties of the algorithm.

Let $\{\mathbf{q}_{\tilde{x}} : \tilde{x} \in \tilde{\mathcal{X}}\}$, where $\tilde{\mathcal{X}}$ is a finite ordered set with $\tilde{n}$ elements, be the set of all extreme vectors of $\mathbb{Q}^n$. Let $\mathbb{P}^{\tilde{n}}$ be a probability simplex of $\tilde{n}$-dimensional standard designs constrained only by (2.1) . The barycentric algorithm works by alternating between the "primary" set $\mathbb{Q}^n$ and the "secondary" set $\mathbb{P}^{\tilde{n}}$, so we should keep the difference in mind.

Denote $\tilde{\mathcal{H}} = \{\mathbf{M}(\mathbf{q}_{\tilde{x}})\}_{\tilde{x}\in\tilde{\mathcal{X}}}$. For a design $\tilde{\mathbf{w}} \in \mathbb{P}^{\tilde{n}}$, the amount of information obtained from the entire experiment is proportional to the information matrix

$$\mathbf{M}_{\tilde{\mathcal{H}}}(\tilde{\mathbf{w}}) = \sum_{\tilde{x}\in\tilde{\mathcal{X}}} \tilde{w}_{\tilde{x}} \mathbf{M}(\mathbf{q}_{\tilde{x}}).$$

Then, we can define $D_{\tilde{\mathcal{H}}}$-optimal design as any design $\tilde{\mathbf{w}}^* \in \mathbb{P}^{\tilde{n}}$ that maximizes $\phi_D(\mathbf{M}_{\tilde{\mathcal{H}}}(\tilde{\mathbf{w}}))$. This problem belongs to the class of so-called *generalized D-optimality* problems, where the information matrix is a convex combination of matrices of ranks not necessarily equal to 1 (in contrast to the standard case (2.3)). From this point of view, one can apply all the properties of generalized D-optimality (see [36], [74]), e.g.:

- There exists at least one $D_{\tilde{\mathcal{H}}}$-optimal design, the $D_{\tilde{\mathcal{H}}}$-optimal information matrix is non-singular and unique.

- There exists a simple formula for a lower bound on the $D_{\tilde{\mathcal{H}}}$-efficiency of a given design.
- There exist simple rules for identifying design points $\tilde{x}$ that do not support any $D_{\tilde{\mathcal{H}}}$-optimal design.

Moreover, for computing $D_{\tilde{\mathcal{H}}}$-optimal designs, there is a straightforward generalization of the standard multiplicative algorithm, for which it is proved that the sequence of information matrices corresponding to individual iterations converges monotonically to the $D_{\tilde{\mathcal{H}}}$-optimal information matrix ([85]).

The direct application of the multiplicative algorithm for $D_{\tilde{\mathcal{H}}}$-optimality requires operating with $\tilde{n}$ matrices $\mathbf{M}(\mathbf{q}_{\tilde{x}})$ of size $m \times m$, which can be numerically very difficult. The idea in [31] is that the multiplicative transformation operating on $\mathbb{P}^{\tilde{n}}$ can be used only implicitly, with all the actual computations performed in the much smaller space $\mathbb{P}^n$.

The key task is to transform design $\mathbf{w} \in \mathbb{Q}^n$ into design $\tilde{\mathbf{w}} \in \mathbb{P}^{\tilde{n}}$, that is, to find a suitable formula for *barycentric coordinates* $\tilde{w}(\mathbf{w}) \in \mathbb{P}^{\tilde{n}}$ expressing a linearly constrained design $\mathbf{w} \in \mathbb{Q}^n$ as a convex combination of the extreme designs $\mathbf{q}_{\tilde{x}}$, $\tilde{x} \in \tilde{\mathcal{X}}$.

The general formula of the barycentric updating rule is

$$\mathbf{T}^B(\mathbf{w}) = m^{-1}\mathbf{D}(\mathbf{w})\mathbf{d}(\mathbf{w}), \tag{2.9}$$

where $\mathbf{D}(\mathbf{w}) = \sum_{\tilde{x}\in\tilde{\mathcal{X}}} \tilde{w}_{\tilde{x}}(\mathbf{w})\mathbf{q}_{\tilde{x}}\mathbf{q}_{\tilde{x}}^\top$ and $\tilde{w} : \mathbb{Q}^n \to \mathbb{P}^{\tilde{n}}$ is a continuous function, such that $\tilde{w}(\mathbf{w})$ represents barycentric coordinates of the constrained design $\mathbf{w}$.

The barycentric updating rule first takes a design $\mathbf{w} \in \mathbb{Q}^n$ and finds its barycentric coordinates $\tilde{w}(\mathbf{w})$ in $\mathbb{P}^{\tilde{n}}$. Design $\tilde{w}(\mathbf{w})$ is then transformed using the multiplicative algorithm in the space $\mathbb{P}^{\tilde{n}}$, and finally put back into the original space $\mathbb{P}^n$ by expressing the result in the standard coordinates again. All of this is included in the formula (2.9).

The monotonic convergence of the sequence $\{\mathbf{M}(\mathbf{w}^{(t)})\}_{t=1}^{\infty}$, $\mathbf{w}^{(t+1)} = \mathbf{T}^B(\mathbf{w}^{(t)})$, to some matrix $\mathbf{M}^\infty$ can be proved. However, for the general barycentric algorithm it is not proved that the matrix $\mathbf{M}^\infty$ is D-optimal.

For any $\epsilon > 0$, we define

$$h_m(\epsilon) = m\left(1 + \frac{\epsilon}{2} - \frac{\sqrt{\epsilon(4+\epsilon-4/m)}}{2}\right). \tag{2.10}$$

The following theorem is not inevitable for the barycentric algorithm itself, but can be very useful: The inequality 1.3 provides a lower bound on the efficiency of any permissible design, which can be used to adjust a reasonable efficiency-based stopping rule. The inequality 1.3 can be used to accelerate the algorithm, since it allows us to completely remove unnecessary design points.

Theorem 1.3 *([31]) Let $\mathbf{Q} = (\mathbf{q}_{\tilde{x}})_{\tilde{x}\in\tilde{\mathcal{X}}}$ be the $n\times\tilde{n}$ matrix composed of all extreme vectors of $\mathbb{Q}^n$. Let $\mathbf{w} \in \mathbb{Q}^n$ be a regular design and let $\epsilon = \max_{\tilde{x}\in\tilde{\mathcal{X}}}(\mathbf{Q}^\top \mathbf{d}(\mathbf{w}))_{\tilde{x}} - m$. Let $\mathbf{w}^* \in \mathbb{Q}^n$ be a D-optimal constrained design. Then,*

$$\mathrm{eff}_D(\mathbf{w}|\mathbf{w}^*) \geq \frac{m}{m+\epsilon}.$$

Furthermore, $w_x^* = 0$ for any $x \in \mathcal{X}$ satisfying

$$\max_{\tilde{x}\in\tilde{\mathcal{X}}:(\mathbf{q}_{\tilde{x}})_x>0} (\mathbf{Q}^\top \mathbf{d}(\mathbf{w}))_{\tilde{x}} < h_m(\epsilon).$$

2 Size- and cost-constrained designs

The research of linearly constrained designs has its place in the area of optimal experimental design. Situations with multiple constraints do occur in practice; they can represent limitations on the availability of various resources consumed by the experiment. One of the simplest cases is the presence of one more linear constraint in addition to the "size" restriction (2.1), as stated in Section 1.1. This additional constraint can be viewed as a restriction on the "cost" of the experiment, yielding so-called size- and cost-constrained designs that we deal with in this chapter.

To start with, we clarify how these two conditions are motivated by the real-life limitations. The limit on the "size" of the experiment is a natural requirement and the research related to the computation of optimal experimental designs used to focus mostly on these single-linear-constraint problems. Now, assume for a while that we are about to conduct an exact experiment consisting of at most N trials. If we denote $w_x = N_x/N$, where N_x

is the number of the trials performed in x, the size constraint can be written as

$$\sum_{x \in \mathcal{X}} w_x \leq 1. \tag{2.11}$$

Note that there is in fact no need to strictly require equality (2.1) and it is more natural to use inequality (2.11) instead.

Now assume that each trial is associated with a known cost $C_x > 0$ depending on the corresponding design point $x \in \mathcal{X}$, and the total cost of the experiment cannot exceed a given budget $B > 0$. For each $x \in \mathcal{X}$, let $c_x = \frac{N}{B} C_x$ be the normalized cost. Then, the total cost constraint can be written in the form

$$\sum_{x \in \mathcal{X}} c_x w_x \leq 1. \tag{2.12}$$

If the values c_x are the same for all $x \in \mathcal{X}$ then the design has, effectively, only a single constraint. However, (2.11) and (2.12) may be both relevant if the costs of trials are unequal, which often occurs in practice.

For instance, in an application described in [82], the design space $\mathcal{X}$ represents time, and the cost of conducting a trial is a non-constant function of the time when the observation is sampled. Similarly, in [52], the design space is the set of all combinations of factor levels, some of which are significantly more expensive than others.

In some situations, the interpretation of the coefficients c_x may be different from direct financial costs. For example, assume that each trial in $x \in \mathcal{X}$ consumes c_x volume units of a specific substrate, as in [87]. Then, the restriction on the total available volume of the substrate can be captured by an inequality of the form (2.12). Yet another example of constraints of the type (2.12) can be found in [19], [20] and [57], in which the design space corresponds to treatment doses and the "costs" represent penalties for doses with low efficacy and high toxicity.

Our goal is to propose a method of constructing a D-optimal design $\mathbf{w}^*$ in the set of all approximate designs that satisfy both the size and the cost constraints (2.11) and (2.12), that is,

$$\mathbf{w}^* \in \arg\max_{\mathbf{w}} \{\phi_D(\mathbf{w}) : \mathbf{w} \geq \mathbf{0}_n,\ \mathbf{1}_n^\top \mathbf{w} \leq 1,\ \mathbf{c}^\top \mathbf{w} \leq 1\}, \tag{2.13}$$

where $\mathbf{0}_n$ is the n-dimensional zero vector, $\mathbf{1}_n$ is the n-dimensional vector of ones, $\mathbf{c}$ is the vector of normalized cost coefficients and D-optimality criterion ϕ_D is given by (2.6).

We assume the linear model (1.1). Note that the vectors $\mathbf{f}(x)$, $x \in \mathcal{X}$, can also be the gradients of the mean-value function of a non-linear regression model. Then, the solution of (2.13) is a size- and cost-constrained locally D-optimal design (see, e.g., Chapter 17 in [1] or Chapter 5 in [60]).

Note that for the problem (2.13), the set of permissible designs is nonempty and compact, therefore the continuity of ϕ implies that (2.13) has at least one optimal solution $\mathbf{w}^*$.

The assumptions of regularity and properties of ϕ_D imply that

$$\arg\max_{\mathbf{w}}\{\phi_D(\mathbf{w}) : \mathbf{w} \geq \mathbf{0}_n, \mathbf{1}_n^\top \mathbf{w} \leq 1\} = \arg\max_{\mathbf{w}}\{\phi_D(\mathbf{w}) : \mathbf{w} \geq \mathbf{0}_n, \mathbf{1}_n^\top \mathbf{w} = 1\}. \quad (2.14)$$

Thus, computing a D-optimal design under (2.11) is equivalent to computing a standard D-optimal design, for which there exist many very efficient algorithms. Similarly, since $c_x > 0$ for all $x \in \mathcal{X}$, we have

$$\arg\max_{\mathbf{w}}\{\phi_D(\mathbf{w}) : \mathbf{w} \geq \mathbf{0}_n, \mathbf{c}^\top \mathbf{w} \leq 1\} = \arg\max_{\mathbf{w}}\{\phi_D(\mathbf{w}) : \mathbf{w} \geq \mathbf{0}_n, \mathbf{c}^\top \mathbf{w} = 1\}, \quad (2.15)$$

which is a problem that can be transformed to (2.14) using a suitable change of regressors $\mathbf{f}(x)$, $x \in \mathcal{X}$; see, e.g., [22] or [1]. However, constructing an approximate optimal design under simultaneous size and cost constraints is less straightforward, as we discuss next.

Proposition 2.1 *Let $\mathbf{w}^s$ be optimal for the size-constrained problem* (2.14) *and let $\mathbf{w}^c$ be optimal for the cost-constrained problem* (2.15)*. Then, either $\mathbf{w}^s$ satisfies the cost constraint* (2.12)*, in which case it solves* (2.13)*, or $\mathbf{w}^c$ satisfies the size constraint* (2.11)*, in which case it solves* (2.13)*, or there exists a solution $\mathbf{w}^{**}$ of the problem* (2.13) *that satisfies both*

$$\begin{aligned} \sum_{x \in \mathcal{X}} w_x^{**} &= 1, \text{ and} \\ \sum_{x \in \mathcal{X}} c_x w_x^{**} &= 1. \end{aligned}$$

The previous discussion and Proposition 2.1 imply that once we are able to find a solution of the "equality" size- and cost-constrained problem

$$\mathbf{w}^* \in \arg\max_{\mathbf{w}} \{\phi(\mathbf{w}) : \mathbf{w} \geq \mathbf{0}_n, \mathbf{1}_n^\top \mathbf{w} = 1, \mathbf{c}^\top \mathbf{w} = 1\}, \tag{2.16}$$

we will have an exhaustive method of solving the practically usually more meaningful "inequality" size- and cost-constrained problem (2.13). That is why we concentrate on the problem (2.16) in the sequel and we refer to it as to the problem of D-optimal size- and cost-constrained designs. The set of designs permissible for 2.16-2.16 is illustrated in Figure 2.1.

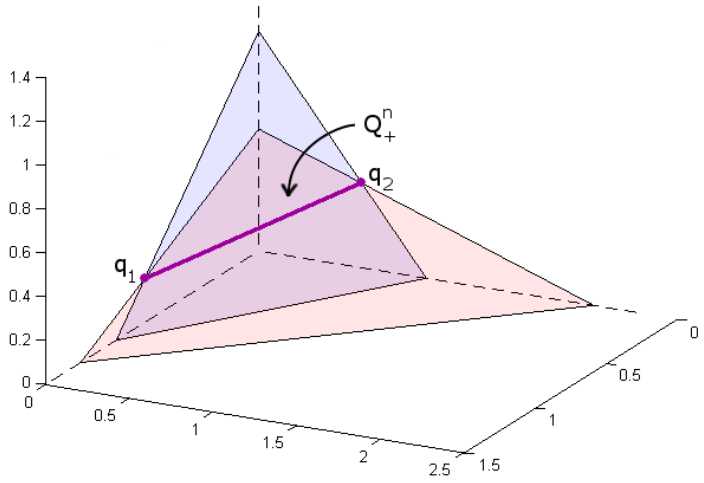


Figure 2.1: Illustration of size and cost constraints in case $|\mathcal{X}| = 3$. The blue triangle (2-simplex) represents designs permissible for 2.16, while the red triangle represents designs permissible for 2.16. Set $\mathbb{Q}_+^n$ (i.e., the set of designs satisfying both 2.16 and 2.16) is denoted by the purple line segment with vertices $\mathbf{q}_1, \mathbf{q}_2$.

Let $\mathbb{Q}^n$ denote the set of all permissible solutions of (2.16). If $\mathbb{Q}^n$ is not empty, it is a convex and compact polyhedron. At the beginning of Section 3, we add some natural assumptions on the normalized costs c_x, $x \in \mathcal{X}$, that guarantee $\mathbb{Q}^n \neq \emptyset$. Then, it is possible to prove a simple "equivalence theorem" for the D-optimal size- and cost-constrained design solving (2.16), and some other theoretical properties, cf. Section 3.

Analytic solutions of (2.16) are possible only in the simplest cases (such as in the illustrative Example 2.1 at the end of this section). However, there are several general

methods that can be used to develop an efficient algorithm specialized to solve (2.16). Some of the most appropriate methods are listed in the introductory Chapter I.

Naturally, from the point of view of applications, the primary objective is to find an exact experimental design with the number of elements bounded from above by some N. When considering "inequalities" problem (2.13), the simplest solution is to "round down" an approximate D-optimal design $\mathbf{w}^*$ by replacing the values w_x^* with $\lfloor N w_x^* \rfloor / N$ for all $x \in \mathcal{X}$, where $\lfloor \cdot \rfloor$ is the floor function. A more efficient solution can be obtained by using an excursion heuristic such as the one proposed in [32], with the result of the simple rounding taken as an initial permissible solution. Alternatively, it is possible to apply the heuristic method from [34] based on integer quadratic programming, which utilizes the information matrix of an approximate D-optimal design.

Before proceeding, we give a small example to illustrate some aspects of the problems (2.13) and (2.16), in particular the fact that for the equality-constrained D-optimal design problem (2.16) the values $c_x = 1$ play special roles.

Example 2.1 *Assume that* $n = 2$, $\mathbf{f}(1) = (1,0)^\top$, $\mathbf{f}(2) = (1,1)^\top$. *In this elementary model, it is simple to verify that for a design* $\mathbf{w} = (w_1, w_2)^\top$ *the criterion of D-optimality is proportional to* $\sqrt{w_1 w_2}$, *and the solutions of both* (2.13) *and* (2.16) *can be calculated analytically: the solution of* (2.13) *is*

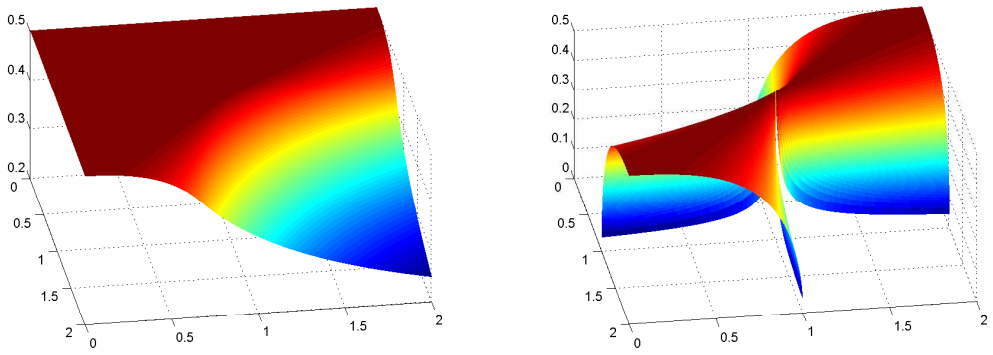
$$(w_1^*, w_2^*)^\top = \begin{cases} (0.5, 0.5)^\top & \text{if } c_1 + c_2 \leq 2, \\ \left(\frac{c_2 - 1}{c_2 - c_1}, \frac{c_1 - 1}{c_1 - c_2}\right)^\top & \text{if } c_1 + c_2 > 2 \text{ and } \frac{1}{2c_1} + \frac{1}{2c_2} > 1, \\ \left(\frac{1}{2c_1}, \frac{1}{2c_2}\right)^\top & \text{if } \frac{1}{2c_1} + \frac{1}{2c_2} \leq 1, \end{cases}$$

and the solution of (2.16) *is*

$$(w_1^*, w_2^*)^\top = \begin{cases} (0.5, 0.5)^\top & \text{if } c_1 = c_2 = 1, \\ \left(\frac{c_2 - 1}{c_2 - c_1}, \frac{c_1 - 1}{c_1 - c_2}\right)^\top & \text{if } (c_1, c_2)^\top \in [(0,1) \times (1,2)] \cup [(1,2) \times (0,1)]. \end{cases}$$

In Figure 2.2, we plotted the values $\phi_D(w_1^*, w_2^*)$, *as they depend on the costs* c_1 *and* c_2. *For the inequality-constrained problem* (2.13) *illustrated in Figure 2.2a, the optimal criterial values are continuous and non-decreasing for decreasing costs, as expected.*

However, the optimal criterial values of the equality-constrained problem (2.16) *behave differently. First, in Figure 2.2b, the domain of the function is restricted to* $\mathcal{C} = [(0,1) \times$



(a) $w_1 + w_2 \leq 1,\ c_1 w_1 + c_2 w_2 \leq 1$ (b) $w_1 + w_2 = 1,\ c_1 w_1 + c_2 w_2 = 1$

Figure 2.2: The values of ϕ_D corresponding to the D-optimal designs with $n = 2$ design points, regressors $\mathbf{f}(1) = (1, 0)^\top$, $\mathbf{f}(2) = (1, 1)^\top$ and the costs c_1, c_2 varying in $(0, 2)$.

$(1, 2)] \cup [(1, 2) \times (0, 1)] \cup \{(1, 1)\}$ *because for couples* $(c_1, c_2) \in [(0, 2) \times (0, 2)] \setminus \mathcal{C}$ *there is no permissible solution of* (2.16) *or the optimal information matrix is singular. Moreover, observe that it is not possible to continuously extend the function in Figure 2.2b to the point* $(1, 1)$*, although for* $c_1 = c_2 = 1$ *the optimization problem* (2.16) *is meaningful with a unique solution. This phenomenon suggests that the points* $x \in \mathcal{X}$ *such that* $c_x = 1$ *play a special role. Finally, note that in Figure 2.2b the optimal criterial value can strictly decrease with decreasing costs.*

3 Theoretical results for approximate D-optimal size- and cost-constrained designs

If $c_x \leq 1$ for all $x \in \mathcal{X}$, i.e., if the costs of all trials are "low", then every design satisfying the size constraint (2.11) satisfies also the cost constraint (2.12), that is, an optimal solution of (2.13) can be found as a solution of (2.14). Analogously, if $c_x \geq 1$ for all $x \in \mathcal{X}$, that is, if the costs of all trials are "high", then every design satisfying the cost constraint (2.12) satisfies also the size constraint (2.11), that is, an optimal solution of (2.13) can be found as a solution of (2.15). Therefore, we can assume that there exist $x_- \in \mathcal{X}$ such that $c_{x_-} < 1$ and $x_+ \in \mathcal{X}$ such that $c_{x_+} > 1$.

Let $\mathcal{X}_+ = \{x \in \mathcal{X} : c_x > 1\}$, $\mathcal{X}_- = \{x \in \mathcal{X} : c_x < 1\}$, $\mathcal{X}_0 = \{x \in \mathcal{X} : c_x = 1\}$, and let n_+, n_-, n_0 be the sizes of these sets. Clearly, our assumptions mean that $\mathcal{X}_+ \neq \emptyset$ and

$\mathcal{X}_- \neq \emptyset$. For simplicity, in Sections 3 and 4, we will assume that $\mathcal{X}_0 \neq \emptyset$; all results can be modified in a straightforward way for the (simpler) case $\mathcal{X}_0 = \emptyset$.

Recall that the set of all permissible designs of (2.16) is denoted by $\mathbb{Q}^n$. We will use the symbol $\mathbb{Q}_+^n$ to denote the set of all designs $\mathbf{w} \in \mathbb{Q}^n$ with all components strictly positive. We will use the symbol $\mathbb{Q}_r^n$ to denote the set of all designs $\mathbf{w} \in \mathbb{Q}^n$ with a non-singular information matrix $\mathbf{M}(\mathbf{w})$. Clearly, the regularity assumptions imply $\mathbb{Q}_+^n \subseteq \mathbb{Q}_r^n$.

Define $\delta_{x_+} = c_{x_+} - 1 > 0$ for $x_+ \in \mathcal{X}_+$ and $\delta_{x_-} = 1 - c_{x_-} > 0$ for $x_- \in \mathcal{X}_-$. Consider the design $\mathbf{w}^{(0)}$ with components

$$w_{x_+}^{(0)} = \tilde{n}^{-1} \sum_{x_- \in \mathcal{X}_-} \frac{\delta_{x_-}}{\delta_{x_+} + \delta_{x_-}}; \quad x_+ \in \mathcal{X}_+, \tag{2.17}$$

$$w_{x_-}^{(0)} = \tilde{n}^{-1} \sum_{x_+ \in \mathcal{X}_+} \frac{\delta_{x_+}}{\delta_{x_+} + \delta_{x_-}}; \quad x_- \in \mathcal{X}_-, \tag{2.18}$$

$$w_{x_0}^{(0)} = \tilde{n}^{-1}; \quad x_0 \in \mathcal{X}_0, \tag{2.19}$$

where $\tilde{n} = n_+ n_- + n_0$. It is straightforward to verify that $\mathbf{w}^{(0)}$ is permissible for (2.16), i.e., $\mathbb{Q}^n \neq \emptyset$. Moreover, $\mathbf{w}^{(0)} \in \mathbb{Q}_+^n$, that is, $\mathbf{w}^{(0)} \in \mathbb{Q}_r^n$. Hence, the information matrix of the design optimal for (2.16) is non-singular. The strict concavity of $\det^{1/m}(\cdot)$ on the set of all positive definite matrices (see [61]) guarantees that the optimal information matrix is unique.

For any $x_+ \in \mathcal{X}_+$, $x_- \in \mathcal{X}_-$ let

$$\mathbf{q}^{(x_+,x_-)} = \frac{\delta_{x_-}}{\delta_{x_+} + \delta_{x_-}} \mathbf{e}^{(x_+)} + \frac{\delta_{x_+}}{\delta_{x_+} + \delta_{x_-}} \mathbf{e}^{(x_-)}$$

and for any $x_0 \in \mathcal{X}_0$ let $\mathbf{q}^{(x_0)} = \mathbf{e}^{(x_0)}$, where $\mathbf{e}^{(x)}$, $x \in \mathcal{X}$, are standard unit vectors. It is simple to show that

$$\{\mathbf{q}^{(x_+,x_-)} : x_+ \in \mathcal{X}_+, x_- \in \mathcal{X}_-\} \cup \{\mathbf{q}^{(x_0)} : x_0 \in \mathcal{X}_0\}$$

is the set of all extreme vectors of the polytope $\mathbb{Q}^n$. In fact, the design $\mathbf{w}^{(0)}$ defined by (2.17)-(2.19) is the "center of mass" of these extreme vectors if they are assigned equal weights. Note also that the form of the extreme vectors of $\mathbb{Q}^n$ and the Caratheodory theorem imply that for any design $\mathbf{w} \in \mathbb{Q}^n$ there exists a size- and cost-constrained design with the same information matrix and the "support" of at most $m(m+1)$ non-zero coordinates.

The following two definitions are used in the sequel to simplify the notation. For any $\mathbf{w} \in \mathbb{Q}^n$, we define

$$s^{\delta}(\mathbf{w}) = \sum_{x_+ \in \mathcal{X}_+} \delta_{x_+} w_{x_+} = \sum_{x_- \in \mathcal{X}_-} \delta_{x_-} w_{x_-}. \tag{2.20}$$

The second equality in (2.20) follows directly from (2.11) and (2.12). For any $\mathbf{w} \in \mathbb{Q}_r^n$ and $x_+ \in \mathcal{X}_+$, $x_- \in \mathcal{X}_-$, we define the weighted variances

$$\tilde{d}_{x_+x_-}(\mathbf{w}) = \frac{\delta_{x_+} d_{x_-}(\mathbf{w}) + \delta_{x_-} d_{x_+}(\mathbf{w})}{\delta_{x_+} + \delta_{x_-}}. \tag{2.21}$$

Theorem 3.1 *Let $\mathbf{w} \in \mathbb{Q}_r^n$. Then, the following three statements are equivalent:*
(i) The design $\mathbf{w}$ is D-optimal in $\mathbb{Q}^n$.
(ii) There exists $h \in \mathbb{R}$ such that

$$d_x(\mathbf{w}) \leq m + h(c_x - 1) \text{ for all } x \in \mathcal{X}.$$

(iii) $\max_{x_0 \in \mathcal{X}_0} d_{x_0}(\mathbf{w}) \leq m$ *and*

$$\max_{x_+ \in \mathcal{X}_+} \frac{d_{x_+}(\mathbf{w}) - m}{\delta_{x_+}} + \max_{x_- \in \mathcal{X}_-} \frac{d_{x_-}(\mathbf{w}) - m}{\delta_{x_-}} \leq 0.$$

Proof
$(i) \Leftrightarrow (ii)$ Consider D-optimality criterion defined by $\phi_D(\mathbf{w}) = \ln\det(\mathbf{M}(\mathbf{w}))$. Karush-Kuhn-Tucker conditions and the concavity of $\phi_D(\mathbf{w})$ imply the following necessary and sufficient condition for maximum at $\mathbf{w}$: There exist constants $\lambda, \mu \in \mathbb{R}$ and $\nu_1, \ldots, \nu_n \geq 0$ such that for all $x \in \mathcal{X}$

$$\frac{\partial \phi_D(\mathbf{M}(\mathbf{w}))}{\partial w_x} = \lambda + \mu c_x - \nu_x,$$

which is equivalent to

$$d_x(\mathbf{w}) = \lambda + \mu c_x - \nu_x, \tag{2.22}$$

since

$$\begin{aligned} \frac{\partial \phi_D(\mathbf{M}(\mathbf{w}))}{\partial w_x} &= \frac{\partial \ln(\det(\mathbf{M}(\mathbf{w})))}{\partial w_x} = \sum_{i,j=1}^{m} \frac{\partial \ln(\det(\mathbf{M}(\mathbf{w})))}{\partial M_{ij}(\mathbf{w})} \frac{\partial M_{ij}(\mathbf{w})}{\partial w_x} \\ &= \sum_{i,j=1}^{m} \{\mathbf{M}^{-1}(\mathbf{w})\}_{ij} f_i(x) f_j(x) = \mathbf{f}(x)^{\top} \mathbf{M}^{-1}(\mathbf{w}) \mathbf{f}(x) = d_x(\mathbf{w}). \end{aligned}$$

By multiplying by w_x and summing through $x \in \mathcal{X}$ we obtain

$$\sum_x d_x(\mathbf{w}) w_x = \lambda \sum_x w_x + \mu \sum_x c_x w_x - \sum_x \nu_x w_x,$$

which yields $m = \lambda + \mu$. Substituting $\lambda = m - \mu$ into (2.22), we get $d_x(\mathbf{w}) = m + \mu(c_x - 1) - \nu_x$, which is equivalent to the part (ii) of the theorem.

$(ii) \Leftrightarrow (iii)$ The implication $(ii) \Rightarrow (iii)$ is straightforward to verify.

Note that for $x_0 \in \mathcal{X}_0$ the condition (iii) is equivalent to the condition (ii). For $x_+ \in \mathcal{X}_+, x_- \in \mathcal{X}_-$ the condition (iii) is equivalent to: For all $x_+ \in \mathcal{X}_+$ and $x_- \in \mathcal{X}_-$ we have $\tilde{d}_{x_+x_-}(\mathbf{w}) \leq m$. But it is simple to check that the values $\tilde{d}_{x_+x_-}(\mathbf{w})$ above have the weighted average equal to m, if the weights are chosen to be

$$\tilde{w}_{x_+x_-}(\mathbf{w}) = \frac{(\delta_{x_+} + \delta_{x_-})w_{x_+}w_{x_-}}{s^\delta(\mathbf{w})};\ x_+ \in \mathcal{X}_+, x_- \in \mathcal{X}_-.$$

$$\begin{aligned}
&\sum_{x_+\in\mathcal{X}_+}\sum_{x_-\in\mathcal{X}_-}\frac{(\delta_{x_+} + \delta_{x_-})w_{x_+}w_{x_-}}{s^\delta(\mathbf{w})}\tilde{d}_{x_+x_-}(\mathbf{w}) \\
= \ & \{s^\delta\}^{-1}(\mathbf{w})\left(\sum_{x_+} w_{x_+}d_{x_+}(\mathbf{w})\sum_{x_-} w_{x_-}\delta_{x_-} + \sum_{x_+} w_{x_+}\delta_{x_+}\sum_{x_-} w_{x_-}d_{x_-}(\mathbf{w})\right) \\
= \ & \sum_{x_+} w_{x_+}d_{x_+}(\mathbf{w}) + \sum_{x_-} w_{x_-}d_{x_-}(\mathbf{w}) = m.
\end{aligned}$$

Hence, the inequality in part (iii) must be in fact equality. Thus, the condition (ii) holds with $h = \max_{x_+\in\mathcal{X}_+}\delta_{x_+}^{-1}(d_{x_+}(\mathbf{w}) - m) = -\max_{x_-\in\mathcal{X}_-}\delta_{x_-}^{-1}(d_{x_-}(\mathbf{w}) - m)$. □

The following theorem is a particular case of Theorem 1.3 for size- and cost-constrained designs. It can be used with any permissible design $\mathbf{w} \in \mathbb{Q}_r^n$ to compute a lower bound for its efficiency and remove the points from $\mathcal{X}$ that cannot be in the support of any D-optimal size- and cost-constrained design.

Theorem 3.2 *Let $\mathbf{w} \in \mathbb{Q}_r^n$, let $\mathbf{w}^*$ be a design that solves* (2.16)*, and let*

$$\epsilon = \max\left(\max_{x_+\in\mathcal{X}_+, x_-\in X_-}\tilde{d}_{x_+x_-}(\mathbf{w}), \max_{x_0\in\mathcal{X}_0} d_{x_0}(\mathbf{w})\right) - m.$$

Then, $\mathrm{eff}_D(\mathbf{w}|\mathbf{w}^*) \geq \frac{m}{m+\epsilon}$*. Let $h_m(\epsilon)$ be defined by* (2.10)*. Then,*
(i) $\max_{x_-\in\mathcal{X}_-}\tilde{d}_{x_+x_-}(\mathbf{w}) < h_m(\epsilon)$ *for some $x_+ \in \mathcal{X}_+$ implies $w^*_{x_+} = 0$.*
(ii) $\max_{x_+\in\mathcal{X}_+}\tilde{d}_{x_+x_-}(\mathbf{w}) < h_m(\epsilon)$ *for some $x_- \in \mathcal{X}_-$ implies $w^*_{x_-} = 0$.*
(iii) $d_{x_0}(\mathbf{w}) < h_m(\epsilon)$ *for some $x_0 \in \mathcal{X}_0$ implies $w^*_{x_0} = 0$.*

The removal of redundant design points based on Theorem 3.2 can greatly enhance the speed of numerical methods for computing optimal designs, including the S&C algorithm derived in the next section.

4 The S&C algorithm

For the constraints (2.11) and (2.12), the barycentric transformation $\mathbf{T}^B : \mathbb{Q}_r^n \to \mathbb{Q}_r^n$ is defined by (2.9), where

$$\mathbf{D}(\mathbf{w}) = \sum_{\tilde{x}\in\tilde{\mathcal{X}}} \tilde{w}_{\tilde{x}}(\mathbf{w})\mathbf{q}_{\tilde{x}}\mathbf{q}_{\tilde{x}}^\top = \sum_{x_+\in\mathcal{X}_+}\sum_{x_-\in\mathcal{X}_-} \tilde{w}_{x_+x_-}(\mathbf{w})\mathbf{q}^{(x_+,x_-)}(\mathbf{q}^{(x_+,x_-)})^\top + \sum_{x_0\in\mathcal{X}_0} \tilde{w}_{x_0}(\mathbf{w})\mathbf{q}^{(x_0)}(\mathbf{q}^{(x_0)})^\top. \tag{2.23}$$

In (2.23), the functions $\tilde{w}_{x_+x_-} : \mathbb{Q}^n \to \mathbb{R}$; $x_+ \in \mathcal{X}_+, x_- \in \mathcal{X}_-$, and $\tilde{w}_{x_0} : \mathbb{Q}^n \to \mathbb{R}$; $x_0 \in \mathcal{X}_0$, are the barycentric coordinates, that is, they are non-negative and satisfy

$$\sum_{x_+\in\mathcal{X}_+}\sum_{x_-\in\mathcal{X}_-} \tilde{w}_{x_+x_-}(\mathbf{w}) + \sum_{x_0\in\mathcal{X}_0} \tilde{w}_{x_0}(\mathbf{w}) = 1, \tag{2.24}$$

$$\sum_{x_+\in\mathcal{X}_+}\sum_{x_-\in\mathcal{X}_-} \tilde{w}_{x_+x_-}(\mathbf{w})\mathbf{q}^{(x_+,x_-)} + \sum_{x_0\in\mathcal{X}_0} \tilde{w}_{x_0}(\mathbf{w})\mathbf{q}^{(x_0)} = \mathbf{w} \tag{2.25}$$

for all $\mathbf{w} \in \mathbb{Q}^n$. For the general theory to be applicable, the barycentric coordinates must be chosen such that they are continuous on $\mathbb{Q}^n$ and strictly positive for any $\mathbf{w} \in \mathbb{Q}_+^n$.

The barycentric coordinates are not uniquely defined, and not all choices of barycentric coordinates are equally good. It turns out that for the problem (2.16) a suitable definition of barycentric coordinates of $\mathbf{w} \in \mathbb{Q}^n$ is

$$\tilde{w}_{x_+x_-}(\mathbf{w}) = \begin{cases} \dfrac{(\delta_{x_+} + \delta_{x_-})w_{x_+}w_{x_-}}{s^\delta(\mathbf{w})} & \text{if } s^\delta(\mathbf{w}) > 0, \\ 0 & \text{if } s^\delta(\mathbf{w}) = 0, \end{cases} \tag{2.26}$$

$$\tilde{w}_{x_0}(\mathbf{w}) = w_{x_0}, \tag{2.27}$$

for all $x_+ \in \mathcal{X}_+$, $x_- \in \mathcal{X}_-$, and $x_0 \in \mathcal{X}_0$.

We summarize the properties of the functions defined by (2.26)-(2.27) in the following proposition.

Proposition 4.1 *Let $x_+ \in \mathcal{X}_+$, $x_- \in \mathcal{X}_-$, and $x_0 \in \mathcal{X}_0$. The functions $\tilde{w}_{x_+x_-} : \mathbb{Q}^n \to \mathbb{R}$, and $\tilde{w}_{x_0} : \mathbb{Q}^n \to \mathbb{R}$ defined by* (2.26) *and* (2.27) *are non-negative, continuous on $\mathbb{Q}^n$, and positive on $\mathbb{Q}_+^n$. Moreover, for any $\mathbf{w} \in \mathbb{Q}^n$ the functions satisfy* (2.24) *and* (2.25)*.*

Let us derive the form of the barycentric transformation for any size- and cost-constrained design $\mathbf{w} \in \mathbb{Q}_+^n$. The diagonal element of the update matrix (2.23) corre-

sponding to $y_+ \in \mathcal{X}_+$ is

$$\begin{aligned}(\mathbf{D}(\mathbf{w}))_{y_+y_+} &= \sum_{x_+\in\mathcal{X}_+}\sum_{x_-\in\mathcal{X}_-}\tilde{w}_{x_+x_-}(\mathbf{w})(q_{y_+}^{(x_+,x_-)})^2 + \sum_{x_0\in\mathcal{X}_0}\tilde{w}_{x_0}(\mathbf{w})(q_{y_+}^{(x_0)})^2 \\ &= \frac{w_{y_+}}{s^\delta(\mathbf{w})}\sum_{x_-\in\mathcal{X}_-}\frac{w_{x_-}\delta_{x_-}^2}{\delta_{y_+}+\delta_{x_-}}. \end{aligned} \tag{2.28}$$

An analogous formula is valid for $y_- \in \mathcal{X}_-$. For $y_+ \in \mathcal{X}_+$ and $y_- \in \mathcal{X}_-$ the element (y_+, y_-) of the update matrix is

$$\begin{aligned}(\mathbf{D}(\mathbf{w}))_{y_+y_-} &= \sum_{x_+\in\mathcal{X}_+}\sum_{x_-\in\mathcal{X}_-}\tilde{w}_{x_+x_-}(\mathbf{w})q_{y_+}^{(x_+,x_-)}q_{y_-}^{(x_+,x_-)} + \sum_{x_0\in\mathcal{X}_0}\tilde{w}_{x_0}(\mathbf{w})q_{y_+}^{(x_0)}q_{y_-}^{(x_0)} \\ &= \frac{w_{y_+}w_{y_-}}{s^\delta(\mathbf{w})}\frac{\delta_{y_+}\delta_{y_-}}{\delta_{y_+}+\delta_{y_-}}. \end{aligned} \tag{2.29}$$

For $y_0 \in \mathcal{X}_0$, the diagonal element of the update matrix corresponding to y_0 is

$$\begin{aligned}(\mathbf{D}(\mathbf{w}))_{y_0y_0} &= \sum_{x_+\in\mathcal{X}_+}\sum_{x_-\in\mathcal{X}_-}\tilde{w}_{x_+x_-}(\mathbf{w})(q_{y_0}^{(x_+,x_-)})^2 + \sum_{x_0\in\mathcal{X}_0}\tilde{w}_{x_0}(\mathbf{w})(q_{y_0}^{(x_0)})^2 \\ &= w_{y_0}. \end{aligned} \tag{2.30}$$

It can be easily checked that all other elements of the update matrix are equal to zero. Equalities (2.28)-(2.30) yield the following form of the barycentric updating rule for $\mathbf{w} \in \mathbb{Q}_+^n$:

$$\mathbf{T}^B(\mathbf{w}) = \mathbf{w} \odot \mathbf{d}^\pi(\mathbf{w}), \tag{2.31}$$

where $\odot$ is the componentwise multiplication and the components of $\mathbf{d}^\pi(\mathbf{w})$ are given by

$$d_{x_+}^\pi(\mathbf{w}) = \frac{\sum_{x_-\in\mathcal{X}_-} w_{x_-}\delta_{x_-}\tilde{d}_{x_+x_-}(\mathbf{w})}{ms^\delta(\mathbf{w})};\ x_+ \in \mathcal{X}_+, \tag{2.32}$$

$$d_{x_-}^\pi(\mathbf{w}) = \frac{\sum_{x_+\in\mathcal{X}_+} w_{x_+}\delta_{x_+}\tilde{d}_{x_+x_-}(\mathbf{w})}{ms^\delta(\mathbf{w})};\ x_- \in \mathcal{X}_-, \tag{2.33}$$

$$d_{x_0}^\pi(\mathbf{w}) = \frac{d_{x_0}(\mathbf{w})}{m};\ x_0 \in \mathcal{X}_0. \tag{2.34}$$

Note that the barycentric transformation uses the numbers $\tilde{d}_{x_+x_-}(\mathbf{w})$, $x_+ \in \mathcal{X}_+, x_- \in \mathcal{X}_-$ and $d_{x_0}(\mathbf{w})$, $x_0 \in \mathcal{X}_0$, which can be directly re-used for computing the lower bound on the design efficiency and for the deletion rules given in Theorem 3.2.

For computations, it may be useful to rewrite the barycentric transformations (2.32)-(2.34) to the following form. For vectors $\mathbf{a} \in \mathbb{R}^{n_+}$ and $\mathbf{b} \in \mathbb{R}^{n_-}$ let $\mathbf{a} \oplus \mathbf{b}^\top$ be the $n_+ \times n_-$ matrix with components $a_{x_+} + b_{x_-}$. This operation can be implemented as a stand-alone

function or using the Kronecker multiplication. For any vector $\mathbf{g} \in \mathbb{R}^n$, let $\mathbf{g}_+ \in \mathbb{R}^{n_+}$, $\mathbf{g}_- \in \mathbb{R}^{n_-}$, and $\mathbf{g}_0 \in \mathbb{R}^{n_0}$ denote the sub-vectors of $\mathbf{g}$ corresponding to $x_+ \in \mathcal{X}_+$, $x_- \in \mathcal{X}_-$, and $x_0 \in \mathcal{X}_0$, respectively. Let $\mathbf{w} \in \mathbb{Q}^n$ be a permissible design and let $\delta = (\delta_1, ..., \delta_n)^\top$. The $n_+ \times n_-$ matrix Δ with components defined in (2.21) is

$$\Delta = \left[(\mathbf{d}_+(\mathbf{w}) \oslash \delta_+) \oplus (\mathbf{d}_-(\mathbf{w}) \oslash \delta_-)^\top\right] \oslash \left[(\mathbf{1}_{n_+} \oslash \delta_+) \oplus (\mathbf{1}_{n_-} \oslash \delta_-)^\top\right],$$

where $\oslash$ denotes the componentwise division. Then, the barycentric transformation (2.31) can be written in the form

$$\mathbf{T}_+^B(\mathbf{w}) = \frac{1}{ms^\delta(\mathbf{w})}\mathbf{w}_+ \odot [\Delta(\mathbf{w}_- \odot \delta_-)], \tag{2.35}$$

$$\mathbf{T}_-^B(\mathbf{w}) = \frac{1}{ms^\delta(\mathbf{w})}\mathbf{w}_- \odot [\Delta^\top(\mathbf{w}_+ \odot \delta_+)], \tag{2.36}$$

$$\mathbf{T}_0^B(\mathbf{w}) = \frac{1}{m}\mathbf{w}_0 \odot \mathbf{d}_0(\mathbf{w}). \tag{2.37}$$

In matrix-based software such as Matlab or R, computations (2.35)-(2.37) can be performed very efficiently. Nevertheless, for larger problems, the explicit evaluation of the matrix Δ might be too memory-consuming. In such cases, it is possible to compute the multipliers (2.32)-(2.34) individually.

Let $\mathbf{w}^{(0)} \in \mathbb{Q}_+^n$ be an initial design and let $\mathbf{w}^{(t+1)} = \mathbf{T}^B(\mathbf{w}^{(t)})$ for $t = 0, 1, 2, ...$ It is evident that the designs $\mathbf{w}^{(t)}$ have all components positive and the matrices $\mathbf{M}(\mathbf{w}^{(t)})$ are non-singular. Recall that the general theory in [31] (described in Subsection 1.3) implies that the sequence $\{\mathbf{M}(\mathbf{w}^{(t)})\}_{t=0}^\infty$ converges to some non-singular matrix $\mathbf{M}^\infty$, but it does not guarantee that $\mathbf{M}^\infty$ is the optimal information matrix. However, it is possible to show that under a mild technical condition the designs $\mathbf{w}^{(t)}$ do converge to the optimum in the sense that their criterial values $\phi_D(\mathbf{w}^{(t)})$ converge to the optimal value of (2.16):

Theorem 4.2 *Let* $\mathbf{w}^{(0)} \in \mathbb{Q}_+^n$ *and let* $\mathbf{w}^{(t+1)} = \mathbf{T}^B(\mathbf{w}^{(t)})$ *for* $t = 0, 1, 2, ...$ *Let* $\liminf_{t\to\infty} s^\delta(\mathbf{w}^{(t)}) > 0$. *Then,* $\lim_{t\to\infty} \phi_D(\mathbf{w}^{(t)}) = \phi_D(\mathbf{w}^*)$, *where* $\mathbf{w}^*$ *is any solution of* (2.16).

The technical condition $\liminf_{t\to\infty} s^\delta(\mathbf{w}^{(t)}) > 0$ is automatically satisfied once $\mathcal{X}_0 = \emptyset$. The case $\mathcal{X}_0 \neq \emptyset$ takes place only if c_x is exactly equal to one for some $x \in \mathcal{X}$, which is likely to occur only seldom in real applications. Moreover, even if this is the case, i.e., if $\mathcal{X}_0 \neq \emptyset$, it is possible to adopt a conservative practical approach by slightly increasing the costs c_{x_0}, $x_0 \in \mathcal{X}_0$. Alternatively, one can use the following lemma.

Lemma 4.3 *Let* $\mathbf{w}^{(0)} \in \mathbb{Q}_+^n$ *and let* $\mathbf{w}^{(t+1)} = \mathbf{T}^B(\mathbf{w}^{(t)})$ *for* $t = 0, 1, 2, \ldots$. *Let* $v_0 = \max\{\phi_D(\mathbf{w}) : \mathbf{w} \geq \mathbf{0}_n, \sum_{x_0 \in \mathcal{X}_0} w_{x_0} = 1\}$, *that is,* v_0 *is the optimal value of the standard problem of approximate D-optimality on* $\mathcal{X}_0$. *Assume that* $\phi_D(\mathbf{w}^{(s)}) > v_0$ *for some* $s \in \{0, 1, 2, \ldots\}$. *Then,* $\liminf_{t\to\infty} s^\delta(\mathbf{w}^{(t)}) > 0$.

In most cases, the value v_0 from Lemma 4.3 is so small, that $\phi_D(\mathbf{w}^{(s)}) > v_0$ is satisfied already for the initial design $\mathbf{w}^{(0)}$. In such cases, the convergence of the barycentric algorithm is guaranteed from the outset.

The previous results establish the monotonic convergence of the "pure" barycentric algorithm, i.e., without the application of the deletion rules from Theorem 3.2. If we do remove the redundant design points during the computation, the actual reduction of the size of the design space can take place only a finite number of times (since $\mathcal{X}$ is finite). Then, the convergence results can be applied to the design space of the final size.

5 Numerical study

Assume the full quadratic linear regression model with homoscedastic uncorrelated observations on a 101×101 equidistant rectangular grid in the square $[0, 1] \times [0, 1]$. For this model, the observations y satisfy

$$E(y) = \beta_1 + \beta_2 r_1(x) + \beta_3 r_2(x) + \beta_4 r_1^2(x) + \beta_5 r_2^2(x) + \beta_6 r_1(x) r_2(x), \tag{2.38}$$

where $r_1(x)$ and $r_2(x)$ transform the index $x \in \mathcal{X} = \{1, 2, ..., 101^2\}$ into two coordinates in $[0, 1]$ by formulas $r_1(x) = \lfloor (x-1)/101 \rfloor / 100$, and $r_2(x) = ((x-1) \bmod 101)/100$. That is, the model has $m = 6$ unknown parameters $\beta_1, \ldots, \beta_6$ and the regressors are given by

$$\mathbf{f}(x) = (1, r_1(x), r_2(x), r_1^2(x), r_2^2(x), r_1(x) r_2(x))^\top.$$

The costs were chosen to be $c_x = 0.1 + 6 r_1(x) + r_2(x)$ for all $x \in \mathcal{X}$. Thus, the sizes of the partitions $\mathcal{X}_+$, $\mathcal{X}_-$, and $\mathcal{X}_0$ are $n_+ = 9465$, $n_- = 720$, and $n_0 = 16$, respectively. Every 16 iterations, we used Theorem 3.2, parts (i)-(iii), to remove redundant design points.

Figures 2.3a, 2.3b, and 2.3c illustrate the designs and the areas of removed design points at the moments when the algorithm reached the lower bounds of efficiencies 0.99, 0.999 and 0.9999, as given by Theorem 3.2. Figure 2.3d shows the time-dependence of

the iteration number and the number of non-removed design points. Note that as the size of the design space shrinks, the speed of the computation (measured by the number of iterations) increases. The resulting D-optimal size- and cost-constrained design resembles the standard approximate D-optimal design, with some support points shifted towards the low-cost regions and with attenuated weights corresponding to more expensive trials.

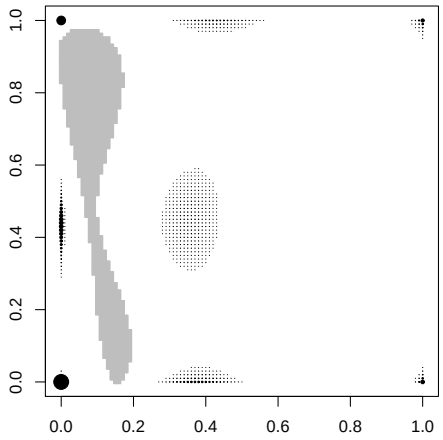
To obtain more general numerical results, we generated random instances of the problem (2.16) with the aim to give statistical information about the speed of computation of the S&C algorithm. Clearly, the execution time can be strongly influenced by the software and the hardware used (we used the Matlab computing environment on 64 bit Windows 7 system running an Intel Core i3-4000M CPU processor at $2,40$ GHz with 4 GB of RAM). Therefore, we also exhibit results about the numbers of iterations, which depend only on the computational method itself.

More specifically, we run the S&C algorithm 1000 times for various combinations of parameters $p_0 = n_0/n$, $p_{+-} = n_+/n_-$ and l, where l is the number of iterations between successive applications of the deletion method based on Theorem 3.2, parts (i)-(iii). In each simulation, we varied one of the parameters $p_0 \in \{0, 0.25, 0.5, 0.75, 1\}$, $p_{+-} \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, or $l \in \{1, 4, 16, 64, \infty\}$, keeping all other parameters fixed (the value $l = \infty$ means that the removal of redundant design points was not performed at all). The size of the design space and the number of model parameters were always the same: $n = 600$ and $m = 4$.

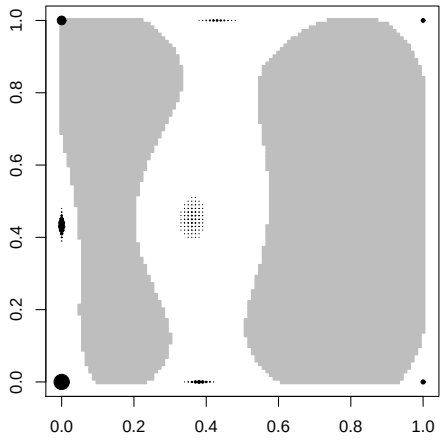
For each couple p_0, p_{+-}, we generated $n_+ = \lfloor (1 - p_0)p_{+-}n \rfloor$ costs independently from the shifted exponential distribution $\mathrm{Exp}(1) + 1$, and $n_- = \lfloor (1 - p_0)(1 - p_{+-})n \rfloor$ costs independently from the uniform distribution on $(0, 1)$. Remaining $n - n_+ - n_- \approx np_0$ costs were set to 1. Regressors $\mathbf{f}(x) \in \mathbb{R}^m, x \in \mathcal{X}$, were sampled independently from $\mathrm{N}_m(\mathbf{0}_m, \mathbf{I}_m)$.

The S&C algorithm started its iterative computation from the initial design $\mathbf{w}^{(0)}$ defined by (2.17)-(2.19). In every step, the current design was updated according to (2.31)-(2.34).

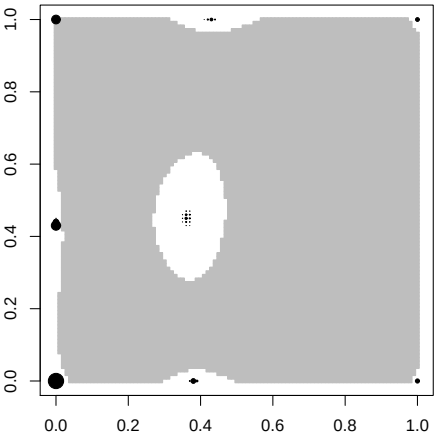
After each successful application of the deletion method, we had to "re-normalize"



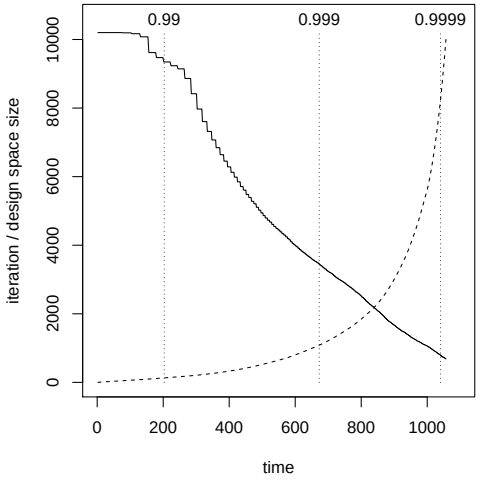
(a) $\mathrm{eff}_D \geq 0.99$



(b) $\mathrm{eff}_D \geq 0.999$



(c) $\mathrm{eff}_D \geq 0.9999$



(d)

Figure 2.3: Figures 2.3a, 2.3b, and 2.3c visualize the designs constructed using the barycentric algorithm for the quadratic regression model (2.38). The weights are denoted by black dots with areas proportional to their numeric values. The gray regions denote the points of the original design space that were removed by the deletion rules from Theorem 3.2. Figure 2.3d shows the iteration number (dashed line) and the number of residual design points (solid line) as they depend on time in seconds. The vertical lines (dotted) denote the time moments when the efficiencies 0.99, 0.999 and 0.9999 were achieved, which corresponds to Figures 2.3a, 2.3b, and 2.3c, respectively.

the design to satisfy the constraints of (2.16). A natural re-normalization is to set $w_{x_+} = h_+ w_{x_+}$ for all $x_+ \in \mathcal{X}_+$, $w_{x_-} = h_- w_{x_-}$ for all $x_- \in \mathcal{X}_-$, and $w_{x_0} = h_0 w_{x_0}$ for all $x_0 \in \mathcal{X}_0$, where h_+, h_- and h_0 are suitably chosen positive constants.

Let $\mathbf{w} \geq \mathbf{0}_n$ be a fixed design with $0 < s = \sum_x w_x \leq 1$ and $0 < \sum_x c_x w_x \leq 1$. Let $s_+ = \sum_{x_+} w_{x_+}$, $s_- = \sum_{x_-} w_{x_-}$, and $s_0 = \sum_{x_0} w_{x_0}$. Let $s_+^\delta = \sum_{x_+} \delta_{x_+} w_{x_+}$, and let $s_-^\delta = \sum_{x_-} \delta_{x_-} w_{x_-}$.

Assume that $s_+, s_- > 0$, which implies $s_+^\delta, s_-^\delta > 0$. For the requirement that the re-normalized design should satisfy both (2.11) and (2.12), the following linear equalities must hold

$$h_+ s_+ + h_- s_- + h_0 s_0 = 1, \tag{2.39}$$

$$h_+(s_+ + s_+^\delta) + h_-(s_- - s_-^\delta) + h_0 s_0 = 1. \tag{2.40}$$

If $s_0 = 0$, then (2.39) and (2.40) give $h_+ = s_-^\delta/(s_+ s_-^\delta + s_- s_+^\delta)$, $h_- = s_+^\delta/(s_+ s_-^\delta + s_- s_+^\delta)$ and h_0 can be arbitrary. If $s_0 > 0$, equalities (2.39) and (2.40) do not uniquely determine any of the re-normalization factors h_+, h_-, h_0. Therefore, motivated by keeping the ratio of the weights of $\mathcal{X}_+ \cup \mathcal{X}_-$ and $\mathcal{X}_0$ the same before and after the re-normalization, we can demand equality

$$\frac{s_+ + s_-}{s_0} = \frac{h_+ s_+ + h_- s_-}{h_0 s_0}, \tag{2.41}$$

which is linear in h_+, h_-, h_0. The solution of the linear system (2.39)-(2.41) is

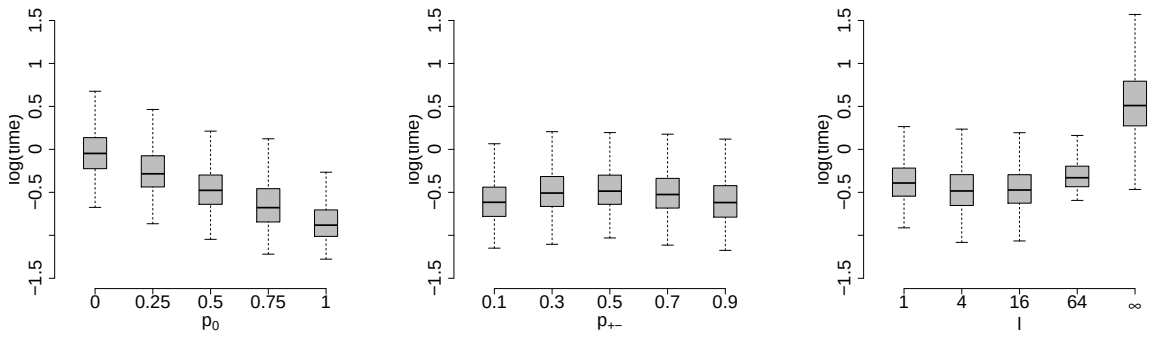
$$h_+ = \frac{s_-^\delta(s_+ + s_-)}{s(s_+ s_-^\delta + s_- s_+^\delta)},\ h_- = \frac{s_+^\delta(s_+ + s_-)}{s(s_+ s_-^\delta + s_- s_+^\delta)},\ h_0 = \frac{1}{s}.$$

If $s_+ = s_- = s_+^\delta = s_-^\delta = 0$, we have $s_0 > 0$, which means that we can simply set $h_0 = 1/s_0$, and h_+, h_- can be arbitrary. In this case, the S&C algorithm is reduced to the standard multiplicative algorithm without the cost constraint.

The required minimal efficiency was set to 0.99999, which means that we stopped the algorithm once this lower bound was reached by the actual design (cf. Theorem 3.2). We remark that the algorithm converged in all 15000 simulated problems.

The results in the form of box-plots are exhibited in Figures 2.4 and 2.5. The results indicate that the problem is computationally more demanding for greater values of $\tilde{n} = n_0 + n_+ n_-$. Thus, with fixed $n = n_0 + n_+ + n_-$, we can generally expect a longer

computation time (and, to a lesser extent, a greater number of iterations) for $n_0 = 0$ and $n_+ \approx n_-$. The numerical results also demonstrate that removal of redundant design points can decrease the computation time by an order of magnitude.



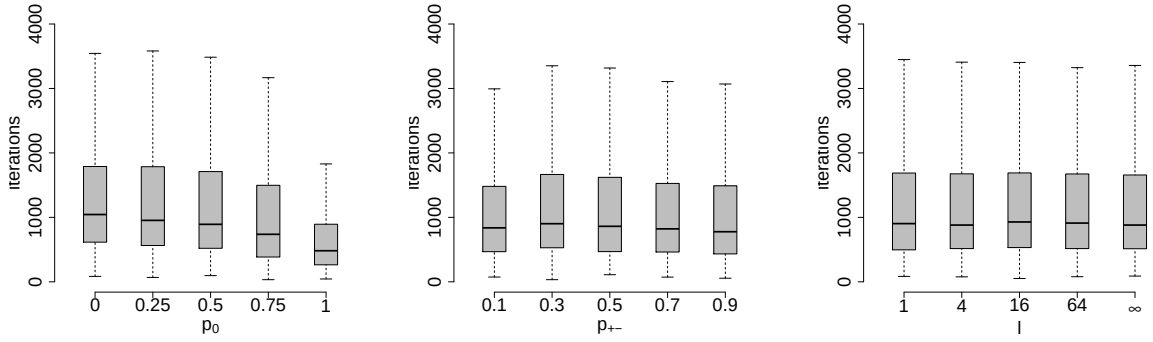
(a) (b) (c)

Figure 2.4: The decimal logarithm of the computation time (in seconds) of the S&C algorithm necessary to achieve the efficiency of 0.99999. Each boxplot is based on 1000 randomly generated problems of the type (2.16) with $n = 600$ and $m = 4$. Figure 2.4a: $p_{+-} = 0.5$, $l = 16$, and p_0 varies in $\{0, 0.25, 0.5, 0.75, 1\}$. Figure 2.4b: $p_0 = 0.5$, $l = 16$, and p_{+-} varies in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. Figure 2.4c: $p_0 = 0.5$, $p_{+-} = 0.5$, l varies in $\{1, 4, 16, 64, \infty\}$. The value $l = \infty$ means that no removals of redundant design points were performed. See the main text for more details.

The Matlab code that implements the S&C algorithm with the deletion method is available at: `www.iam.fmph.uniba.sk/design/` .

Finally, we compared the S&C algorithm (without and also with the application of the deletion rules) to two natural competitors: the vertex-direction (VD) algorithm, and the semidefinite programming (SDP) algorithm implemented by the SeDuMi solver for Matlab. Each algorithm was tested on 100 problems randomly generated with $n = 600$, $m = 4$, $p_0 = 0.5$ and $p_{+-} = 0.5$. The S&C algorithm and the VD algorithm were terminated once the efficiency bound from Theorem 3.2 guaranteed the efficiency of at least 0.99999. The stopping rule of the SDP algorithm was defined by the standard settings except for the parameter `sedumi.eps` which was set to 10^{-5}.

As shown in Figure 2.6, the S&C algorithm with removal of redundant design points is approximately twice as fast as the SDP algorithm, one order of magnitude faster than



(a) (b) (c)

Figure 2.5: The number of iterations of the S&C algorithm necessary to achieve the efficiency of 0.99999. Each boxplot is based on 1000 randomly generated problems of the type (2.16) with $n = 600$ and $m = 4$. Figure 2.5a: $p_{+-} = 0.5$, $l = 16$, and p_0 varies in $\{0, 0.25, 0.5, 0.75, 1\}$. Figure 2.5b: $p_0 = 0.5$, $l = 16$, and p_{+-} varies in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. Figure 2.5c: $p_0 = 0.5$, $p_{+-} = 0.5$, l varies in $\{1, 4, 16, 64, \infty\}$. The value $l = \infty$ means that no removals of redundant design points were performed. See the main text for more details.

the pure S&C algorithm, and about three orders of magnitude faster than the VD algorithm. The S&C algorithms produced the most efficient designs and the VD algorithm produced the least efficient designs. Note that the advantage of the SDP method is that the computation time is very stable. However, using the SDP solver, we were able to solve the problems (2.13) and (2.16) only for dimensions smaller than $n = 4000$. On the other hand, the barycentric and the VD algorithms can be also used for much larger problems.

6 Miscellaneaous remarks

6.1 A pair of general positive linear constraints

The theoretical results of this chapter and the S&C algorithm can be used for a general pair of positive linear constraints, with many practical interpretations besides the limit on the size and on the cost of the experiment. Indeed, consider the D-optimal design

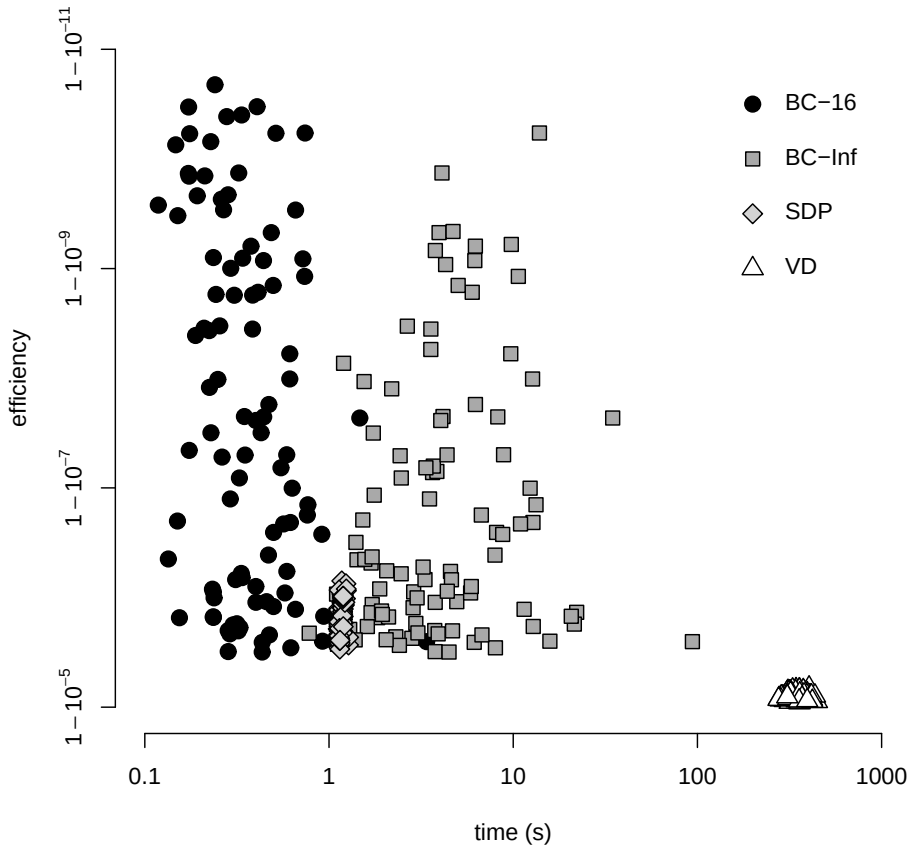


Figure 2.6: A comparison of selected algorithms for computing approximate D-optimal size- and cost-constrained designs. Each symbol represents the computation time (in seconds) and the achieved efficiency relative to the optimum for one of the competing algorithms applied to a randomly generated problem. Black circles, dark gray squares, light gray diamonds and white triangles represent the results of the S&C algorithm with the application of the deletion rules ($l = 16$), the S&C algorithm without the application of the deletion rules ($l = \infty$), the SDP algorithm, and the VD algorithm, in the respective order. Each algorithm was tested on 100 randomly generated problems. See the text for details.

problem

$$\tilde{\mathbf{w}}^* \in \arg\max_{\tilde{\mathbf{w}}} \{\phi_D(\tilde{\mathbf{w}}) : \tilde{\mathbf{w}} \geq \mathbf{0}_n, \sum_x c_x^{(1)} \tilde{w}_x \leq 1, \sum_x c_x^{(2)} \tilde{w}_x \leq 1\}, \tag{2.42}$$

where $c_x^{(1)}, c_x^{(2)} > 0$, for all $x \in \mathcal{X}$, are given constants. Let $\tilde{\mathbf{f}}(x)$, $x \in \mathcal{X}$, denote the regressors. It turns out that the problem (2.42) can be transformed to (2.13) in an

analogous way as the single-cost constrained problem (2.15) can be transformed to the standard problem (2.14). Specifically, it is possible to use transformations $w_x = c_x^{(1)}\tilde{w}_x$, $c_x = c_x^{(2)}/c_x^{(1)}$, and $\mathbf{f}(x) = \tilde{\mathbf{f}}(x)/\sqrt{c_x^{(1)}}$ for all $x \in \mathcal{X}$. Similarly, the problem (2.42) with the inequality constraints replaced with equality constraints can be transformed to (2.16).

6.2 Cost minimization with a prescribed information matrix

A related problem is to find the minimum-cost design in the class of all designs with a prescribed information matrix $\mathbf{M}^*$ (note that the class of such designs can be infinite), see, e.g., [63]. For the finite design space $\mathcal{X}$, this leads to the linear programming problem

$$\mathbf{w}^* \in \arg\min_{\mathbf{w}} \{\sum_x c_x w_x : \mathbf{w} \geq \mathbf{0}_n, \sum_{x\in\mathcal{X}} w_x \mathbf{f}(x)\mathbf{f}^\top(x) = \mathbf{M}^*\}.$$

Hence, once we identify an approximate D-optimal size- and cost-constrained design, it is reasonable to solve the above-mentioned linear programming problem, which might provide an alternative optimal design with a strictly lower cost.

6.3 Relations to stratified D-optimality

Let $c_{x_+} \equiv c_+ > 1$ for all $x_+ \in \mathcal{X}_+$, $c_{x_-} \equiv c_- \in (0,1)$ for all $x_- \in \mathcal{X}_-$ and let $\mathcal{X}_0 = \emptyset$. Define $s_+(\mathbf{w}) = \sum_{x_+} w_{x_+}$ and $s_-(\mathbf{w}) = \sum_{x_-} w_{x_-}$. If $\mathbf{w}$ is permissible for (2.16), then $s_+(\mathbf{w}) + s_-(\mathbf{w}) = 1$ and $c_+ s_+(\mathbf{w}) + c_- s_-(\mathbf{w}) = 1$. Hence the values

$$s_+ = s_+(\mathbf{w}) = \frac{\delta_-}{\delta_+ + \delta_-},\ s_- = s_-(\mathbf{w}) = \frac{\delta_+}{\delta_+ + \delta_-}$$

do not depend on $\mathbf{w}$. Therefore, the set $\mathbb{Q}^n$ of designs satisfying (2.11)-(2.12) is the same as the set of stratified designs with partitions $\mathcal{X}_+$, $\mathcal{X}_-$ and weights s_+, s_-; see [31]. In other words, in this situation the size- and cost-constrained D-optimality coincides with the stratified D-optimality. We show that for this particular case, the results described in this chapter are the same as the results from [31], including the equivalence theorem, the "deletion rules" and the S&C algorithm.

We now provide the equivalence theorem for stratified D-optimality, as stated in [31], simplified for two partitions, and show its coincidence with Theorem (3.1).

Theorem 6.1 *Let $\mathbf{w} \in \mathbb{Q}_r^n$. Then the following statements are equivalent.*
(i) The design $\mathbf{w}$ is D-optimal in $\mathbb{Q}^n$.

(ii) There exists $g_+, g_- \in \mathbb{R}$ *such that* $s_+ g_+ + s_- g_- = 0$, $d_{x_+}(\mathbf{w}) \leq m + g_+$ *for all* $x_+ \in \mathcal{X}_+$ *and* $d_{x_-}(\mathbf{w}) \leq m + g_-$ *for all* $x_- \in \mathcal{X}_-$.
(iii) $s_+ \max_{x_+ \in \mathcal{X}_+} d_{x_+}(\mathbf{w}) + s_- \max_{x_- \in \mathcal{X}_-} d_{x_-}(\mathbf{w}) \leq m$.

Equivalence between (ii) in Theorem (6.1) and (ii) in Theorem (3.1) can be easily verified by setting $h = g_+/\delta_+ = -g_-/\delta_-$ and using the fact that $s_+ = \delta_-/(\delta_+ + \delta_-)$ and $s_- = \delta_+/(\delta_+ + \delta_-)$. Similarly, equivalence between (iii) in Theorem (6.1) and (iii) in Theorem (3.1) can be verified.

Extreme points of the set of permissible designs can be written in the form

$$\mathbf{q}_{x_+x_-} = \frac{\delta_-}{\delta_+ + \delta_-}\mathbf{e}_{x_+} + \frac{\delta_+}{\delta_+ + \delta_-}\mathbf{e}_{x_-} = s_+\mathbf{e}_{x_+} + s_-\mathbf{e}_{x_-};\ x_+ \in \mathcal{X}_+,\ x_- \in \mathcal{X}_-.$$

Both efficiency bound and deleting rule from Theorem 3.2 are based on ϵ:

$$\begin{aligned}
\epsilon &= \max_{x_+ \in \mathcal{X}_+, x_- \in X_-} \tilde{d}_{x_+x_-}(\mathbf{w}) - m = \max_{x_+ \in \mathcal{X}_+, x_- \in X_-} \frac{\delta_+ d_{x_-}(\mathbf{w}) + \delta_- d_{x_+}(\mathbf{w})}{\delta_+ + \delta_-} - m \\
&= \frac{\delta_-}{\delta_+ + \delta_-} \max_{x_+ \in \mathcal{X}_+} d_{x_+}(\mathbf{w}) + \frac{\delta_+}{\delta_+ + \delta_-} \max_{x_- \in \mathcal{X}_-} d_{x_-}(\mathbf{w}) - m \\
&= s_+ \max_{x_+ \in \mathcal{X}_+} d_{x_+}(\mathbf{w}) + s_- \max_{x_- \in \mathcal{X}_-} d_{x_-}(\mathbf{w}) - m,
\end{aligned}$$

which suggests that efficiency bounds and deleting rules are the same from both points of view.

Let $\mathbf{w} \in \mathbb{Q}^n$. Then using $s^\delta(\mathbf{w}) \equiv \delta_+ s_+ = \delta_- s_- = \delta_+\delta_-/(\delta_+ + \delta_-)$, we can define the barycentric coordinates of $\mathbf{w}$ for every $x_+ \in \mathcal{X}_+, x_- \in \mathcal{X}_-$ by

$$(\tilde{w}(\mathbf{w}))_{x_+x_-} = \frac{(\delta_+ + \delta_-)w_{x_+}w_{x_-}}{s^\delta(\mathbf{w})} = \frac{(\delta_+ + \delta_-)^2 w_{x_+}w_{x_-}}{\delta_+\delta_-} = \frac{w_{x_+}}{\frac{\delta_-}{\delta_+ + \delta_-}} \frac{w_{x_-}}{\frac{\delta_+}{\delta_+ + \delta_-}} = \frac{w_{x_+}}{s_+}\frac{w_{x_-}}{s_-}.$$

Obviously, coincidence of barycentric coordinates $\tilde{w}(\mathbf{w})_{x_+x_-}$ and extreme points $\mathbf{q}_{x_+x_-}$ results in the coincidence of the barycentric updating rules.

6.4 A re-normalization method

For computing D-optimal stratified designs, there exists a fast re-normalization heuristic, and extensive numerical computations suggest that it always converges to the optimum. However, to our best knowledge, there is no analogous re-normalization heuristic for the general size- and cost-constrained problem (2.16). For instance, an obvious suggestion would be using an alternate application of the standard multiplicative algorithm (which

could transform a design from $\mathbb{Q}_+^n$ outside of $\mathbb{Q}_+^n$), and the re-normalization described in Section 5 (which transforms any positive design back to $\mathbb{Q}_+^n$). Numerical experiments suggest that this method fails to produce a convergent sequence of designs.

6.5 The middle role of the S&C algorithm

It is worth noting that in Section 5 of [70], multiplicative algorithms were viewed to have a "middle" role, a sequel to a vertex-direction algorithm to be used when most non-optimal design points had been eliminated and a precursor to more powerful methods when convergence of the multiplicative algorithm becomes weak. However, the deletion of redundant points reduces the problem of the slow final convergence of the S&C algorithm, cf. Fig. 2.6.

Chapter III

Privacy sets

Parts of the results of this chapter (Section 2 and Section 3) were already published in [6] and [50].

In a majority of real-life problems, the number of the measurements is limited by some prescribed number $N \in \mathbb{N}$. For a finite design space $\mathcal{X}$, such an *exact design* can be represented by a vector of weights as in II. However, we drop the assumption of finiteness of $\mathcal{X}$ in this chapter and permit all design spaces that are compact (in $\mathbb{R}^d$). Instead of a vector of weights, we define a design ξ as a finite subset of $\mathcal{X}$:

$$\xi \subseteq \mathcal{X}, |\xi| = N. \tag{3.1}$$

Definition 3.1 implies that no replicated observations in the same design points are allowed, which is a natural restriction for designs considered in this chapter (see the beginning of Section 1.1). Note that the concept of a subset of size N enables us to include also design spaces that are finite, but large-scale, and operating with such enormous vectors of design weights would be computationally infeasible.

1 Preliminaries

Similarly to the previous chapter, we focus on the designs limited by some linear constraints. For exact replication-free designs defined by (3.1), l general linear restrictions can be written as

$$\frac{1}{N} \sum_{x \in \xi} a_{xj} \leq b_j \quad j = 1, \ldots, l, \tag{3.2}$$

where a_{xj} and b_j are constants. Some of the common linear restrictions are listed in Section 1.1 of Chapter II. They are formulated in terms of the experimental design defined as a vector of weights (see (2.1)), but can be easily expressed in accordance with the replication-free definition of a design (3.1). Also note that excluding replications itself yields for a finite design space finite number of linear constraints on the design (as described in Section 1.1 of Chapter II).

Analogously to (2.3), let us now define the standardized information matrix of a design ξ for a given regression function $\mathbf{f} : \mathcal{X} \to \mathbb{R}^m$ and a given number of trials N:

$$\mathbf{M}(\xi) = \frac{1}{N} \sum_{x \in \xi} \mathbf{f}(x)\mathbf{f}(x)^\top. \tag{3.3}$$

The problem of finding an optimal linearly constrained experimental design is to maximize criterion ϕ in the set of all permissible designs Γ, that is, to find

$$\xi^* \in \arg\max_{\xi \in \Gamma} \phi(\xi),$$

where $\Gamma \subseteq \Xi$ and Ξ denotes the set of all designs.

In optimal design of experiments, the target function ϕ is typically a function of the information matrix of ξ, that is, $\phi(\xi) = \phi(\mathbf{M}(\xi))$. For instance, the standardized D-optimality can be for exact designs defined as

$$\phi_D(\xi) = \det^{1/m}(\mathbf{M}(\xi)), \quad \xi \in \Gamma. \tag{3.4}$$

However, this is not typical in case of so-called space-filling designs: Below, we explain how these two areas of experimental design differ and present an idea of how to "bind" them together by the notion of *privacy sets*.

1.1 Space-filling designs

In computer simulation experiments, one usually runs a deterministic code rather than performs the real (stochastic) data generating process. This implies that replicating measurements in the same design point becomes irrelevant and the universal recipe is to spread out the measurements "as uniformly as possible" across the design space, yielding so-called *space-filling* designs. This approach justifies the use of definition 3.1 for designs considered in this chapter.

1.1.1 "Soft" space-filling methods

The classic approach of the space-filling methods is to focus on the target function ϕ to be optimized, so that the corresponding optimal design would have the desired space-filling properties. That is, ϕ does not measure the amount of the information via the information matrix and the corresponding regression model, but says how well the points are distributed in the design space, see [59].

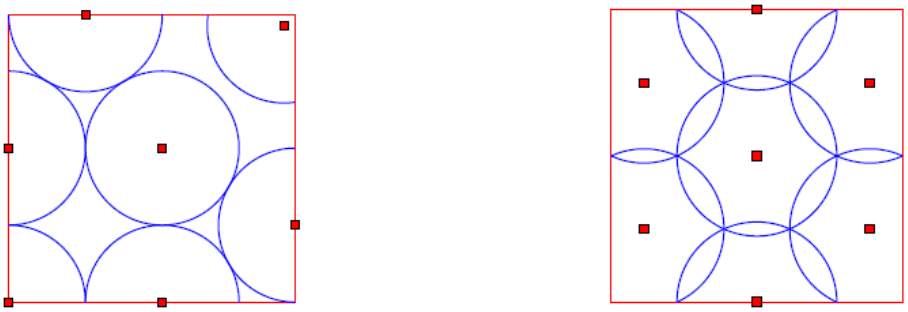
Among these criteria, two are the most fundamental (see [38]): *Maximin* criterion

$$\phi_{Mm}(\xi) = \min_{x,y\in\xi, x\neq y} \rho(x,y),$$

which, when maximized, makes the smallest distance between neighbouring points in ξ as large as possible. Alternatively, minimizing *minimax* criterion

$$\phi_{mM}(\xi) = \max_{y\in\mathcal{X}} \min_{x\in\xi} \rho(x,y)$$

makes the maximum distance from all the points in $\mathcal{X}$ to their closest design point in ξ as small as possible. Examples of these designs for $N = 7$ points are illustrated in Fig.3.1.



(a) Maximin optimal design (b) Minimax optimal design

Figure 3.1: Maximin and minimax distance designs for $N = 7$ points in $[0,1]^2$. The circles have radius $\phi_{Mm}(\xi)/2$ in 3.1a and radius $\phi_{mM}(\xi)$ in 3.1b. Source: [59].

It is usual in computer experiments that only a few factors turn out to be "active". That is, it may be desirable for the design to be space-filling in lower dimensional projections as well. We call such designs "non-collapsing". In [21] the *average reciprocal distance* (ARD) criterion was proposed, modified such that the optimal design has good projection properties onto a given set of subspaces of the design space. Let $J \subset \{1,2,\ldots,d\}$ be a nonempty index set of dimensions of subspaces we would like to consider and let $\mathcal{X}_j$ denote the set of all $\binom{d}{j}$ standard coordinate subspaces of dimension j for every $j \in J$.

The idea of ARD criterion is that the average reciprocal pairwise distance between design points should be minimized. Hence, we use the following formulation of ARD:

$$\phi_{ARD}(\xi) = \left\{ \frac{1}{\binom{N}{2} \sum_{j \in J} \binom{d}{j}} \sum_{j \in J} \sum_{\mathcal{Y} \in \mathcal{X}_j} \sum_{\substack{x,y \in \xi \\ x \neq y}} \left(\frac{j^{1/z}}{\rho_z(x^*_{\mathcal{Y}}, y^*_{\mathcal{Y}})} \right)^{\lambda} \right\}^{1/\lambda}, \tag{3.5}$$

where $z \geq 1$ and $\lambda \geq 1$ are given constants, $x^*_{\mathcal{Y}}$ is the projection of $x \in \mathcal{X}$ onto subspace $\mathcal{Y}$, and ρ_z is for any couple $x = (x_1, \ldots, x_d)^\top$, $y = (y_1, \ldots, y_d)^\top$ defined by $\rho_z(x, y) = \left(\sum_{i=1}^{d} |x_i - y_i|^z \right)^{1/z}$.

Since the active factors are not known in advance, it may be convenient to ensure good space-filling properties in all subspaces of the factors. This issue is dealt with in [40], where the following *maximum projection* (MaxPro) criterion is minimized:

$$\phi_{MaxPro}(\xi) = \left\{ \sum_{\substack{x,y \in \xi \\ x \neq y}} \frac{1}{\Pi_{i=1}^{d} \rho^2(x^*_i, y^*_i)} \right\}^{1/d},$$

where x^*_i denotes projection of $x \in \mathcal{X}$ onto the i-th subspace.

These geometric criteria do not necessarily have to be model-free: They are sometimes combined with an estimation or prediction oriented criterion (like suggested in [49]). We will call all these methods focused on proposing criterion ϕ "soft" space-filling methods.

1.1.2 "Hard" space-filling methods

Non-collapsingness of designs does not have to be ensured only by the appropriate optimization criterion. The search for designs can be, for example, restricted to the class of *Latin hypercube designs (LHDs)* introduced in [45]: Let $\mathcal{X} = [0, 1]^d$ be the design space. Any Latin hypercube design ξ distributes design points within the discrete regular grid of $n = N^d$ points, with levels $0, 1/(N-1), \ldots, 1$ for every dimension, such that for all $x, y \in \xi$ it holds that

$$x_i \neq y_i \text{ for all } i = 1, \ldots, d.$$

Clearly, these designs have the property that any of their one-dimensional projections is spread out optimally with respect to the Maximin criterion. Paper [49] suggests to search

for the optimal Maximin design within the class of Latin hypercube designs, minimizing the criterion

$$\phi_{MmLHD}(\xi) = \left\{ \sum_{\substack{x,y \in \xi \\ x \neq y}} \frac{1}{\rho^k(x,y)} \right\}^{1/k},$$

where $k > 0$ is chosen large enough to achieve maximin distance.

LHDs are widely used in many forms and variations. They represent some restrictions posed on the design, ensuring good projection properties. It may be for example reasonable to search for a D-optimal design among LHDs. Moreover, new restrictions may extend the idea of LHDs. For instance, *Bridge designs (BDs)* introduced in [39] aim to "bridge" the gap between D-optimal designs restricted only by the experimental size and D-optimal LHDs: Let $\mathcal{X} = [0,1]^d$. Any two points $x, y, x \neq y$, of a Bridge design ξ must satisfy

$$|x_i - y_i| > \delta \text{ for all } i = 1, \ldots, d,$$

where δ is a given positive constant. These designs have the property that projections onto any coordinate axis will space the points no closer than some specified distance δ, which may be viewed as a tuning parameter.

Another natural requirement could be to make sure that design points are sufficiently far away from each other. This would for all $x, y \in \xi, x \neq y$ mean that

$$\rho(x,y) > \delta,$$

where δ is a given positive constant. These designs will be called *Minimum-distance designs (MDDs)*. In contrast to LHDs and BDs, Minimum-distance designs space design points in $\mathcal{X}$ directly rather than focusing on the one-dimensional design projections. We deal with these restrictions in Section (4).

This approach (LHDs, BDs, MDDs) is different from the "soft" methods described above; the space-filling properties are not a result of a suitable geometric criterion, but are enforced strictly by the design constraints that must hold. These "hard" space-filling methods then allow for a secondary criterion to be used for optimization (e.g., D- or A-optimality). In Section 2, we introduce the central notion of *privacy sets* for dealing with the "hard" methods. This notion provides a unifying framework for LHDs, BDs and

MDDs, among many others. In fact, we think that our concept of privacy sets represents a much more fundamental "bridge" between the two important ways of designing an experiment (in comparison to Bridge designs dealing with one specific type of constraints only). This can be schematically depicted as follows:

$$\overbrace{\underbrace{\text{OPTIMAL DESIGNS}}_{\text{(do not distribute points across } \mathcal{X})} \qquad \underbrace{\text{SPACE-FILLING DESIGNS}}_{\text{(do not consider regression model)}}}^{\text{PRIVACY SETS}}$$

2 Privacy sets

Let us start with the definition of the central notion for our approach to the generation of (constrained) space-filling designs.

Definition 2.1 *For each $x \in \mathcal{X}$, let $\mathcal{P}(x) \subseteq \mathcal{X}$, $x \in \mathcal{P}(x)$, be a given privacy set of the point x. For any design $\xi \in \Xi$, let $\mathcal{P}(\xi) = \cup_{x \in \xi} \mathcal{P}(x)$ be the privacy set of the design ξ.*

We assume that there is a given upper limit on the size of the experiment $N \in \mathbb{N}$. A design ξ will then be called permissible, if

$$\begin{aligned} |\xi| &\leq N, \\ x &\notin \mathcal{P}(y) \text{ for all } x, y \in \xi, x \neq y. \end{aligned}$$

A permissible design ξ will be called maximal permissible if it cannot be augmented without violation of some of the constraints, i.e., if $\xi \cup \{x\}$ is not permissible for all $x \in \mathcal{X} \setminus \{\xi\}$.

Assumption 2.1 *We assume that $|\xi| = N$ for any maximal permissible design.*

Note that Assumption 2.1 guarantees that any optimal design is of size N, which is in accordance with definition (3.1). We just allow designs of sizes smaller than N to be called permissible, too, in order to explain our algorithm and our results more clearly.

We would like to emphasize that the idea of privacy sets is not artificial, but can be used for example to ensure various space-filling properties. In fact, many of the widely-used designs can be formulated in the terms of privacy sets. We list here some of the most common examples, some of them defined in Section 1.1.2, the others defined as approximate designs on finite $\mathcal{X}$ in Section 1.1 of Chapter II.

1. *Latin hypercube designs* introduced in [45]. The design space $\mathcal{X}$ is a d-dimensional square grid. The size of the design space is $n = N^d$. For any $x \in \mathcal{X}$, the privacy set $\mathcal{P}(x)$ is given by

$$\mathcal{P}(x) = \{y : \exists i \in \{1, \ldots, d\} : x_i = y_i\}.$$

2. *Bridge designs* introduced in [39]. The design space $\mathcal{X}$ is a d-dimensional square grid with $n = L^d$ for some $L \in \mathbb{N}$. For any $x \in \mathcal{X}$, the privacy set $\mathcal{P}(x)$ is given by

$$\mathcal{P}(x) = \{y : \exists i \in \{1, \ldots, d\} : |x_i - y_i| \leq \delta\}, \tag{3.6}$$

   where δ is a given positive constant. Bridge designs represent a generalization of Latin hypercube designs.

3. *Replication-free designs* (cf. for example [11]): For any $x \in \mathcal{X}$, the privacy set $\mathcal{P}(x)$ is given by

$$\mathcal{P}(x) = \{x\},$$

   which means that there is no other constraint except for at most one trial in each design point. For our design definition (3.1), this condition is automatically satisfied.

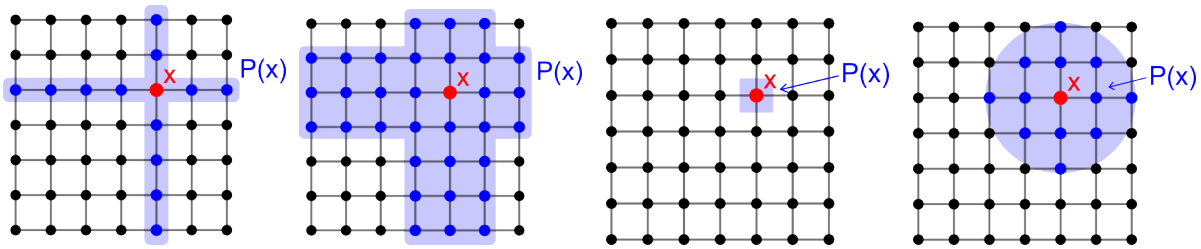
4. *Minimum-distance designs* (discussed in detail in Section 4): For any $x \in \mathcal{X}$, the privacy set $\mathcal{P}(x)$ is given by

$$\mathcal{P}(x) = \{y : \rho(x, y) \leq \delta\}, \tag{3.7}$$

   where δ is a given positive constant and $\rho(.,.)$ is the Euclidean metric. The metric in (3.7) can be replaced by any other metric, which would result in different "shapes" of privacy sets.

5. *Stratified designs* introduced in [31] (as a generalization of *marginally restricted designs*): Let $\mathcal{X}_1, \ldots, \mathcal{X}_k$ be a decomposition of $\mathcal{X}$ into nonempty non-overlapping partitions (strata). Let $\mathcal{X}_j \cap \xi \leq 1$ for $j = 1, \ldots, k$. This setting represents a special case of stratified designs, where at most one trial in every partition is allowed. For $x \in \mathcal{X}_j$, the privacy set $\mathcal{P}(x)$ is then given by

$$\mathcal{P}(x) = \mathcal{X}_j.$$



(a) LHD (b) BD (c) RFD (d) MDD

Figure 3.2: Examples of different privacy sets, namely a Latin hypercube design (LHD), a Bridge design (BD), a replication-free design (RFD) and a Minimum-distance design (MDD) displayed in figures 3.2a, 3.2b, 3.2c, 3.2d, respectively. The blue area denotes the privacy set of the red point x.

Figure 3.2 illustrates four different examples of various privacy sets. For simplicity, the design space is in all four cases a two-dimensional square grid.

Figure 3.3 shows an example of privacy sets of a stratified design with $k = 3$ partitions. Privacy sets of the points in the same partition are identical.

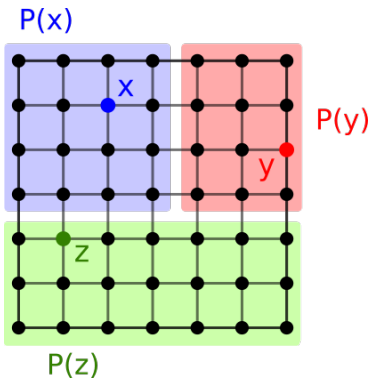


Figure 3.3: An illustration of the privacy sets of a stratified design with 3 partitions. All the points in the "blue" partition have the same privacy set $\mathcal{P}(x)$; the same holds for the points in the "red" partition and the points in the "green" partition.

Note that the use of privacy sets is meaningful not only for computer experiments, but also for the physical ones. They cover, for example, time-separation constraints, where the designs space represents time, and consecutive trials must be performed at least δ time units apart (see, e.g., the second example of Section 5 in paper [65]). In this case, $\mathcal{X} = \mathbb{R}_0^+$ and $\mathcal{P}(x) = (x - \delta, x + \delta) \cap \mathcal{X}$ for all $x \in \mathcal{X}$.

The set $\mathcal{P}(x)$ is typically some neighbourhood of x (containing also x itself) securing point x some "privacy", although this is not strictly required by the definition itself. Using the privacy sets defined above we obtain the optimization problem

$$\xi^* \in \arg\max_{\xi \in \Xi} \{\phi(\xi); |\xi| \leq N \text{ and } x \notin \mathcal{P}(y) \text{ for all } x, y \in \xi, x \neq y\}. \tag{3.8}$$

In above, we automatically assumed restrictions enforced by privacy sets to belong to the class of the linear constraints (3.2). Let us now explicitly explore the relation. Let $\mathcal{P}(x)$ be the privacy set of $x \in \mathcal{X}$. For all $y \in \mathcal{P}(x)$ it must hold that $|\xi \cap \{x, y\}| \leq 1$. Let us now say $x \sim y$ (x and y are "in a relation"), iff $x \in \mathcal{P}(y)$ or $y \in \mathcal{P}(x)$.

Obviously, it is advantageous to reduce the amount of the linear restrictions to the minimum (by preserving their original meaning). This is necessary for example when implementing an algorithm that requires the linear constraints as the input data. For privacy sets, this can be done by forming partitions $\mathcal{X}_j \subset \mathcal{X}$, such that for all $x, y \in \mathcal{X}_j$ it holds that $x \sim y$ and at the same time the size of $\mathcal{X}_j$ is the maximal possible (that is, any augmentation of $\mathcal{X}_j$ would violate the condition $x \sim y$ for all $x, y \in \mathcal{X}_j$).

We now have a set of (in general) overlapping partitions, such that $|\xi \cap \mathcal{X}_j| \leq 1$ and $\cup_j \mathcal{X}_j = \mathcal{X}$. This can be considered a generalization of the idea of a stratified design (see point 5 of the examples listed in above). Note that if, except for the *symmetry*, the relation "$\sim$" possessed also the property of the *transitivity*, the partitions $\mathcal{X}_j$ would not overlap and we would deal exactly with a stratified design as defined in [31].

2.1 Privacy Sets Algorithm

In the following, we present a framework for an exchange-type algorithm - Privacy Sets Algorithm (PSA) - for solving optimization problem (3.8). In general, the specification of the individual steps depends greatly on the design space $\mathcal{X}$ and on the constraints given by the sets $\mathcal{P}(x)$, $x \in \mathcal{X}$, as well as on the optimization criterion ϕ.

A characteristic feature of PSA is the ability to temporarily violate "privacy" of one or more design points. This offers a wider range of possibilities than when performing only permissible changes and prevents the algorithm from getting stuck, for instance when the privacy constraints are very strict. Let $\mathcal{A}(\xi) \subseteq \mathcal{X} \setminus \xi$ denote a set of "candidate

points" that can possibly augment a maximal design ξ. The set $\mathcal{A}(\xi)$, in contrast to $\mathcal{P}(\xi)$, is not an attribute of the problem itself, but can be adjusted in order to ensure the optimum performance of the algorithm. Note that we do not require $\mathcal{A}(\xi)$ to contain solely permissible points $x \notin \mathcal{P}(\xi)$.

One of the key parts of PSA is the efficient augmentation of a design that is not maximal with the remaining runs to achieve the full size N. This is done by employing the following forward-type procedure (Algorithm 1) which adds permissible design points one-by-one until a maximal design is obtained.

Algorithm 1: Greedy Procedure (GrP)

Input : A permissible design ξ, $|\xi| = N^* < N$.

Output: A permissible design ξ, $|\xi| = N$.

1 **for** $i = 1 : N - N^*$ **do**

2 | Augment ξ with the point x, which maximizes $\phi(\xi \cup \{x\})$ subject to $x \notin \mathcal{P}(\xi)$.

3 **end**

4 **return** ξ

One-point permissible augmentation from Step 2 can be crucial in implementing the PSA algorithm effectively. One of the straightforward solutions is to use the exhaustive enumeration of $\mathcal{X} \setminus \mathcal{P}(\xi)$ (for smaller problems) or to use a blind random search, that is, to choose the best point from a set of candidates sampled independently from $\mathcal{X} \setminus \mathcal{P}(\xi)$. This can be performed by a direct rejection method, which, however, tends to be very inefficient for some cases. Therefore, we recommend exploiting particularities of the privacy constraints, if possible (as for example in Sections 3 or 4).

For any maximal design ξ and any point $x \in \mathcal{A}(\xi)$, let $\eta(\xi, x)$ be a possibly random maximal permissible design based on ξ, containing x. We can view $\{\eta(\xi, x) : x \in \mathcal{A}(\xi)\}$ as a randomly generated neighbourhood of the design ξ in the set of all maximal permissible designs, or, in other words, a set of slight "mutations" of the design ξ. The mutation procedure given by Algorithm 2 calculates $\eta(\xi, x)$ for any permissible design ξ and $x \in \mathcal{A}(\xi)$.

Algorithm 2: Mutation Procedure (MuP)

Input : A permissible design ξ, $|\xi| = N$ and a candidate point $x \in \mathcal{A}(\xi)$.

Output: A permissible design ξ, $|\xi| = N$.

Remove from ξ all those points that belong in $\mathcal{P}(x)$.

Let $\xi = \xi \cup \{x\}$.

if $|\xi| = N + 1$ **then**

    Remove the design point from ξ that leads to the smallest drop in the criterion value.

else if $|\xi| < N$ **then**

    Augment the design ξ using GrP to the maximal design.

end

return ξ

The main body of the Privacy Sets Algorithm can then be written in the scheme of Algorithm (3).

PSA does not pose any restrictions on the design space $\mathcal{X}$. Due to implementation reasons, however, we always assume a finite design space of size $n \in \mathbb{N}$. If n is relatively small (up to thousands of design points, say), the implementation of PSA is rather simple for all kinds of privacy sets. This is mainly true because of the ability to store information about availability of each individual point of the design space. However, with n increasing, it becomes computationally intensive or even infeasible to keep n-dimensional vectors in the computer memory and certain specific features of a particular class of privacy sets have to be considered.

3 Privacy sets for Bridge designs

Let δ be a given positive constant. In case of Bridge designs (BDs), the privacy set of a point $x \in \mathcal{X}$ is given by (3.6). This suggests that the focus of BDs is on the space-fillingness of one-dimensional projections of the design points, since no two levels of a given factor can be closer than δ. Motivation for this requirement can be drawn from the non-collapsingness of the design in the case that some of the factors turn out to be

Algorithm 3: Privacy Sets Algorithm (PSA)

```
Construct an initial design ξ using GrP.
repeat
    Set ξ_old ← ξ.
    Construct candidate set A(ξ).
    for x ∈ A(ξ) do
        Set η ← η(ξ, x) using MuP.
        if φ(η) > φ(ξ) then
            Set ξ ← η.
            break the for cycle
        end
    end
until φ(ξ_old) ≥ φ(ξ);
return ξ
```

irrelevant (see Section 1.1).

In this section, we assume $\mathcal{X} = [-1, 1]^d$, that is, every factor takes values in a bounded interval, which can be scaled to $[-1, 1]$. We assume that $\mathcal{X}$ is discretized into a grid of $n = L^d, L \in \mathbb{N}$, equally spaced points with each coordinate from $\{-1+\frac{2k}{L-1}, k = 0, 1, 2, \ldots, L-1\}$. Without loss of generality, we will choose the minimum spacing δ from the set $\left\{\frac{2k}{L-1}, k = 1, 2, \ldots\right\}$.

The requirement of N experimental runs implies $\delta \leq 2/(N-1)$ and $L \geq N$. Note that for the special case $L = N$, we obtain Latin hypercube designs (LHDs). If $L > N$ and $\delta = 2/(L-1)$, the resulting design can be viewed as an "incomplete" LHD.

In general, the most computationally difficult part of PSA is the one-point permissible augmentation in Algorithm 1. Using the specific nature of Bridge designs defined on $[-1, 1]^d$, we implemented this step in two parts.

In the first part, we perform a blind random search by repeated sampling from $\mathcal{X} \setminus \mathcal{P}(\xi)$ and selecting that point which leads to the biggest increase of the criterion value. In the case of Bridge designs, it is enough to store just an $L \times d$ logical matrix representing

permissible levels of factors. Points $x \in \mathcal{X} \setminus \mathcal{P}(\xi)$ can then be easily selected independently, coordinate by coordinate. In the case where additional constraints on the design space are present (see Section 3.2), we used the rejection method.

In the second part, we tune the best point found by using a local search optimization procedure. Its main idea is to sequentially improve the position of the design point added in the first part, always varying only one coordinate at a time. Since all other design points remain unchanged, we are allowed to move only in the permissible area, where no collision with another design point occurs. The process of checking feasibility of the prospective design space point can be handled easily when considering Bridge restrictions (see the reasoning above). Note that this procedure does not require storing all design points (or regressors associated with the design points) of $\mathcal{X}$ in the computer memory. Therefore PSA for Bridge designs can be applied to very large design spaces.

3.1 Examples: D-optimal Bridge designs on a cubical design space

This section provides examples of Bridge designs for specific choices of the optimization criterion and the design space. We compare these to the results from [39], where the same settings were considered. More precisely, we consider D-optimality criterion ϕ_D as defined in (3.4), with the standardized information matrix given by (3.3).

This criterion leads to the optimization problem

$$\xi^* \in \arg\max_{\xi \in \Xi} \{\phi_D(\xi); |\xi| \leq N \text{ and } |x_i - y_i| \geq \delta \text{ for all } i = 1, \ldots, d\}.$$

It is natural to require the design space to be a d-dimensional hypercube $[-1, 1]^d$ (for example when conducting a computer experiment), hence we restrict ourselves to this assumption.

With the number of trials N given, the density of the design space grid needs to be chosen. First, we know that the minimal distance δ cannot exceed $2/(N-1)$, but it is recommended to set it to a smaller value in order to maintain a certain level of "freedom". Now, with the value of δ determined, we can set $L = \lfloor 2k/\delta \rfloor + 1$, where $k \in \mathbb{N}$ and $\lfloor . \rfloor$ represents the lower integer part. In [39], the parameter k is always set to 1, yielding the

"incomplete" LHD. In general, the choice of small L can accelerate the algorithm, but also impair the quality of the resulting design, which suggests that a certain compromise should be made.

In the following examples, we illustrate the performance of our algorithm applied to the problems of various dimensions. We compare the results obtained by our algorithm to the results from [39], based on the relative efficiency of the best designs found in a fixed time interval.

To make the comparison as fair as possible, we implemented both algorithms in Matlab computing environment. The algorithm of [39] was translated from JMP scripting language which was available on the supplement area of the published article. We used 64 bit Windows 7 system running an Intel Core i3-4000M CPU processor at 2.40 GHz with 4 GB of RAM.

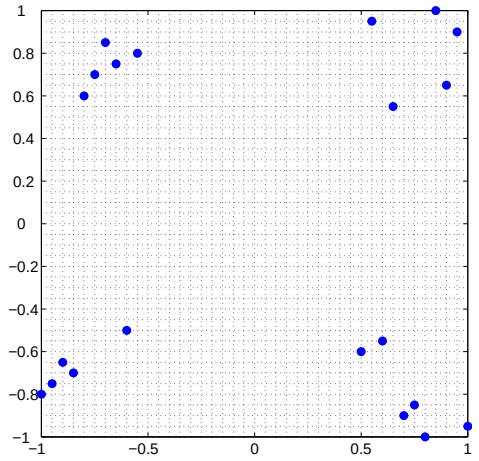
3.1.1 Bridge designs for 2 factors in 21 runs

As the first example, we consider the two-dimensional design space with 21 trials to be allocated. Since this problem instance is rather small, we set the computing time to 60 seconds and run both algorithms several times, until the whole time given is spent. The designs with the highest criterion value found by PSA are displayed in Figure 3.4, with the minimum spacing constant δ set to 0.05 and 0.025 for both the linear regression function $\mathbf{f}(x) = (1, x_1, x_2)^\top$ and the full quadratic regression function $\mathbf{f}(x) = (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2)^\top$.

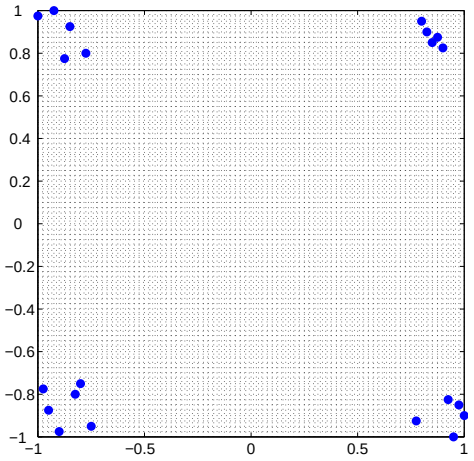
When compared to the results of [39], we observe better results of our algorithm in all four examined situations. The mutual efficiencies of the designs of [39] relative to our designs presented in the figures 3.4a, 3.4b, 3.4c, 3.4d are equal to 0.79, 0.82, 0.96, and 0.96, in the respective order.

This suggests that our algorithm provides better results especially when the spacing constant δ is rather large, which is not surprising. First, the finer design space grid automatically makes the relative difference in the output designs less significant. Second, the greater the value of δ is in comparison to its upper bound $2/(N-1)$, the less space there is for "manoeuvring" for the algorithm. For example, the marginal case $\delta = 2/(N-1)$

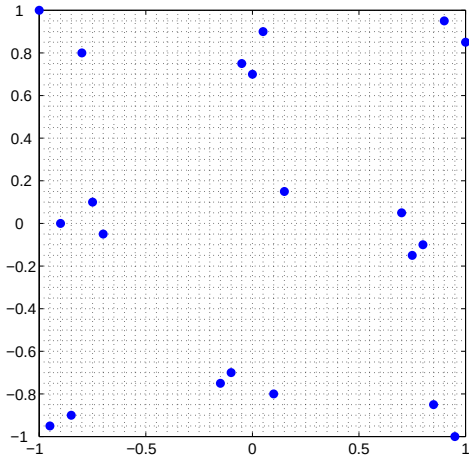
leads to the standard LHD, where no observation can be added without violating some of the privacy sets constraints. These restrictions would fully disable the coordinate-exchanges in the algorithm of [39], but could still be handled by PSA.



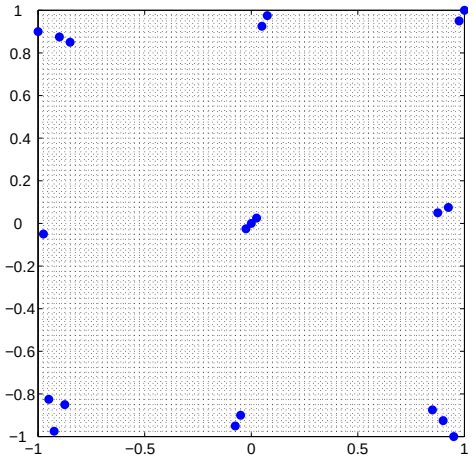
(a) linear regressor, $\delta = 0.05$



(b) linear regressor, $\delta = 0.025$



(c) full quadratic regressor, $\delta = 0.05$



(d) full quadratic regressor, $\delta = 0.025$

Figure 3.4: Bridge designs for $N = 21$ and $d = 2$ found in 60 sec. In 3.4a and 3.4b, the linear regressor was used and the minimal distance δ was set to 0.05 (3.4a) and 0.025 (3.4b). In 3.4c and 3.4d the full quadratic regressor was used and the minimal distance δ was set to 0.05 (3.4c) and 0.025 (3.4d). Efficiencies of the best designs found by the algorithm of [39] relative to the designs presented in the figures 3.4a, 3.4c,3.4b,3.4d are 0.79, 0.82, 0.96 and 0.96, respectively.

3.1.2 A numerical study on Bridge designs

The comparison of the algorithm of [39] and PSA can be extended into more-dimensional cases as well. We performed a small comparative numerical study on a few examples of quadratic regression of various dimensions, similar to the examples provided in Sections 3 and 4 of [39]. For every example, we ran the algorithms for a restricted time, observing the time dependence of the criterion value of the actually best design found by the algorithm. We repeated this procedure several times in order to provide responses from multiple random starts.

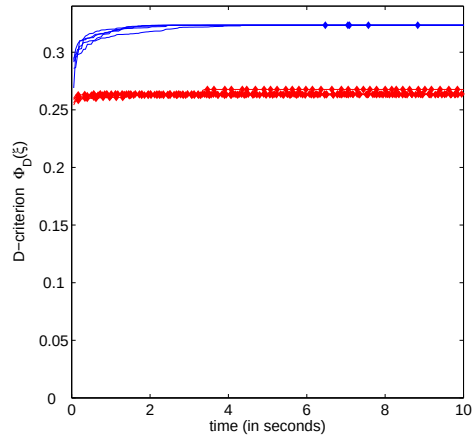
In Figure 3.5, we present results of the two competing algorithms for dimensions $d = 2, 4, 6, 8$ with the numbers of trials $N = 21, 41, 61, 81$, respectively. The minimum spacing δ was in all four cases set to the value $\delta = 1/(N-1)$, which corresponds to the value recommended in [39]. Every t seconds, we plotted the criterion values of the best designs found by the algorithm of [39] (represented by the red lines) and PSA (represented by the blue lines) versus the time displayed on the x-axis. Both algorithms were restarted 5 times, yielding 5 red and 5 blue lines for each problem instance.

The total computational time T, as well as the time period t, were chosen such that they increase with the increasing size of the problem. If an algorithm terminated during the given time T, it was automatically restarted and the resulting value found by its run was stored in the memory. These restarts are denoted by the red and the blue diamonds. We considered the D-optimality criterion given by (3.4) and plotted its values on the y-axis.

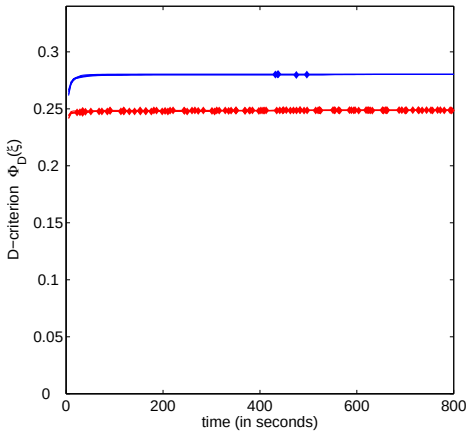
Note that PSA was able to significantly outperform the algorithm of [39] in all four examined cases. We also note that PSA yielded an efficient design in a relatively short time, which suggests that it can be reasonable to stop PSA before reaching an actual local optimum and reduce the execution time.

3.2 Space-filling designs on a constrained design space

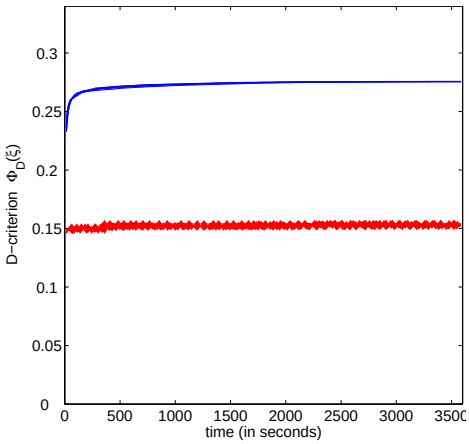
Note that we can practically think of any design enforcing space-fillingness of its one-dimensional projections as a particular instance of a "Bridge design" and that this does not necessarily include D-optimality and a cubical design region. In fact, for any such



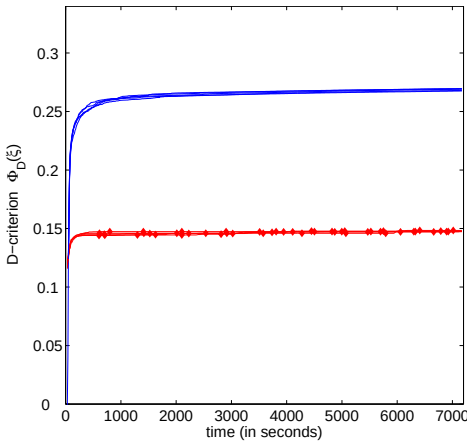
(a) $d = 2$, $N = 21$, $T = 10$, $t = 0.05$



(b) $d = 4$, $N = 41$, $T = 800$, $t = 2$



(c) $d = 6$, $N = 61$, $T = 3600$, $t = 5$



(d) $d = 8$, $N = 81$, $T = 7200$, $t = 10$

Figure 3.5: Comparison of the performance of the algorithm of [39] (red lines) and PSA (blue lines). Y-axis shows the best D-optimality values found by the time displayed on the x-axis (in seconds). The red and the blue diamonds denote restarts of the corresponding algorithms. For every example both algorithms were run 5 times for the time period T.

design, it is enough to satisfy (3.6), that is, restrictions ensuring non-collapsing properties of the design. As an example, in this section we present space-filling on a constrained design space.

For that purpose, we choose one of the space-filling criteria to be optimized, which means that we combine together "soft" and "hard" methods described in Section 1.1. We consider the (modified) average reciprocal distance (ARD) criterion defined by (3.5).

Each design point has to be selected from a design space, which will be some linearly constrained region in this case. However, the PSA algorithm for Bridge designs on cubical regions, described at the beginning of Section 3, can be rather straightforwardly adapted

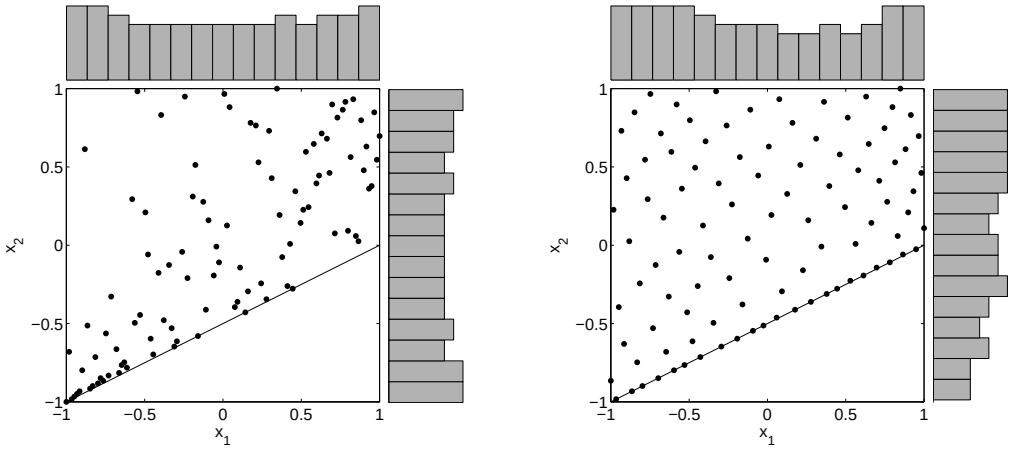
for the constrained design regions.

Without loss of generality, assume that the design space $\mathcal{X}$ is a subset of $[-1,1]^d$ with some additional linear constraints $Ax \leq b$ to be satisfied for every $x \in \mathcal{X}$, where A is an $k \times d$ matrix, $k \in \mathbb{N}$, and $b \in \mathbb{R}^k$. For a given number of trials N, let us have the design space discretized by first making a grid of L^d points on $[-1,1]^d$, and then accepting only those satisfying $Ax \leq b$. The parameter L, determining the density of the grid, has to be chosen large enough, such that not only a permissible design $\xi \in \mathcal{X}$ exists, but also that Assumption 2.1 is satisfied.

The only question is how to implement the one-point permissible augmentation from Algorithm 1. We utilize blind random search on the set $\mathcal{X} \setminus \mathcal{P}(\xi)$ and select the best point found. In the case of a Bridge design on $[-1,1]^d$, we have a simple tool to sample from $\mathcal{X} \setminus \mathcal{P}(\xi)$, by keeping track of permissible and non-permissible levels of individual factors (as described at the beginning of Section 3). If additional linear restrictions are present, $x \in \mathcal{X} \setminus \mathcal{P}(\xi)$ can be generated in the same way and then simply accepted if the condition $Ax \leq b$ holds and rejected otherwise. Clearly, effectiveness of this rejection method depends on the design region defined by the constraints, as well as on the dimension d. In every step, we have to check the restriction on the design space, but we do not have to check the collision with other design points in terms of privacy sets (due to the convenient Bridge constraints), which makes the rejection method more efficient than in the case of general privacy sets.

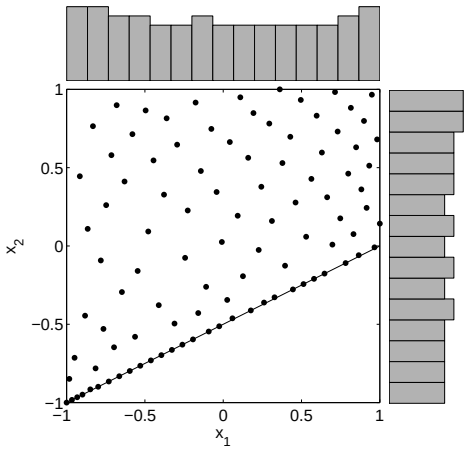
Figure 3.6 displays three resulting designs obtained by PSA for different variants of the ARD criterion. The design space is in all three situations the square $[-1,1]^2$ additionally restricted by $\frac{1}{2}x_1 - x_2 \leq \frac{1}{2}$. We considered designs with $N = 100$ runs and the minimum spacing parameter $\delta = \frac{2}{120-1}$. We note that in this case, it is not possible to set $\delta = \frac{2}{N-1}$ ("LHD-setting"), since Assumption 2.1 would not be fulfilled and the PSA algorithm could not be executed.

The parameters of the ARD criterion (3.5) were set to $z = 1$ and $\lambda = 1$. The nonempty index set $J \subseteq \{1,2\}$ varies in figures 3.6a, 3.6b, 3.6c through all three possibilities, leading to different output designs. Space-filling properties of these designs for various settings of ARD criterion were compared and it shows that the designs constructed with $J = \{1,2\}$ are almost optimal also for criteria with $J = \{1\}$ and $J = \{2\}$.



(a) $J = \{1\}$

(b) $J = \{2\}$



(c) $J = \{1, 2\}$

Figure 3.6: Designs on the two-dimensional constrained design region, obtained by PSA optimizing the ARD criterion for $z = 1$, $\lambda = 1$ and different values of $J \subseteq \{1, 2\}$. PSA allocated 100 trials for the minimum spacing constant $\delta = \frac{2}{120-1}$.

Histograms in Figure 3.6 display space-fillingness of one-dimensional projections of the designs. Ideally, we would like to have these projections as uniformly distributed as possible (corresponding to Latin hypercube constraints), which would ensure non-collapsing properties in one-dimension. One could quite easily think of a heuristics forcing perfectly uniform one-dimensional projections of a design. However, we think it can be more beneficial to "relax" constraints from LHD to BD for some smaller value of δ, if this allows us to improve another optimization criterion, e.g., the criterion assessing space-filling properties of more-dimensional projections (ARD for $J = \{2\}$ or $J = \{1, 2\}$). In this sense, we can compare designs of Figure 3.6 to the design from the paper [55], given in Figure 4 (a) of Section 2.3, which differs only in the scaling of the design space. Although

the design of [55] strictly satisfies Latin hypercube constraints (see the histograms of one-dimensional projections), it completely ignores space-fillingness in other dimensions.

Figure 3.7 presents an example of a design resulting from PSA for ARD criterion in three factors. The three-dimensional design space cube $[-1, 1]^3$ is additionally constrained by $\frac{2}{3}x_1 - x_2 \leq \frac{1}{3}$ and $\frac{3}{4}x_2 - x_3 \leq \frac{1}{4}$. The design consists of $N = 100$ trials with their one-dimensional projections not closer together than $\delta = \frac{2}{140-1}$. The ARD criterion was employed for $z = 1$, $\lambda = 1$ and $J = \{2, 3\}$, which means that we combined the "hard" method forcing one-dimensional space-fillingness and the "soft" method ensuring space-filling properties for dimensions 2 and 3.

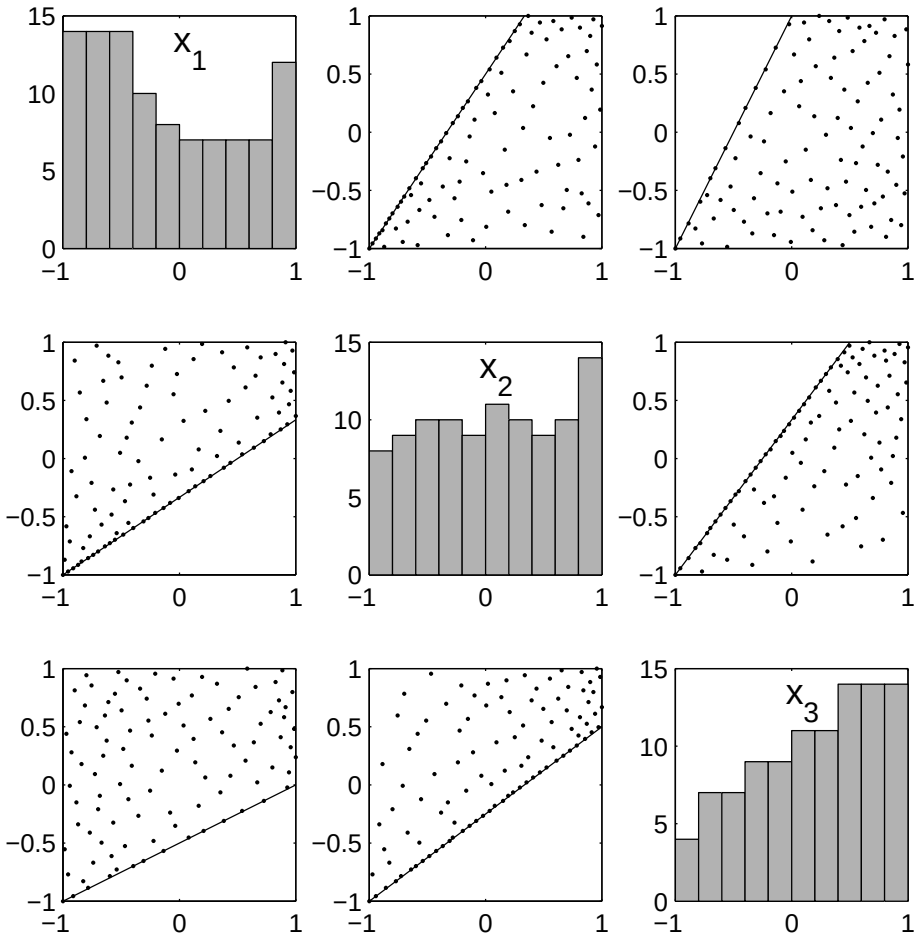


Figure 3.7: Design on the three-dimensional constrained design region, obtained by PSA optimizing the ARD criterion for $z = 1$, $\lambda = 1$ and $J = \{2, 3\}$. PSA allocated 100 trials for the minimum spacing constant $\delta = \frac{2}{140-1}$.

4 Privacy sets for Minimum-distance designs

In Section 2.1 we have proposed the prototype algorithm PSA, which was efficiently implemented for Bridge designs (Section 3). The reason was that some specific features of Bridge designs could be used in order to save both the computer memory and the computational time, so that even large design spaces could be dealt with. Analogously, we solve large-scale problems for another type of privacy sets in this section - privacy sets based on the Euclidean distance - defined by (3.7). These *Minimum-distance designs* (MDDs) seem to be very natural and adequate for the "hard" space-filling methods, but there is no appropriate efficient algorithm yet to handle these restrictions.

4.1 Motivation

It seems straightforward to require space-filling designs to satisfy Minimum-distance constraints. Interestingly, there is some additional motivation arising from other scientific disciplines. More precisely, the same type of "privacy sets" was discovered in some of the patterns in the nature. Most of the information presented in this section can be found in the popular-scientific article [81].

One of the most recent examples stems from biology: In the eye of a chicken, the color-sensitive cone cells that carpet the retina exhibit a curious mix of randomness and regularity, as shown in Figure 3.8. If we separate the cone types according to the color,

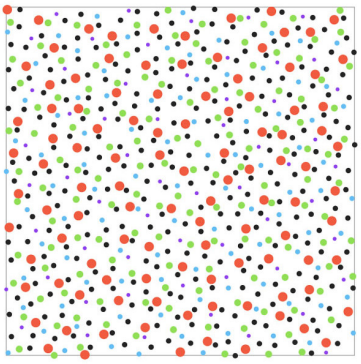


Figure 3.8: Apparent disorder in the arrangement of green, blue, red, violet and double-type (black) cone photoreceptors in a chicken's retina. (Image source: [81].)

we can see that they never get "too close to each other". In other words, there is a Minimum-distance privacy set (in the terminology of the cited article, an *exclusion region*)

of a certain size δ corresponding to each type of the cone cells, see Figure 3.9. This phenomenon was named *hyperuniformity*.

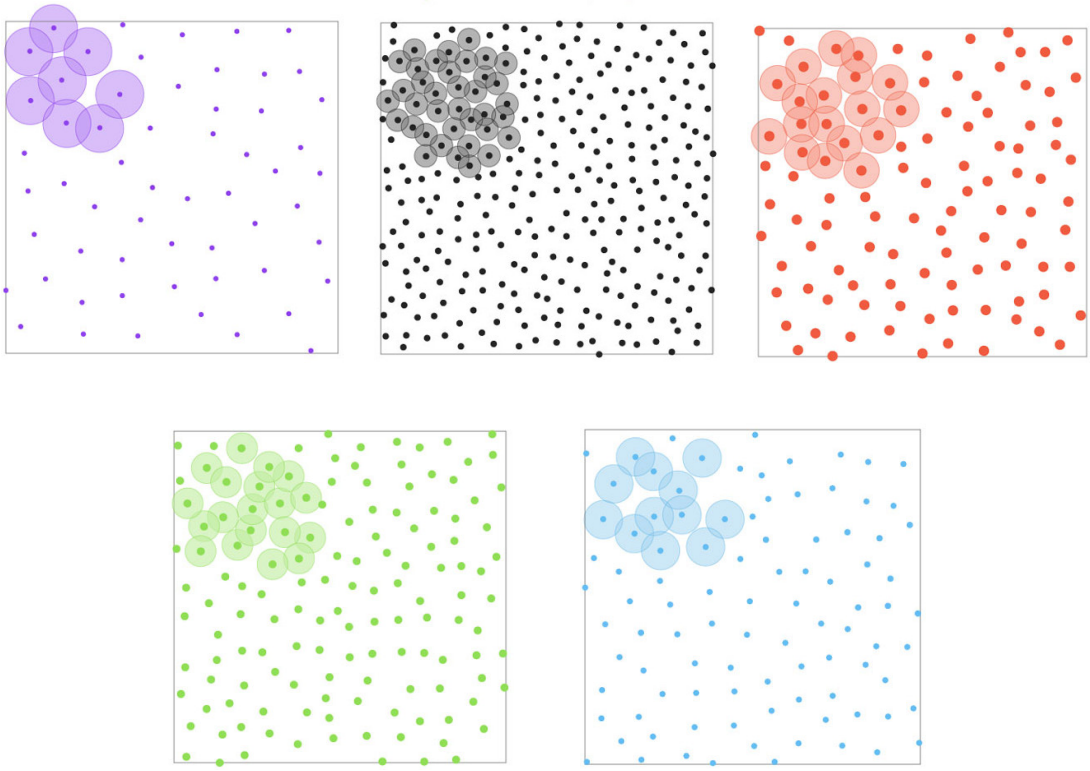


Figure 3.9: Minimum-distance privacy sets (exclusion regions) for five different types of cone cells, each of a different size. (Image source: [81].)

Beyond bird's eyes the similar kind of the "disordered hyperuniformity" was found in quasicrystals, as well as in random matrices, the large-scale structure of the universe, quantum ensembles, and soft-matter systems like emulsions and colloids. Another simple example of this pattern is a "shaken box of marbles" which fall into an arrangement called maximally random jammed packing.

4.2 Voronoi diagrams and Delaunay triangulations

Every type of the privacy sets has its own characteristics that must be taken into account when applying PSA. Step 2 in Algorithm 1 (Greedy Procedure) is crucial in the PSA implementation, see Section 2.1. The essence of this step is to increment the current design with the best (or at least a good-enough) permissible point $x \in \mathcal{X} \setminus \mathcal{P}(\xi)$. However,

solely generating any permissible x may be a difficult task in the case of Minimum-distance restrictions. Of course, a straightforward solution is to sample uniformly on $\mathcal{X} = [0,1]^d$, rejecting impermissible points $x \in \mathcal{P}(\xi)$. It is not hard to imagine though how ineffective this method becomes - in higher dimensions, or simply in case of a small permissible area, as shown for instance in Figures 3.14c and 3.16.

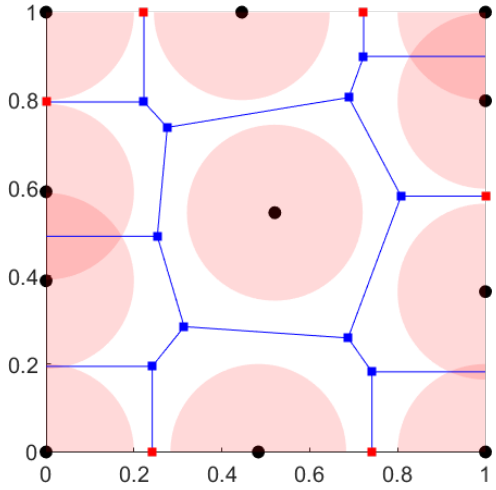


Figure 3.10: An example of a Minimum-distance design in $\mathcal{X} = [0,1]^2$ with $\delta = 0.2$. The blue squares denote the vertices of the Voronoi diagram. The red squares represent permissible intersections of the edges of the Voronoi diagram with the boundary of $[0,1]^2$ (see Lemma 4.7).

The blue squares displayed in Figure 3.10 represent the main "tool" of the method proposed in this section - the vertices of the so-called *Voronoi diagram* (VD). Formally, let $\mathcal{C}$ be a set of N points in $\mathcal{X} \subset \mathbb{R}^d$, called the generating points. For each $c \in \mathcal{C}$ let

$$\mathcal{R}_c = \{x \in \mathbb{R}^d : \rho(x,c) \leq \rho(x,\tilde{c}) \text{ for all } \tilde{c} \in \mathcal{C} \setminus \{c\}\}$$

denote its Voronoi cell (also called the Voronoi region).

The Voronoi diagram (or the Voronoi tessellation) is then the polyhedral cell complex given by cells $\{\mathcal{R}_c\}_{c\in\mathcal{C}}$, where a *polyhedral cell complex* denotes a collection of polyhedral cells and its faces of all dimensions. In other words, Voronoi diagram of $\mathcal{C}$ (VD($\mathcal{C}$)) divides $\mathcal{X}$ into N regions $\{\mathcal{R}_c\}_{c\in\mathcal{C}}$, such that region $\mathcal{R}_c$ consists of points of $\mathbb{R}^d$ whose distance from $c \in \mathcal{C}$ is smaller than (or at most equal to) the distance from any other generating point. We remark that a VD is not a "decomposition" in a strictly mathematical sense, because some of the points of $\mathbb{R}^d$ may belong to two or more Voronoi cells. The former

and the following definitions, as well as the theoretical information on VDs, are mostly taken from [4], eventually from [3].

For clarity, we provide the following definition.

Definition 4.1 *In d-dimensional Euclidean space, let k-flat denote its k-dimensional affine subspace, and let $(d-1)$-sphere denote the set of points $\{x \in \mathbb{R}^d : \rho(x,c) = r\}$, where $c \in \mathbb{R}^d$ is the centre of the $(d-1)$-sphere and $r \in \mathbb{R}$ is its radius.*

Assumption 4.1 *We assume no $d+1$ points of $\mathcal{C}$ are located in the same $(d-1)$-flat, and, simultaneously, no $d+2$ points lie on the same $(d-1)$-sphere. Then, points $\mathcal{C}$ are considered to be in a* general position*.*

In the following, we always assume the general position of the points $\mathcal{C}$, since this can be in practice easily achieved by a slight shift of the particular points.

Note that while the former assumption of Assumption 4.1 corresponds to the standard *general linear position* of points $\mathcal{C}$ in $\mathbb{R}^d$, the latter one relates to the "lifting" transformation of $\mathcal{C}$ into a space one dimension higher. There is an equivalence between the problem of constructing the VD($\mathcal{C}$) and the problem of finding the convex hull of the set of points in $\mathbb{R}^{d+1}$ obtained by giving each point $c \in \mathcal{C}$ an extra coordinate equal to $\|c\|^2$. In fact, one of the most fundamental algorithms of computing VDs is based on this very principle, see Section 4.4. This close relationship between VDs in $\mathbb{R}^d$ and convex polyhedra in $\mathbb{R}^{d+1}$ relies on the fact that the transformation onto the paraboloid in $\mathbb{R}^{d+1}$ maps $(d-1)$-spheres in $\mathbb{R}^d$ bijectively onto the nonempty intersections of the paraboloid in $\mathbb{R}^{d+1}$ with the non-vertical hyperplanes (see, e.g., Section 4.1 in [2]). This implies that the latter assumption of the "non-cocircularity" of $d+2$ points exactly corresponds to the lifted points being in a general linear position.

VD is a so-called *face-to-face cell complex* in $\mathbb{R}^d$ that consists of the polyhedral faces of various dimensions. (A cell complex is face-to-face, if a nonempty intersection of two cells is a face of both cells.) In particular, each Voronoi cell is a convex polyhedron (bounded or unbounded). Moreover, Assumption 4.1 guarantees that exactly $d+1$ edges, $\binom{d+1}{2}$ facets and $d+1$ cells meet at each Voronoi vertex. Such complexes are called *simple cell complexes* in $\mathbb{R}^d$.

The concept of the Voronoi diagrams is tightly related to the following dual structure: Let any two points $c, \tilde{c} \in \mathcal{C}$ be connected by an *Delaunay edge*, iff there exists a *facet* (i.e., a $(d-1)$-dimensional face) in $\mathrm{VD}(\mathcal{C})$ shared by both $\mathcal{R}_c$ and $\mathcal{R}_{\tilde{c}}$. Since VDs are structurally simple cell complexes (as discussed in above), their duals are necessarily the so-called *simplicial complexes* (and vice versa), where a "simplicial complex" is a polyhedral cell complex such that all of its cells are d-dimensional simplices. In some literature, such a simplicial complex is called simply a *triangulation*. For simplicity, we have also decided to use term "triangulation" throughout the thesis to denote a simplicial complex in any dimension (not just in the dimension $d = 2$).

Provided the generating points are in a general position (as defined in above), the set of the Delaunay edges forms a unique triangulation in $\mathbb{R}^d$ - the so-called *Delaunay triangulation* of $\mathcal{C}$ ($\mathrm{DT}(\mathcal{C})$). That is, Assumption 4.1 guarantees that no Delaunay simplex is degenerate, and, simultaneously, ensures uniqueness of the triangulation.

Delaunay triangulation of the point set $\mathcal{C}$ can also be equivalently defined to contain all those d-simplices with vertices in $\mathcal{C}$ whose circum-hyperspheres do not contain any other points of $\mathcal{C}$ (see, e.g., [4]). This property is often referred to as the *empty-hypersphere-property* (or more commonly in $\mathbb{R}^2$ - *empty-circle-property*). In this work, we also refer to this assumption as the "DT condition". To check the DT condition one needs to construct the circum-hypersphere of every simplex. The center of this hypersphere is obviously a Voronoi vertex, which is called *dual* to the original simplex. In the same fashion, any point $c \in \mathcal{R}_c$ is considered dual to the Voronoi cell $\mathcal{R}_c$ itself.

We would like to note that the terminology varies in the literature: For instance, "triangulation" is sometimes considered exclusively a two-dimensional simplicial complex, and is replaced by term "tessellation" in higher dimensions. On the other hand, "tessellation" can also denote a cell complex that admits other polyhedrical cells (other than simplicial) as well - for example in the case of $d+2$ generating points lying on the same $(d-1)$-sphere. In such cases, [51] operates with the term "pre-triangulation" in $\mathbb{R}^2$, eventually "pre-tessellation" in higher dimensions.

4.2.1 Voronoi diagrams and Delaunay triangulations in $\mathbb{R}^2$

In two-dimensional Euclidean space, simplices are merely triangles and every Voronoi vertex is equidistant to exactly three generating points (under Assumption 4.1). Every Delaunay edge connecting points $c, \tilde{c}$ is dual to the Voronoi edge shared by the Voronoi cells $\mathcal{R}_c, \mathcal{R}_{\tilde{c}}$. Moreover, these two mutually dual edges are *orthogonal* (but do not necessarily have to intersect each other).

Let us now proceed by clarifying the connection of Voronoi diagrams to PSA for MDDs. The following Lemma 4.2 uses the term *bounded* Voronoi cell; we remark that this refers to the VD in $\mathbb{R}^2$, not in $[0,1]^2$, where it would be automatically satisfied for each Voronoi cell.

Lemma 4.2 *Let $\mathcal{C}$ be a set of generating points in $\mathbb{R}^2$. Let $c \in \mathcal{C}$ be such that its Voronoi cell $\mathcal{R}_c$ is bounded. For every $x \in \mathcal{R}_c$, let us measure the distance $\tilde{\rho} : x \to \rho(x, c)$. Then, for any point $x \in \mathcal{R}_c \setminus \{c\}$, there exists a Voronoi vertex v of $\mathcal{R}_c$, such that the directional derivative of $\tilde{\rho}$ in x and in the direction of v is non-negative, i.e., $\langle \nabla\tilde{\rho}(x), v - x\rangle \geq 0$.*

Proof
First, let us construct a ray starting in c and passing through x. For a bounded $\mathcal{R}_c$, the intersection of this ray and the border of the convex polyhedron $\mathcal{R}_c$ is a single point y (it holds that c lies in the interior of $\mathcal{R}_c$). Point y can either be a Voronoi vertex itself, or it can lie in some Voronoi edge e. In the former case, simply denote the Voronoi vertex by v. In case of the latter, let v denote the vertex of e for which it holds that angle $\angle cyv$ is obtuse. (Note that if $cy \not\perp e$ there always exists such an obtuse angle. In case $cy \perp e$ let v be any of the two Voronoi vertices of edge e.)

One can easily verify that $\tilde{\rho}$ is differentiable in $\mathcal{R}_c \setminus \{c\}$ and that $\nabla\tilde{\rho}(x) = \frac{x-c}{\rho(x,c)}$. Now, we have $\langle \nabla\tilde{\rho}(x), v - x\rangle = \langle \frac{x-c}{\rho(x,c)}, v - x\rangle = \rho(x,c)^{-1}\langle x - c, v - x\rangle$. When neglecting the positive constant $\rho(x,c)^{-1}$, we have

$$\begin{aligned}
\langle x - c, v - x\rangle &= \left\langle x - \frac{c+v}{2} + \frac{v-c}{2}, -\left(x - \frac{c+v}{2}\right) + \frac{v-c}{2}\right\rangle = \\
&= -\rho^2\left(x, \frac{c+v}{2}\right) + \left\langle x - \frac{c+v}{2}, \frac{v-c}{2}\right\rangle - \left\langle \frac{v-c}{2}, x - \frac{c+v}{2}\right\rangle + \frac{\rho^2(v,c)}{4} = \\
&= -\rho^2\left(x, \frac{c+v}{2}\right) + \frac{\rho^2(v,c)}{4}. \qquad (3.9)
\end{aligned}$$

We know that the angle $\angle cyv$ is obtuse (or at least right) and also that x lies in the line segment $\overline{cy}$. Thus, the angle $\angle cxv$ is obtuse (or at least right) as well and Thales's theorem implies that x lies in the interior of the circle with the center at $\frac{c+v}{2}$ and the radius $\frac{\rho(v,c)}{2}$. In other words, it holds that

$$\rho(x, \frac{c+v}{2}) \leq \frac{\rho(v,c)}{2}. \tag{3.10}$$

Now applying (3.10) onto (3.9), we get

$$\langle x-c, v-x\rangle \geq -\frac{\rho^2(v,c)}{4} + \frac{\rho^2(v,c)}{4} = 0,$$

which concludes the proof. □

The key role of Voronoi vertices in our algorithm is justified by the following theorem.

Theorem 4.3 *Let ξ be a permissible Minimum-distance design in $\mathcal{X} = [0,1]^2$ with privacy sets given by* (3.7) *and let us have the Voronoi diagram on ξ. Let $q \notin \xi$ be a permissible point in $\mathcal{X}$, such that $q \in \mathcal{R}_c$ for some bounded $\mathcal{R}_c$, $c \in \xi$. Then, there exists a Voronoi vertex v, such that the whole line segment $\overline{vq}$ is permissible. In particular, the Voronoi vertex v itself is permissible.*

Proof
Let v be constructed identically as in Lemma 4.2, with q substituted for x. For function $f : \lambda \to \rho(\lambda q + (1-\lambda)v, c)$, $\lambda \in [0,1]$, it holds that

$$f'(\lambda) = \langle \nabla\tilde{\rho}(\lambda q + (1-\lambda)v), q - v\rangle = -\lambda^{-1}\langle \nabla\tilde{\rho}(\lambda q + (1-\lambda)v), v - (\lambda q + (1-\lambda)v)\rangle,$$

where $\tilde{\rho}$ is defined as in Lemma 4.2. Now denoting $x_\lambda = \lambda q + (1-\lambda)v$ and employing Lemma 4.2, we get

$$f'(\lambda) = -\lambda^{-1}\langle \nabla\tilde{\rho}(x_\lambda), v - x_\lambda\rangle \leq 0.$$

Thus, function f is non-increasing, which implies $\rho(x_\lambda, c) \geq \rho(q, c)$. From the permissibility of q it follows that $\rho(x_\lambda, c) > \delta$ for every $x_\lambda \in \overline{vq}$.

Since $x_\lambda = \lambda q + (1-\lambda)v \in \mathcal{R}_c$, we know that $\rho(x_\lambda, \tilde{c}) \geq \rho(x_\lambda, c) > \delta$ for all design points $\tilde{c} \in \xi$ and all values of $\lambda \in [0,1]$, which concludes the proof. □

Theorem 4.3 allows for the following metaphor: For a minimum distance design given, the design region is partly blocked by the privacy sets of the design points. The remaining

space, permissible for the possible augmentation point, can represent a "Swiss-cheese-like" gallery. If we position guards to the Voronoi vertices, then these guards can oversee each point of the gallery (provided that the gallery consists only of the *bounded* Voronoi cells, which is discussed in more detail in Section 4.3). One can also be interested in finding the minimum number of the necessary guards, which leads to a particular variation of the well-studied optimization problem of the computational geometry - the so-called *art gallery problem.*

4.2.2 Voronoi diagrams and Delaunay triangulations in $\mathbb{R}^d$

In a general-dimensional Euclidean space, we assume the following conjecture to hold. Our assumption is based not only on the geometric intuition, but also on the numerical results of Privacy sets algorithm for MDDs (see Section 4.7). The proof could extend the ideas of the proof of Theorem 4.3, but the complexity of the geometry of VDs in $\mathbb{R}^d$ makes this generalization not straightforward at all. One should also keep in mind that PSA is a heuristic method, which means that its theoretical justification is welcome, but not crucial.

Conjecture 4.4 *Let ξ be a permissible Minimum-distance design in $\mathcal{X} = [0,1]^d$ with privacy sets given by* (3.7) *and let us have the Voronoi diagram on ξ. Let $q \notin \xi$ be a permissible point in $\mathcal{X}$, such that $q \in \mathcal{R}_c$ for some bounded $\mathcal{R}_c$, $c \in \xi$. Then, there exists a Voronoi vertex v, such that the whole line segment $\overline{vq}$ is permissible. In particular, the Voronoi vertex v itself is permissible.*

4.3 Power diagrams and regular triangulations

Theorem 4.3 assumes the permissible point q to belong to a *bounded* Voronoi cell. However, it can easily happen that the Voronoi cell containing q is unbounded. This can be avoided by augmenting $\mathcal{C} \subset \mathbb{R}^2$ with three extra points forming a large triangle, the so-called *super-simplex*, which contains the whole design space $\mathcal{X}$. Moreover, the vertices of the super-simplex have to be located far enough not to change the VD on $\mathcal{X}$; in other words, no point from $\mathcal{X}$ is allowed to lie in a Voronoi cell of one of these three additional points. But there are only three unbounded Voronoi cells in the VD - exactly those corresponding to the three vertices of the super-simplex. Thus, the assumptions of Theorem

4.3 hold for every $q \in \mathcal{X}$. However, if the "permissible" Voronoi vertex v lies outside of $\mathcal{X}$, it cannot be used as a starting point for the design augmentation (see Step 9 of Algorithm 5). We discuss this issue more closely in this section.

Intuitively, the red squares in Figure 3.10 suggest to search for the intersections of the faces of the VD with the boundary of the cuboid design space (in addition to the Voronoi vertices lying in $\mathcal{X}$). For the two-dimensional space, in Lemma 4.7 we will prove that these intersections can be determined by constructing the so-called *power vertices* on the particular edge of $\mathcal{X} = [0, 1]^2$.

The power vertices are simply vertices of the so-called *power diagram*, as defined in the following. Let $w_c \in \mathbb{R}$ be a (not necessarily non-negative) weight corresponding to some $c \in \mathbb{R}^d$ and denote this couple by $g = (c, w_c)$. Now define the "power" of a point $x \in \mathbb{R}^d$ with respect to g as

$$\mathrm{pow}(x, g) = \rho^2(x, c) - w_c. \tag{3.11}$$

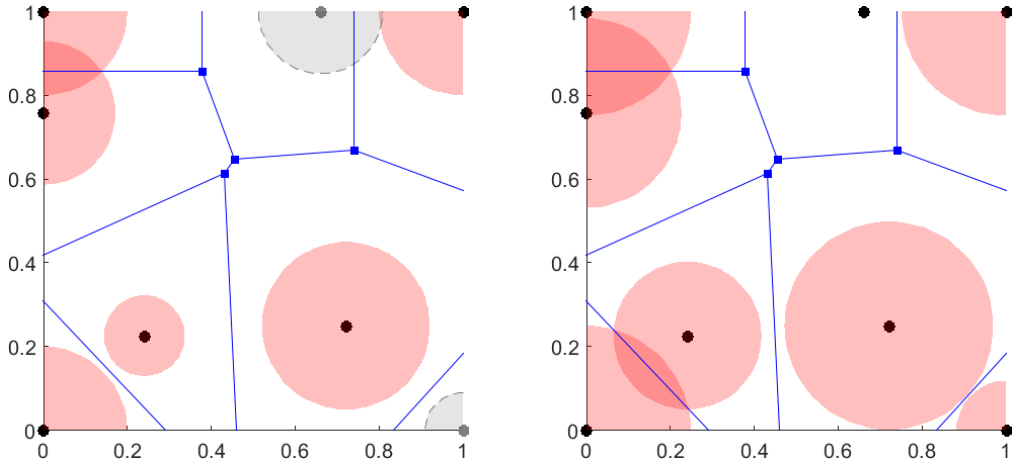
For every g from a given finite set $\mathcal{G}$, the region

$$\mathcal{R}_g = \{x \in \mathbb{R}^d : \mathrm{pow}(x, g) \leq \mathrm{pow}(x, \tilde{g}) \text{ for all } \tilde{g} \in \mathcal{G} \setminus \{g\}\}$$

denotes the power cell of g. The collection of all power cells $\mathcal{R}_g$, $g \in \mathcal{G}$ (and all their faces included), is called the *power diagram* of $\mathcal{G}$ ($\mathrm{PD}(\mathcal{G})$), see Figure 3.11 for illustration. Similarly to VDs, PDs are face-to-face cell complexes with each cell being a convex polyhedron.

PDs are clearly generalizations of VDs defined in Section 4.2, with each of the generating points having its own weight - the greater the weight of a generating point, the more it "attracts" other points from $\mathcal{X}$. If w_c is positive, it can be represented by a hypersphere with centre c and radius $r_c = \sqrt{w_c}$. Note that our definition of a power diagram is taken from [4]. In some of the previous works on the topic (e.g., in [2]), the term "power diagram" relates to the non-negative weights only and is replaced by term "weighted Voronoi diagram" when negative weights are admissible as well.

Now assume $w_c = r_c^2 > 0$ for some $g = (c, w_c) \in \mathcal{G}$. For a point x lying outside of the hypersphere S with centre c and radius r_c, the Pythagorean theorem yields the



(a) (b)

Figure 3.11: An illustration of the power diagrams on one of the faces of the cube $\mathcal{X} = [0,1]^3$ for the example displayed in Figure 3.15. Various sets of weights can yield the same diagram. The red circles denote positive weights, the dashed grey circles negative ones. We have shifted these weights into the non-negative values, keeping the resulting PD unchanged (Figure 3.11b).

geometrical representation of $\mathrm{pow}(x, g)$ as the square of the length of a line segment from x to a point t of tangency with hypersphere S, as illustrated in Figure 3.12. Point x lying exactly on S has $\mathrm{pow}(x, g) = 0$, and the power of a point inside S with respect to g is negative.

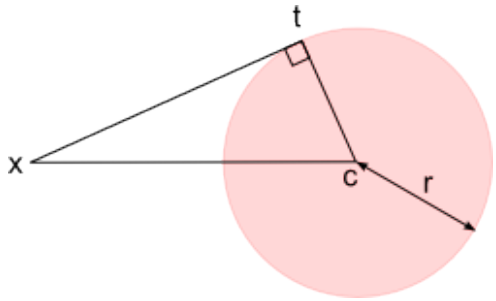


Figure 3.12: An illustration of the power function (3.11) in dimension $d = 2$.

Let $g = (c, w_c)$ and $\tilde{g} = (\tilde{c}, w_{\tilde{c}})$ be two weighted generating points in $\mathbb{R}^d$, $c \neq \tilde{c}$. The set of points x satisfying $\mathrm{pow}(x, g) = \mathrm{pow}(x, \tilde{g})$ forms a hyperplane perpendicular to the line connecting c and $\tilde{c}$, given by

$$h : 2\langle x, c - \tilde{c}\rangle = w_{\tilde{c}} - w_c - \|\tilde{c}\|^2 + \|c\|^2. \tag{3.12}$$

Hyperplane h is called the *radical axis* of g and $\tilde{g}$.

PDs are uniquely determined by the set of the generating points and their corresponding weights. This does not hold in the opposite direction, though: Various sets of weights may yield the same PD. For instance, equal weights $w_c = const$ for all generating points c obviously yield the same diagram - the standard Voronoi diagram, no matter the size of the constant. The equation of the radical axis (3.12) of two weighted generating points $g = (c, w_c)$ and $\tilde{g} = (\tilde{c}, w_{\tilde{c}})$ implies that if the difference $w_{\tilde{c}} - w_c$ is constant, the radical axis remains the same. Thus, the simple adding of a constant to the weights of all generating points does not change the resulting diagram.

Also note that one of the main differences between PDs and VDs is that there does not necessarily have to exist a power cell corresponding to every generating point in a PD. The power cell of such a point with the weight too small (in comparison to the other generating points) is simply empty (see [2] for more information).

Analogously to Assumption 4.1, the general linear position of points $\mathcal{G} \subset \mathbb{R}^d$ is assumed, that is, no $d+1$ points lie in the same $(d-1)$-flat. The second assumption of no $d+2$ "cocircular" points is replaced by its weighted analogue: There exists *no* point from $\mathbb{R}^d$ for which it holds that its power with respect to at least $d+2$ generating points is equal. This condition is equivalent to the general linear position of points $g = (c, w_c) \in \mathcal{G}$ lifted to $\mathbb{R}^{d+1}$ by obtaining an extra coordinate equal to $\|c\|^2 - w_c$ (see, for instance, Definition 5 in [76]).

The geometric dual (in the sense described in Section 4.2) of a power diagram $\mathrm{PD}(\mathcal{G})$ will be called a *regular triangulation* - $\mathrm{RT}(\mathcal{G})$, since this, what might be called a "weighted Delaunay triangulation", has a property of *regularity*, which is explained as follows: First, it is possible to show that for any simplex Δ there exists a unique sphere σ *orthogonal* to each of its vertices. The orthogonality in this sense is defined as $\rho^2(c, s) = w_c + w_s$, where c is a vertex of Δ with weight w_c, s is the centre of σ and w_s is the squared radius of σ. Clearly, sphere σ can be also viewed as a weighted point in $\mathbb{R}^d$.

Similarly, one could denote two weighted points $\tilde{g} = (\tilde{c}, w_{\tilde{c}}), \sigma = (s, w_s)$ as *obtuse* iff

$$\rho^2(\tilde{c}, s) > w_{\tilde{c}} + w_s, \tag{3.13}$$

and, subsequently, for a given set of the generating points $\mathcal{G}$, call simplex Δ *regular*, if any other point of $\mathcal{G}$ not belonging to Δ is obtuse to σ (see [4] for more detailed information). A triangulation (i.e., a simplicial complex), is then called regular, if it contains regular simplices only. Note that the weighted points with empty power cells are not included in the RT: they may be simply left out.

Orthogonality of $\sigma = (s, w_s)$ to each (weighted) vertex g in fact means $w_s = \mathrm{pow}(s, g)$ for every g, that is, s can be viewed as a *weighted circumcenter* of Δ. DT condition in the standard case says that no generating point is closer to the circumcenter of a simplex than the vertices of the simplex themselves. A weighted analogue to the DT condition - "RT condition" - is ensured by the assumption of obtuseness of every $\tilde{g} \in \mathcal{G}, \tilde{g} \notin \Delta$, since (3.13) is equivalent to $w_s < \mathrm{pow}(s, \tilde{g})$. In case of the non-weighted generating points, this assumption coincides with the empty-hypersphere-property defined in Section 4.2.

Let us now proceed to explaining the role of the power diagrams and the power itself in our Privacy sets algorithm. First, we need to generalize the class of Minimum-distance designs by allowing δ to be a function defined on $\mathcal{X}$:

Definition 4.5 *Let $\mathcal{X}$ be a design space and let $\delta : \mathcal{X} \to \mathbb{R}_0^+$. Then ξ is a* weighted Minimum-distance design *if the privacy set of any $x \in \mathcal{X}$ is given by*

$$\mathcal{P}(x) = \{y : \rho(x, y) \leq \delta(x)\}. \tag{3.14}$$

Let $\xi \subset \mathcal{X}$ be a weighted MDD and let $q \notin \xi$ be a permissible point in $\mathcal{X}$. The relation between weighted MDDs and PDs is then explained by the following equivalences:

$$\rho(q, c) > \delta(c) \quad \Leftrightarrow \quad \rho^2(q, c) - \delta^2(c) > 0 \quad \Leftrightarrow \quad \mathrm{pow}(q, (c, \delta^2(c))) > 0 \quad \text{for all} \quad c \in \xi.$$

Thus, point $q \in \mathcal{X}$ is permissible iff its power with respect to each design point $c \in \xi$ with the weight given by $w_c = \delta^2(c)$ is positive. Note that in case $\delta(c) = 0$ for some $c \in \xi$, the power is simply the Euclidean distance, which is positive for every $q \neq c$.

4.3.1 Power diagrams and regular triangulations in $\mathbb{R}$

Consider the situation in the two-dimensional space, as depicted in Figure 3.10. The relatively uncomplicated geometry of VDs and DTs in $\mathbb{R}^2$ allowed us to derive some key results in Section 4.2.1. Nevertheless, as argued in the previous section, finding a

permissible Voronoi vertex makes sense only as far as it lies in the design space. Otherwise, the intersection points denoted by the red squares in Figure 3.10 are of interest.

In the following, we will show that these intersection points (the red squares) can be determined as vertices of the one-dimensional power diagram. Let us first think of the geometry of PDs and RTs in $\mathbb{R}$. A one-dimensional simplex is a simple interval with its two extreme points and its power vertex. A triangulation is just a collection of consecutive intervals, which is not unique, though: the power cells of some of the generating points may simply be empty and these points are then "left out" from the triangulation.

For power diagrams in $\mathbb{R}$, an analogue of Theorem 4.3 can be proved quite straightforwardly:

Theorem 4.6 *Let ξ be a permissible weighted Minimum-distance design with the privacy sets given by* (3.14)*. Let $\mathcal{G} = \{(c, w_c)\}$ be a set of weighted generating points, where $c \in \xi$ and its weight is given by $w_c = \delta^2(c)$. Let $q \notin \xi$ be a permissible point in $\mathcal{X} = [0, 1]$, such that $q \in \mathcal{R}_{\tilde{g}}$, where $\mathcal{R}_{\tilde{g}}$ is bounded. Then, there exists a power vertex v of $PD(\mathcal{G})$, such that the whole line segment $\overline{vq}$ is permissible. In particular, the power vertex v itself is permissible.*

Proof
In $\mathbb{R}$, power cells are just intervals: Since $\mathcal{R}_{\tilde{g}}$ is bounded, it has two extreme points - two power vertices. Let v be its power vertex such that q lies between $\tilde{g}$ and v, i.e., $\mathrm{pow}(v, \tilde{g}) \geq \mathrm{pow}(q, \tilde{g})$.

Permissibility of q means that for all $c \in \xi$ it holds that

$$\rho(q, c) > \delta(c).$$

Since both sides of the inequality 4.3.1 are non-negative, 4.3.1 is equivalent to $\rho^2(q, c) - \delta^2(c) > 0$, which means $\mathrm{pow}(q, g) > 0$ for all $g = (c, \delta^2(c)) \in \mathcal{G}$.

We have $\mathrm{pow}(v, \tilde{g}) \geq \mathrm{pow}(q, \tilde{g}) > 0$ and, simultaneously, $v \in \mathcal{R}_{\tilde{g}}$ implies $\mathrm{pow}(v, g) \geq \mathrm{pow}(v, \tilde{g})$ for any $g \in \mathcal{G}$. Thus, it holds that $\mathrm{pow}(v, g) > 0$ for all $g \in \mathcal{G}$, or, in other words, v is permissible.

For any $x_\lambda = \lambda v + (1-\lambda)q$, $\lambda \in (0,1)$, it clearly holds that $\mathrm{pow}(x_\lambda, \tilde{g}) \geq \mathrm{pow}(q, \tilde{g}) > 0$ and this, together with the fact that $x_\lambda \in \mathcal{R}_{\tilde{g}}$, yields the permissibility of x_λ for any $\lambda \in (0,1)$. $\square$

4.3.2 Intersection points in case $d = 2$

Assume now we have a (non-weighted) MDD $\xi \subset \mathcal{X} = [0,1]^2$ and a permissible point $q \in \mathcal{X}$. Theorem 4.3 yields a permissible Voronoi vertex v. However, let us assume $v \notin \mathcal{X}$. In the next step, we choose that edge $\mathcal{F}$ of $\mathcal{X}$ which is intersected by the line segment $\overline{vq}$. Denote the intersection point by u - Theorem 4.3 ensures its permissibility.

Let us now consider those design points $\xi_s \subseteq \xi$, whose privacy sets do intersect with $\mathcal{F}$, and project them onto $\mathcal{F}$ by an orthogonal projection function $P_{\mathcal{F}}$. Denote the projected sub-design (of size not greater than N) by $\xi_s^{\mathcal{F}} = P_{\mathcal{F}}(\xi_s)$. For each projected $c^{\mathcal{F}} \in \xi_s^{\mathcal{F}}$, define its privacy set by

$$\mathcal{P}(c^{\mathcal{F}}) = \{y \in \mathbb{R} : |y - c^{\mathcal{F}}| \leq \sqrt{\delta^2 - \rho^2(c, c^{\mathcal{F}})}\},$$

where $\delta^2 - \rho^2(c, c^{\mathcal{F}}) \geq 0$ is guaranteed by the assumption of the nonempty intersection of $\mathcal{P}(c)$ and $\mathcal{F}$ for every $c \in \xi_s$.

If we consider $\mathcal{F}$ to be our new design space, we can directly employ Theorem 4.6: The weighted MDD $\xi_s^{\mathcal{F}}$ has weights given by $\delta^{\mathcal{F}} : \delta^{\mathcal{F}}(c^{\mathcal{F}}) = \sqrt{\delta^2 - \rho^2(c, c^{\mathcal{F}})}$. The way of constructing privacy sets of $\xi_s^{\mathcal{F}}$ implies equivalence of the permissibility in $\mathcal{X}$ with respect to δ and the permissibility in $\mathcal{F}$ with respect to $\delta^{\mathcal{F}}$. Thus, u is permissible in $\mathcal{F}$ and Theorem 4.6 yields a permissible power vertex $v^{\mathcal{F}}$.

The assumptions of Theorem 4.6 are satisfied due to the presence of the weighted super-simplex (in this case an interval) added for every edge $\mathcal{F}$ separately. If the weights of the two vertices of the super-simplex are small enough (see Section 4.5 for the details), the corresponding unbounded power cells do not intersect $\mathcal{F}$ at all; as a result, each point from $\mathcal{F}$ lies in a power cell that is *bounded*.

Power vertex $v^{\mathcal{F}}$ may but does not necessarily have to lie in $\mathcal{F}$ - in case of the latter, we project $v^{\mathcal{F}}$ onto the closest vertex of $\mathcal{F}$ (i.e., one of the points $(0,0)$, $(0,1)$, $(1,0)$ and $(1,1)$). This vertex must be permissible due to Theorem 4.6.

The following lemma specifies the character of the power vertices $v^{\mathcal{F}}$.

Lemma 4.7 *Let $\mathcal{G} = \{g\}$ be the set of the weighted generating points in $[0,1]^2$. Let t belong to the edge $\mathcal{F}$ of the square $[0,1]^2$, and let $P_{\mathcal{F}} : \mathcal{X} \to \mathcal{F}$ denote the orthogonal projection function. Then for any $g = (c, w_c), \tilde{g} = (\tilde{c}, w_{\tilde{c}}) \in \mathcal{G}$, it holds that*

$$pow(t, g) = pow(t, \tilde{g}) \quad \textit{iff} \quad pow(t, g^{\mathcal{F}}) = pow(t, \tilde{g}^{\mathcal{F}}) \quad \textit{for } t \in \mathcal{F},$$

where

$$\begin{aligned} g^{\mathcal{F}} &= (c^{\mathcal{F}}, w_{c^{\mathcal{F}}}), \quad \tilde{g}^{\mathcal{F}} = (\tilde{c}^{\mathcal{F}}, w_{\tilde{c}^{\mathcal{F}}}), \\ c^{\mathcal{F}} &= P_{\mathcal{F}}(c), \quad \tilde{c}^{\mathcal{F}} = P_{\mathcal{F}}(\tilde{c}), \\ w_{c^{\mathcal{F}}} &= w_c - \rho^2(c, c^{\mathcal{F}}), \quad w_{\tilde{c}^{\mathcal{F}}} = w_{\tilde{c}} - \rho^2(\tilde{c}, \tilde{c}^{\mathcal{F}}). \end{aligned} \tag{3.15}$$

Proof

Without loss of generality, let $x_1 = 1$ for all $x \in \mathcal{F}$ (other options would be $x_1 = 0$, $x_2 = 1$ and $x_2 = 0$ for all $x \in \mathcal{F}$). Thus, it holds that

$$\rho^2(c, c^{\mathcal{F}}) = (1 - c_1)^2 \quad \text{and} \quad \rho^2(\tilde{c}, \tilde{c}^{\mathcal{F}}) = (1 - \tilde{c}_1)^2. \tag{3.16}$$

Now $\mathrm{pow}(t, g) = \mathrm{pow}(t, \tilde{g})$ is equivalent to

$$\rho^2(t, c) - w_c = \rho^2(t, \tilde{c}) - w_{\tilde{c}},$$

$$(t_1 - c_1)^2 + (t_2 - c_2)^2 - w_c = (t_1 - \tilde{c}_1)^2 + (t_2 - \tilde{c}_2)^2 - w_{\tilde{c}}.$$

Substituting $t_1 = 1$ and employing (3.16) we get

$$\rho^2(c, c^{\mathcal{F}}) + (t_2 - c_2)^2 - w_c = \rho^2(\tilde{c}, \tilde{c}^{\mathcal{F}}) + (t_2 - \tilde{c}_2)^2 - w_{\tilde{c}},$$

which can be rewritten as

$$\rho^2(c, c^{\mathcal{F}}) + (t - c^{\mathcal{F}})^2 - w_c = \rho^2(\tilde{c}, \tilde{c}^{\mathcal{F}}) + (t - \tilde{c}^{\mathcal{F}})^2 - w_{\tilde{c}},$$

$$\rho^2(t, c^{\mathcal{F}}) + \rho^2(c, c^{\mathcal{F}}) - w_c = \rho^2(t, \tilde{c}^{\mathcal{F}}) + \rho^2(\tilde{c}, \tilde{c}^{\mathcal{F}}) - w_{\tilde{c}},$$

$$\rho^2(t, c^{\mathcal{F}}) - w_{c^{\mathcal{F}}} = \rho^2(t, \tilde{c}^{\mathcal{F}}) - w_{\tilde{c}^{\mathcal{F}}},$$

which completes the proof. □

For a power vertex $v^{\mathcal{F}}$ lying on edge $\mathcal{F}$, the power with respect to at least 2 points of $\xi_s^{\mathcal{F}}$ is equal (or with respect to *exactly* two points if the general position is assumed).

According to Lemma 4.7, $v^{\mathcal{F}}$ belongs to a Voronoi edge of the original VD in $\mathbb{R}^2$, which in fact means that $v^{\mathcal{F}}$ is a point of intersection of the original VD and edge $\mathcal{F}$. Thus, computing power vertices on the edges of $\mathcal{X} = [0,1]^2$ yields the desired intersection points (the red squares in Figure 3.10).

Let us now recall the crucial Step 2 in Algorithm 1. For a non-maximal design ξ, we are searching for a permissible point in $\mathcal{X} = [0,1]^2$. The theoretical results derived so far lead us to the following scenario: First, compute the VD(ξ) and remember all the permissible Voronoi vertices that lie in $\mathcal{X}$. Second, enumerate all the edges $\mathcal{F}$ of $\mathcal{X}$ and for each one compute the PD of the projected (sub-)design PD($\xi_s^{\mathcal{F}}$). Store the permissible power vertices that lie in $\mathcal{F}$. If a power vertex does not lie in $\mathcal{F}$, store the corresponding vertex of $\mathcal{F}$ (and thus of $\mathcal{X}$) instead (i.e., one of the points $(0,0)$, $(0,1)$, $(1,0)$ and $(1,1)$).

Note that so far, we have projected only a particular subset of the design points - ξ_s. It is true that projecting only those points whose privacy sets intersect with $\mathcal{F}$ already yields a permissible power vertex. However, we opted for projecting all those points that are inevitable for computing *all* the intersections of the VD with the edge $\mathcal{F}$. The reason is the wider range of prospective starting points for augmenting the design, see Step 9 of Algorithm 5.

In other words, our goal is to keep the PD constructed on $\mathcal{F}$ the same as if all of the design points of ξ were projected. Projected points without the intersection receive negative weights. It is always correct to project all the points of ξ - the points too distant from $\mathcal{F}$ will be projected with the weight too small to affect the resulting PD. However, this approach could obviously significantly slow down the algorithm. For more details on the choice of the projected points, read further Section 4.6.

Another observation is that although both $v^{\mathcal{F}}$ and q are permissible, the permissibility of the line segment $\overline{v^{\mathcal{F}}q}$ is so far not guaranteed. It holds, nevertheless, that both $\overline{qu}$ and $\overline{uv^{\mathcal{F}}}$ are permissible, which means there exists a continuous permissible "path" connecting q and $v^{\mathcal{F}}$. This property is crucial for justifying the use of the intersection points in the way described above. It could be an interesting object of the further research to prove, eventually disprove, that the whole line segment $\overline{v^{\mathcal{F}}q}$ is permissible.

Last, we remark that there are other methods of computing intersections apart from the one proposed in this thesis, some of them mentioned for example in Section 2.4. of [58]. It is not the aim of this thesis to compare the speed of these methods. What is more, being able to compute power vertices can be necessary for instance in case of a weighted MDD (see Definition 4.5) and PSA can be in the future adapted to handle weighted MDDs as well.

4.3.3 Power diagrams and regular triangulations in $\mathbb{R}^d$

In a general-dimensional Euclidean space, we assume the following conjecture to hold. Analogously to Conjecture 4.4, the proof would be a non-trivial extension of the ideas of the proof of Theorem 4.6. Note that generalization of the proof of Theorem 4.6 could be very intricate since the geometry in the one-dimensional space is extremely simple in comparison to higher-dimensional spaces.

Conjecture 4.8 *Let ξ be a permissible weighted Minimum-distance design with the privacy sets given by* (3.14)*. Let $\mathcal{G} = \{(c, w_c)\}$ be a set of the weighted generating points, where $c \in \xi$ and its weight is given by $w_c = \delta^2(c)$. Let $q \notin \xi$ be a permissible point in $\mathcal{X} = [0,1]^d$, such that $q \in \mathcal{R}_{\tilde{g}}$, where $\mathcal{R}_{\tilde{g}}$ is bounded. Then, there exists a power vertex v of $PD(\mathcal{G})$, such that the whole line segment $\overline{vq}$ is permissible. In particular, the power vertex v itself is permissible.*

4.4 Computing VDs and DTs

Consider now the original problem of constructing Voronoi diagram on a given set of points. The duality of VDs and DTs suggests it is enough to calculate DT(ξ), implicitly obtaining VD(ξ), too. In the following, we list some of the algorithms computing Delaunay triangulations and point out the methods preferred in our implementation of PSA.

First, it is possible to show that the sum of two opposite angles of two adjacent triangles in a two-dimensional DT is at most 180° (and vice versa - a triangulation with this property is DT). This allows for a simple flip algorithm ([37]): Start with an arbitrary triangulation and search through all adjacent pairs of triangles. If two triangles violate

this property, flip their common edge, that is, switch the diagonal of the convex quadrilateral formed by uniting the two triangles. This approach can be generalized to higher dimensions, nevertheless, its convergence is proved for $d = 2$ only.

Keeping the above mentioned property in mind, one could add points to DT sequentially, editing just the affected parts of the triangulation. The triangle containing the new point is split into three triangles and then the flip algorithm is applied. Existence of a triangle that includes the new point is ensured by adding a super-triangle to the triangulation at the very beginning (as described in the previous sections). This technique, first introduced in [29], can be extended into higher dimensions as well.

Another incremental method works by adding points one-by-one, each time checking if the DT condition is satisfied. A generalized version of this so-called *Bowyer-Watson algorithm* is implemented in our PSA for computing Minimum-distance designs and described in more detail in Section 4.5. One of its advantages is its incremental nature, which is in accordance with the point-by-point design augmentation in the Greedy procedure of PSA. Additionally, checking DT condition requires constructing the circum-hypersphere of every simplex. If the triangulation is Delaunay, its circumcenter is a Voronoi vertex and Voronoi vertices are of main interest from the point of view of PSA for MDDs.

From non-sequential approaches, there is, for instance, the so-called DeWall algorithm ([12]), which is a generalization of the divide-and-conquer algorithm in $d = 2$ into higher-dimensional spaces. DeWall method divides the point-set into two subsets by a cut plane, computes DT along the plane (called the "wall"), subsequently followed by determining two disjoint DTs on the two sides of the "wall". This technique is known to be one of the fastest; for our purpose, however, it is more effective to choose one of the incremental methods described above.

The problem of constructing the DT on a set of points in $\mathbb{R}^d$ can be transformed into the problem of finding the minimum convex hull of points in $\mathbb{R}^d$: Elevate points in $\mathbb{R}^d$ onto a paraboloid in $\mathbb{R}^{d+1}$, compute the convex hull of the elevated points and project its lower envelope back onto $\mathbb{R}^d$, yielding the DT. This allows all convex hull algorithms to be applied for computing DT.

Last, the so-called sweep-line algorithm (see [27]) can be employed to compute DTs

in dimension $d = 2$. This idea is further developed in [67], where the sweep-hull hybrid algorithm was proposed. Analogously, the so-called plane-sweep algorithm can be employed in dimension $d = 3$. However, in higher dimensions, the intricate geometry makes this approach infeasible.

4.5 Computing PDs and RTs: Generalized Bowyer-Watson incremental algorithm

In contrast to the previous section, we do not provide a list of available methods of computing power diagrams and their duals regular triangulations. Instead, we focus on one technique only and describe it in detail.

Classic Bowyer-Watson algorithm is an incremental method of computing DTs (and implicitly VDs) which we briefly discuss in Section 4.4. It was first published simultaneously, but independently, in [9] and [78]. Bowyer-Watson algorithm can be also adapted for PDs and this, together with its incremental nature, makes it the perfect choice for our PSA implementation.

The generalized Bowyer-Watson algorithm works by adding points one-by-one, each time modifying only the part of the triangulation where augmenting the new point caused violation of the RT condition. This adaptation of the classical Bowyer-Watson method is so straightforward, that we provide it without any references. A pseudocode of this method is given by Algorithm 4.

The effective implementation of Algorithm 4 can be reached by using the simplex-connectivity when locating simplices to be removed, see [9] for details. There are several more ways to accelerate the algorithm, which, however, are not crucial from our point of view. In the case of computing MDDs via PSA, the incremental character of the method already increases its speed rapidly (compared to any non-incremental procedure). Moreover, localizing weighted circumcenters for all simplices of the triangulation, which is required in Step 6, determines the desired power vertices without additional effort.

In Step 2 of Algorithm 4, the issue of the weights assigned to the individual vertices of the super-simplex is not discussed. Naturally, the weights must be chosen small enough

Algorithm 4: Generalized Bowyer-Watson Algorithm

Input : Set of weighted generating points $\mathcal{G} = \{g\}$.

Output: Regular triangulation $\mathcal{T}$.

Set $\mathcal{T} = \emptyset$.
Add super-simplex to $\mathcal{T}$. `// super-simplex contains all generating points`
for $g = (c, w_c) \in \mathcal{G}$ **do**
    Set $badS = \emptyset$. `// set of simplices violating RT condition`
    for $T \in \mathcal{T}$ **do**
        if $pow(v_T, g) < r_T$ **then**
            `//` v_T, r_T `denote the weighted circumcenter and weighted radius of simplex` T`, respectively`
            Add T to $badS$.
        end
    end
    Construct $holeFacets$ - the set of all facets of the polytopal hole $\cup_{T \in badS} T$.
    for $T \in badS$ **do**
        Remove T from $\mathcal{T}$.
    end
    for $\mathcal{F} \in holeFacets$ **do**
        Form the new simplex by connecting $\mathcal{F}$ and c and add it to $\mathcal{T}$.
    end
end
Delete redundant simplices from $\mathcal{T}$. `// simplices containing vertices of the super-simplex`
return $\mathcal{T}$

not to influence the PD($\mathcal{G}$). This formally means that for all $x \in \mathcal{X}$ there exists $g \in \mathcal{G}$ such that $\mathrm{pow}(x, g) \leq \mathrm{pow}(x, \hat{g})$ for every vertex $\hat{g}$ of the super-simplex.

In practise, this can be achieved for instance by choosing the super-simplex such that for each its vertex $\hat{g} = (\hat{c}, w_{\hat{c}})$ it holds that

$$\min_{x \in \mathcal{X}} \rho(x, \hat{c}) \geq \max_{x,y \in \mathcal{X}} \rho(x, y), \tag{3.17}$$

and, simultaneously $w_{\hat{c}} \leq w_c$ for all $g = (c, w_c) \in \mathcal{G}$. In case of $\mathcal{X} = [0, 1]^d$, the right-hand side of (3.17) obviously equals $\sqrt{d}$.

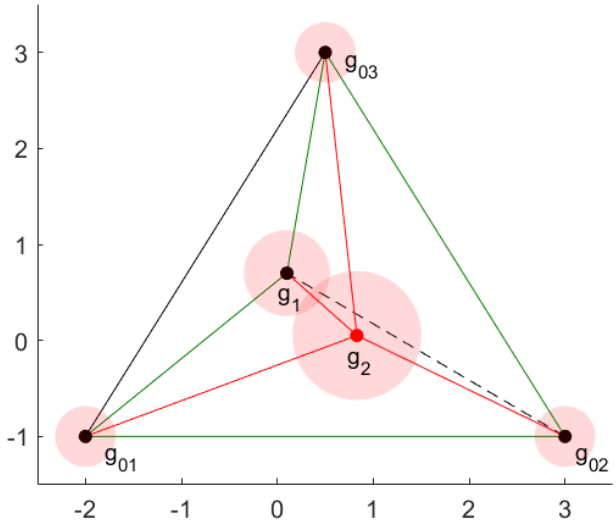


Figure 3.13: One iteration of the Algorithm 4 in dimension $d = 2$. Weighted points g_{01}, g_{02}, g_{03} denote the vertices of the super-simplex, g_1 denotes a weighted point that is already in the triangluation, g_2 is a weighted point to be added. The star-shaped polytopal hole (marked in green) is to be re-triangulated by excluding an old edge (marked as a dashed line) and including new edges (marked in red).

4.5.1 Deletion methods

The exchange character of PSA requires updating the RT not only after a point insertion, but also after a point deletion. Re-triangulating the "polytopal hole" in Figure 3.13 in order to include also point g_2 is easy - g_2 simply forms a new simplex with each of the facets of the polytopal hole (see Step 15 in Algorithm 4). Nevertheless, re-triangulation of the polytopal hole that emerges as a result of a point removal is not that straightforward and can be handled by several methods, some of them listed below.

Since our implementation of PSA was done in Matlab, we decided to utilize the available package called "Power Diagrams" ([44]), which constructs the PD of points in $\mathbb{R}^d$ (and the corresponding RT) by finding the minimum convex hull of points "lifted" to a paraboloid in $\mathbb{R}^{d+1}$, see Section 4.4. Using the package, we simply compute the RT on the vertices of the polytopal hole and subsequently delete the redundant simplices, i.e., simplices that lie inside of the convex hull of the vertices, but outside of the polytopal hole.

Since there was no point added, all Delaunay simplices outside of the polytopal hole keep their empty-hypersphere property and remain Delaunay in the new triangulation. Therefore the polytopal hole can be re-triangulated without any further changes outside of it. Computing DT on the vertices of the hole thus yields a triangulation in which there exists a union of triangles forming the polytopal hole exactly. The remaining triangles not included in the union are then those to be removed.

Alternatively, one can for instance employ the algorithm proposed in [18], which makes use of the duality between DTs in $\mathbb{R}^d$ and convex hulls in $\mathbb{R}^{d+1}$, too. This method is based on the *shelling* algorithm and can be generalized for dimensions higher than $d = 2$, as well as for RTs.

4.6 Implementation of PSA for Minimum-distance designs

This section provides particularities about the implementation of PSA for MDDs. Recall that we still employ the general privacy sets framework 3, which includes also augmentation of a permissible non-maximal design (Step 2 in 1). Let us have a permissible $\xi, |\xi| = N^* < N$ in the design space $\mathcal{X} = [0,1]^d$. In order to augment ξ with remaining $N - N^*$ points, we add points one-by-one, as given by Algorithm 5.

First note that we consider only design spaces of the cubical shape in case of MDDs, which can be without loss of generality scaled into $[0,1]^d$. This assumption is required in Lemma 4.7; computing intersections of a VD and the boundaries of a *constrained* design region would most probably get much more complex.

Another issue is determining relevant design points to be projected in Step 5 of Algorithm 5. If all vertices of the Voronoi cell of some design point $x \in \mathcal{X}$ lie inside of $\mathcal{X}$, the

Algorithm 5: One-point permissible augmentation

Input : A permissible design $\xi, |\xi| = N^* < N$.

Output: A permissible design $\xi, |\xi| = N^* + 1$.

Set $\mathcal{V} = \emptyset$. `// set of permissible power vertices of all faces of` $\mathcal{X}$

Construct $\mathcal{T}$ - RT(ξ), by setting $w_x = 0$ for all $x \in \xi$ and employing Algorithm 4.

Add all permissible $v_T, T \in \mathcal{T}$, into $\mathcal{V}$. `//` v_T `is the weighted circumcenter of` T

for *every face* $\mathcal{F}$ *of* $\mathcal{X}$ **do**

  Project "relevant" $x \in \xi$ onto $\mathcal{F}$ adjusting weights by (3.15). `// see the text for discussion`

  Construct $\mathcal{T}_{\mathcal{F}}$ - RT of the projected points using Algorithm 4.

  Add all permissible $v_T, T \in \mathcal{T}_{\mathcal{F}}$, into $\mathcal{V}$.

end

Sample randomly from $\mathcal{V}$ and perform short permissible random walks starting at the sampled points. Let $\mathcal{V}_s$ denote the resulting set of points.

Set $x^* \in \arg\max_{x \in \mathcal{V}_s} \phi(\xi \cup \{x\})$.

Set $\xi = \xi \cup \{x^*\}$.

return ξ

projection $P_{\mathcal{F}}(x)$ onto any face $\mathcal{F}$ clearly plays no role in the PD computed on $\mathcal{F}$. Thus, when projecting onto a $(d-1)$-dimensional face $\mathcal{F}$, it is enough to project only points with at least one Voronoi vertex lying in the half-space not containing $\mathcal{X}$, with respect to the dividing-hyperplane given by $\mathcal{F}$. This approach is adopted for the lower-dimensional faces recursively in the same fashion. Note that projecting unnecessary points can speed down the computations, but cannot change the resulting PD - their cells will simply be empty.

Choosing a vertex $x \in \mathcal{V}$ is subsequently followed by performing a short random walk in the permissible design area, see Step 9 of Algorithm 5. This random walk can locally improve the design, similarly to the local search procedure of PSA for Bridge designs - as described in Section 3. It is not inevitable, though, the random walk can simply bring additional variability into the procedure. In our implementation, we choose the length of the step randomly and always change just one coordinate at a time.

4.7 Examples

In this section, we provide some examples of Minimum-distance designs found by PSA. In order to be able to evaluate the performance of our algorithm, we compare the results to the designs obtained by some of the competing methods. Unfortunately, due to the complexity of the optimization problem (3.8) for privacy sets given by (3.7), there are not many suitable algorithms available. On the other hand, the lack of appropriate methods is one of the reasons we find our approach contributory to this area of research.

Though the main theoretical results supporting the use of our PSA for MDDs were proved for the two-dimensional space only (Theorem 4.3 and Theorem 4.6), we implemented the method for $\mathbb{R}^d$ based on Conjecture 4.4 and Conjecture 4.8. Our results suggest that the algorithm can be employed in higher dimensions without any problems (although coding of the programme for higher dimensions is technically challenging).

Our aim was to compare to the best and the most accessible algorithms and we present two of them below. The first one - *Grid algorithm* is a simple exchange-type algorithm, that we designed as a straightforward approach one would come up with when dealing with privacy sets (3.7). Its simplicity allows for an easy implementation and makes

the algorithm incomparably fast. The second method - *Resource constraints heuristic* or *RC heuristic* - is based on paper [32] and can handle a broad spectrum of linear restrictions (called resource constraints). However, its capability to deal with large or high-dimensional optimization problems is very limited. Both algorithms were, similarly to PSA, implemented in Matlab computational environment, making the comparisons as fair as possible.

Third possibility would be the use of methods of mathematical programming, namely the so-called mixed integer second-order cone programming (MISOCP) - see [64]. Nonetheless, this would require the use of more advanced specialized software and, because of its incapability to accommodate large-scale problems, there is no reason to expect the results to significantly outperform those of RC heuristic.

Grid algorithm

Let us have a minimum-spacing constant δ and a given number of observations N. Let us construct a grid of L^d points in $\mathcal{X} = [0,1]^d$, such that no two points of the grid collide (i.e., any two points are at least δ apart from each other). Naturally, it is advisable to create the grid as dense as possible, which, in case of equal spacing, means $L = \lfloor 1/\delta \rfloor + 1$. This setting ensures permissibility of any N-point design with points located on the grid.

In the first step, we add design points one-by-one using a forward greedy procedure. In the next step, we try to increase the criterion value by exchanging some of the design points while varying only one coordinate at a time (the so-called coordinate-exchange algorithm, see [46]).

The equally spaced grid is certainly not the only discretization of $\mathcal{X}$ that ensures condition (3.7) to hold. Any arrangement of non-overlapping d-dimensional spheres (called also sphere packing) of radius $\delta/2$ in $\mathcal{X}$ can represent a "permissible" discretization. (To be mathematically correct - if $\mathcal{X} = [0,1]^d$, then we arrange the spheres in $[-\delta/2, 1+\delta/2]^d$). One would naturally search for the densest packing possible, such as so-called close-packed structures for the three-dimensional design space (for example, FCC and HCP lattices, see, e.g., [13]). However, the highest densities are not known in all the dimensions and for our purpose, the whole process of discretizating could get unnecessarily complex.

Resource constraints heuristic

Paper [32] offers a more sophisticated heuristic that can handle Minimum-distance constraints as a special case of the so-called resource constraints. This method works on finite design spaces, viewing a design as a vector $\mathbf{w}$ of integer non-negative values representing numbers of measurements in the points of the design space, i.e., $\sum_{x\in\mathcal{X}} w_x = N$. Resource constraints are linear constraints of form (2.2), where all constants a_{xj} are non-negative, b_j are positive and for all $x \in \mathcal{X}$ there is some $j \in \{1, \ldots, l\}$ such that $a_{xj} > 0$.

In order to formulate (3.7) in the terms of (2.2), let us discretize $[0, 1]^d$ into a design space of $n = L^d$ equally spaced points, where $L \in \mathbb{N}$ is an appropriately chosen parameter. We remark that L does not have to coincide with the parameter L in the Grid algorithm described above. In fact, it is possible to set L to greater values, which allows for more possibilities of arranging a permissible design. On the other hand, the greater L is, the slower the heuristic gets, therefore a certain trade-off must be made.

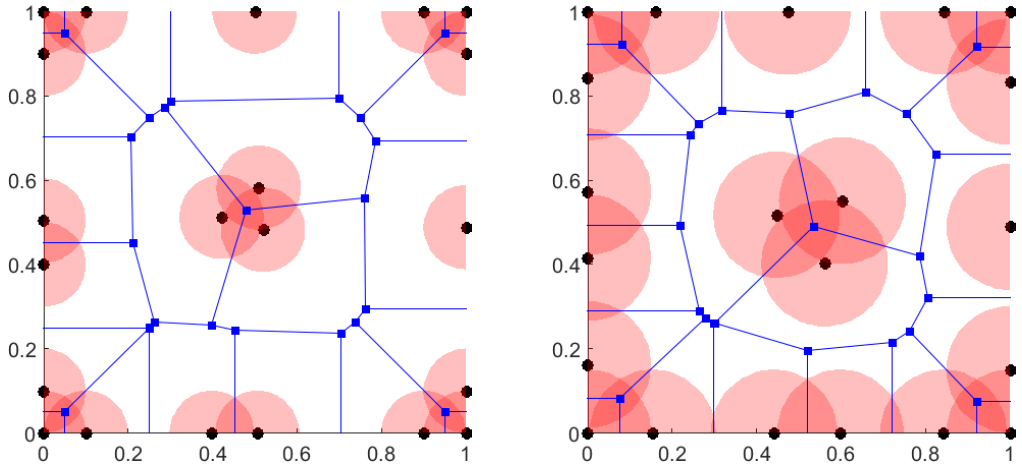
Let $\{\mathcal{X}_j\}_{j=1}^{l}$ be the system of all such subsets of $\mathcal{X}$ that for every $j = 1, \ldots, l$, there exists $c_j \in [0, 1]^d$ such that $\rho(x, c_j) \leq \delta/2$ for all $x \in \mathcal{X}_j$. Consequently, for every $j = 1, \ldots, l$, we get the linear constraint $\sum_{x\in\mathcal{X}_j} w_x \leq 1$, which can be rewritten into $\sum_{x\in\mathcal{X}} I(x \in \mathcal{X}_j)w_x \leq 1$, where $I(.)$ is the indicator function.

4.7.1 Minimum-distance designs for 2 factors in 21 runs

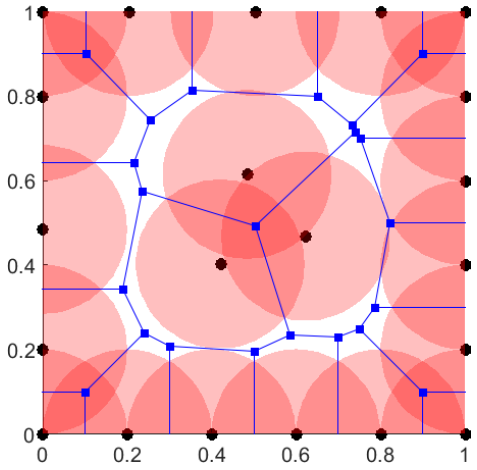
Similarly to 3.1.1, let us have $N = 21$ measurements located in the two-dimensional design space. Let the dependence of y on x be modelled by the full quadratic regression function, that is, $\mathbf{f}(x) = (1, x_1, x_2, x_1^2, x_2^2, x_1x_2)^\top$.

Figure 3.14 shows examples of the resulting designs from Privacy sets algorithm for three various values of the minimum-spacing constant δ. The initial design was constructed by the forward greedy procedure on the grid, the same way as in Grid algorithm. All three computations were completed in less than 60 seconds and the criterion values of the resulting designs were 0.0667, 0.0644 and 0.0630, respectively (the criterion function was the standardized D-optimality function given by (3.4)).

The efficiencies of the designs obtained by Grid algorithm relative to the designs displayed in Fig. 3.14 were 0.9990, 0.9914 and 0.99250, respectively. It should be said



(a) $\delta = 0.1$ (b) $\delta = 0.15$



(c) $\delta = 0.2$

Figure 3.14: Minimum-distance designs resulting from PSA. In all the three cases, there are $N = 21$ points allocated, the statistical model is full quadratic, and the minimum spacing parameter δ takes on values 0.1, 0.15 and 0.2 in Figures 3.14a,3.14b and 3.14c, respectively.

that Grid algorithm computed the designs in a fraction of a second. The efficiencies of the designs found by RC heuristic relative to the designs displayed in Fig. 3.14 were 0.9982, 0.9161 and 0.9301, respectively. RC heuristic was run for 60 seconds, but we can see that more time would be needed in order to obtain comparable results.

4.7.2 Minimum-distance designs for 3 factors in 16 and in 35 runs

In this example, we illustrate the behaviour of our algorithm in the three-dimensional design space $\mathcal{X} = [0,1]^3$ with the regression function given by

$$\mathbf{f}(x) = (1, x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3, x_1^3, x_2^3, x_3^3)^\top.$$

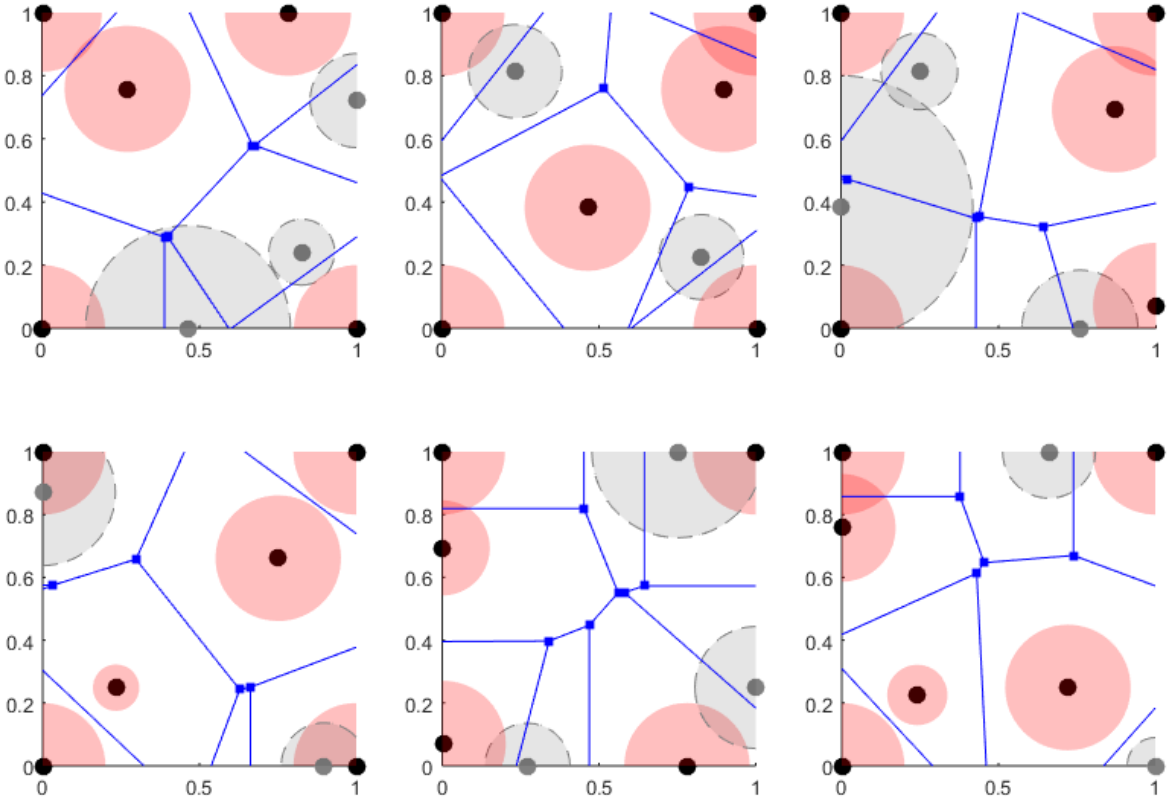


Figure 3.15: Two-dimensional projections of the Minimum-distance design resulting from PSA for $\delta = 0.2$. There are $N = 16$ points located in $[0,1]^3$, but only relevant points are projected onto the faces of the cube. Red circles denote positive weights, while weights of the points associated with grey circles are negative - see the text for further details.

Figure 3.15 displays the two-dimensional projections of the resulting design for $N = 16$ measurements that are located at least $\delta = 0.2$ apart from each other. First thing one would immediately notice is that not all of the 16 design points are displayed in the projections. The aim of this example is to illustrate the necessity of being able to deal with power diagramsin PSA (Voronoi diagrams alone would not suffice). We do not

project the whole design, but only those points which, when projected, have non-empty power cells (see Section 4.6 for determining these points).

The areas of the red circles are proportional to the weights of the corresponding points, i.e., we have $w_c = r_c^2$. On the other hand, points with the grey dashed circles are more distant from the particular face and their weights are proportional to the negative areas of the circles: $w_c = -r_c^2$. Note that these circles are not privacy sets anymore, although they are not completely unrelated - the red circles are exactly the intersections of the three-dimensional privacy sets with the faces of the cube $\mathcal{X} = [0, 1]^3$.

The efficiency of the design resulting from Grid algorithm relative to the design plotted in Figure 3.15 was 0.9748, although there was a significant difference in the speed of the computation in favour of Grid algorithm - similarly to Example 4.7.1.

The design found by PSA was computed in less than 40 seconds. The performance of RC heuristic depends very much on its initial setting - the discretization of the design space. For the best of these settings, the efficiency of the design found by RC heuristic in 40 seconds relative to the design in Figure 3.15 was 1.0422.

In Figure 3.16, we plot two-dimensional projections of the resulting design of PSA for $N = 35$ and $\delta = 0.34$. Onto each face of $\mathcal{X} = [0, 1]^3$, we project only points significant for the power diagram on that particular face, as discussed in Section 4.6. The weights of the projected points are given by $w_c = r_c^2$, where r_c is the radius of the corresponding red circle.

The design displayed in Figure 3.16 was computed in less than 250 seconds. RC heuristic - in its best setting - could find in 250 seconds a design with efficiency 0.9988 (relative to the design in Figure 3.16). Grid algorithm, however, was unable to provide any solution at all: The relatively big spacing constant δ allows only for a coarse grid, which cannot accommodate 35 design points. For a larger N, it may also happen that none of these three algorithms will come up with a solution (a permissible design), see more detailed comparisons in the numerical studies below.

We would like to emphasize the illustrative character of this example: We did not choose it as a proper representative of the mutual efficiencies. Its aim was rather to

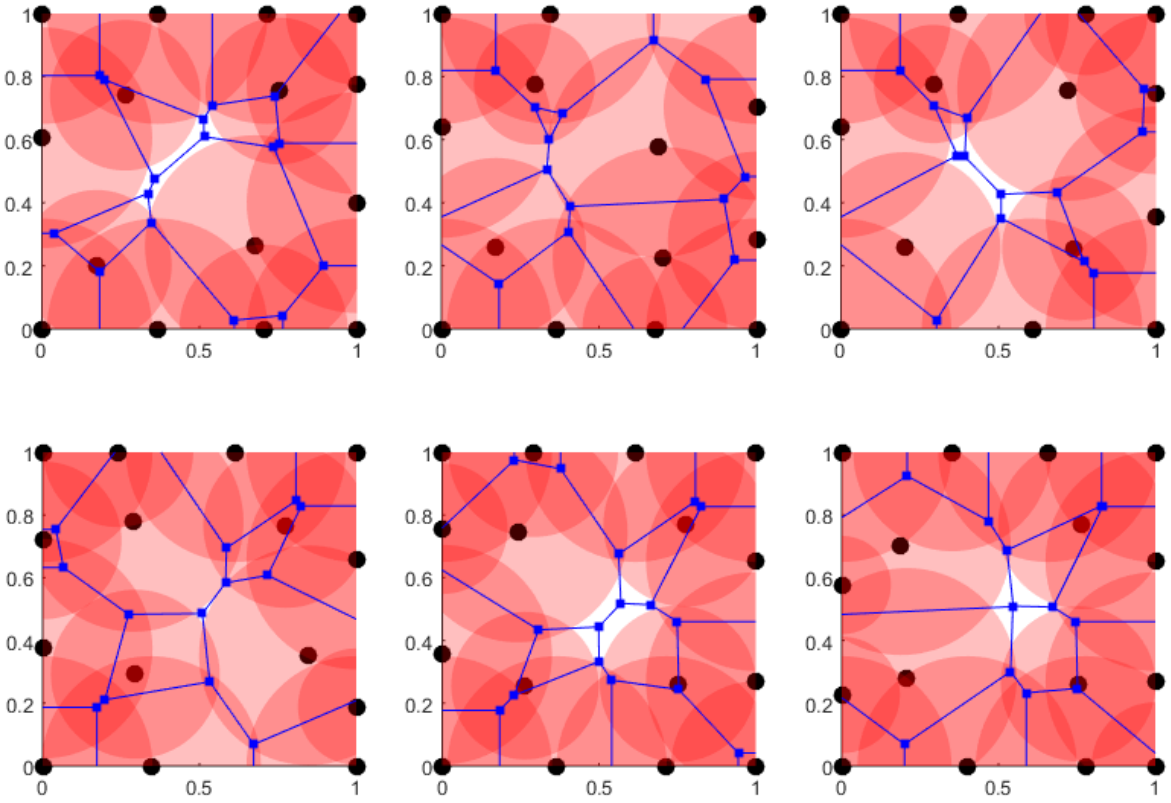


Figure 3.16: Two-dimensional projections of the Minimum-distance design resulting from PSA for $\delta = 0.34$. There are $N = 35$ points located in $[0, 1]^3$, but only relevant points are projected onto the faces of the cube. Red circles denote weights of the design points - see the text for further details.

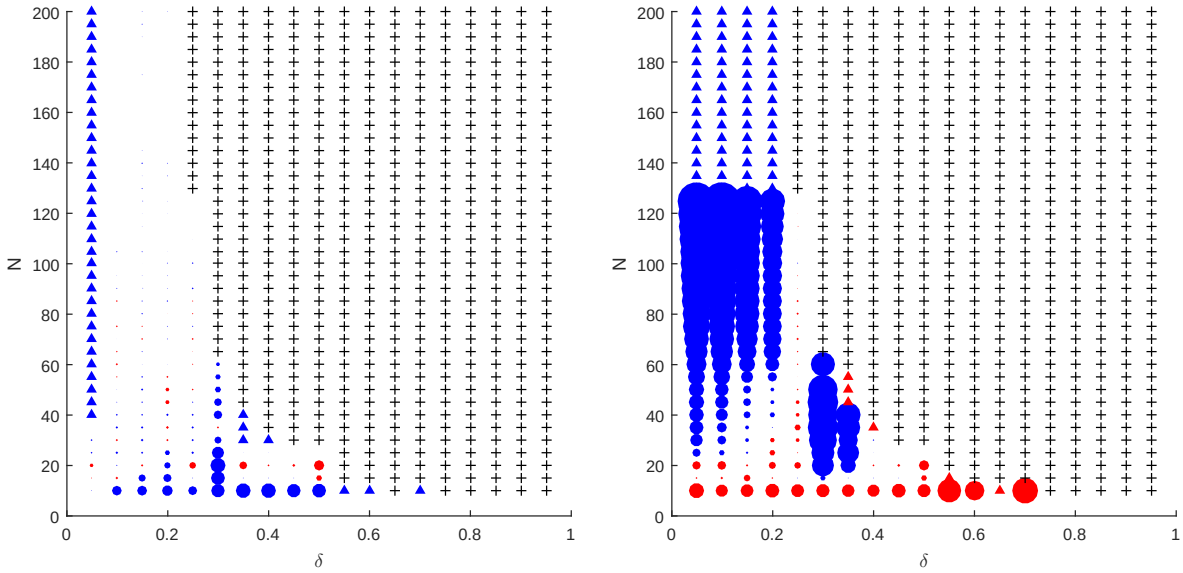
elucidate the mechanism of PSA on MDDs and clarify various situations that may arise on the boundaries of the cuboid design space.

4.7.3 Numerical study: 3 factors

We conclude this chapter with numerical studies on Minimum-distance designs. Our goal was to review the qualities of the three competing methods (PSA, Grid algorithm, RC heuristic) in various situations that might occur in practice. We would like to underline the difficulty of making such a comparison fair and unbiased - this is mainly due to the great difference between the methods.

The basic idea was to let the algorithms find the best design in a given time period

for given values of δ and N. For all the combinations, we considered the full quadratic regression model and the criterion of D-optimality (3.4).



(a) PSA vs. Grid algorithm

(b) PSA vs. RC heuristic

Figure 3.17: Pairwise comparisons of PSA, Grid algorithm and RC heuristic for various values of δ and N for 3 factors. The color of a circle denotes which algorithm had performed better in the given time of 60 seconds - the blue color stands for PSA, the red for Grid algorithm, event. RC heuristic. The size of a circle designates how well the "winning" algorithm had performed relative to the "losing" one - see the text for more details. The blue triangle was drawn if PSA was the only algorithm of the two to find a solution, the red triangle in the opposite case. The black plus sign signalizes that no permissible design was found by any of the two considered methods. Note that for most of the black plus signs, a permissible design simply does not exist.

The results of the studies are presented in Figure 3.17. The comparisons were done in pairs, that is, Figure 3.17a confronts the resulting values of PSA versus Grid algorithm, while Figure 3.17b depicts the values of PSA in comparison to RC heuristic. For a pair of δ (x-axis) and N (y-axis), each of the three algorithms had 60 seconds to search for a design with the highest criterial value.

In both figures, a blue circle designates that the criterial value of the design found by PSA was greater than the value of the design found by the competing method (Grid algorithm on the left, RC heuristic on the right). The radius of the particular blue circle is

proportional to the difference $1 - \text{eff}_D(\xi_{\text{GRID}}|\xi_{\text{PSA}})$ (eventually $1 - \text{eff}_D(\xi_{\text{RC}}|\xi_{\text{PSA}})$), where ξ_{PSA}, ξ_{GRID} and ξ_{RC} denote the best designs found by PSA, Grid algorithm and RC heuristic, respectively. A red circle denotes the same for the corresponding competing algorithm. The scales of the circles are kept the same in both figures.

Let us now discuss Figure 3.17a in more detail. First, there is a difference in the total computational time of PSA and Grid algorithm, although there are 60 seconds allotted to each of them. While PSA works slowly and steadily on improving the initial design, Grid algorithm (with the initial design pre-computed) is due to its simplicity finished in a few seconds or even less. However, constructing an initial design can be computationally demanding and sometimes can require more than 60 seconds. For example, in the figure there is a column of the blue triangles for most of the computations with $\delta = 0.05$. On the other hand, PSA can start from the same initial design as Grid algorithm (and when possible, it does), but when required can come up with its own initial design based on constructing Voronoi diagrams.

RC heuristic, studied in Figure 3.17b, has some specifics. Most importantly, it views designs as arrays of weights and works on a finite design space. The key point in the effective use of this technique is then the discretization of $\mathcal{X}$ into L^d grid points, which is inevitably the result of a trade-off: The large L slows down the computations, while the small L significantly restricts diversity of permissible solutions. Our setting in this study was $L = 5$, which is rather small but brought in general the best results, since the greater values were making the computations extremely slow. In the ideal case, one would best try to find the most suitable value of L for every combination of δ and N separately, but this is obviously only hypothetical and inapplicable in practice.

Clearly, the restriction of $n = 5^3 = 125$ points of the design space makes the calculations with more than 125 design points infeasible - see the blue plus signs in the figure. Also note that the relatively coarse grid makes RC heuristic not very suitable for small values of δ - PSA can find a better solution by enabling the design points to be closer to each other. On the other hand, RC heuristic performs better for greater δ and it seems it can come up with a solution in some cases when both other competing algorithms fail. Last, we point out the decreasing "performance" of RC heuristic with the increasing number of measurements N. Naturally, both algorithms tend to do worse as N increases,

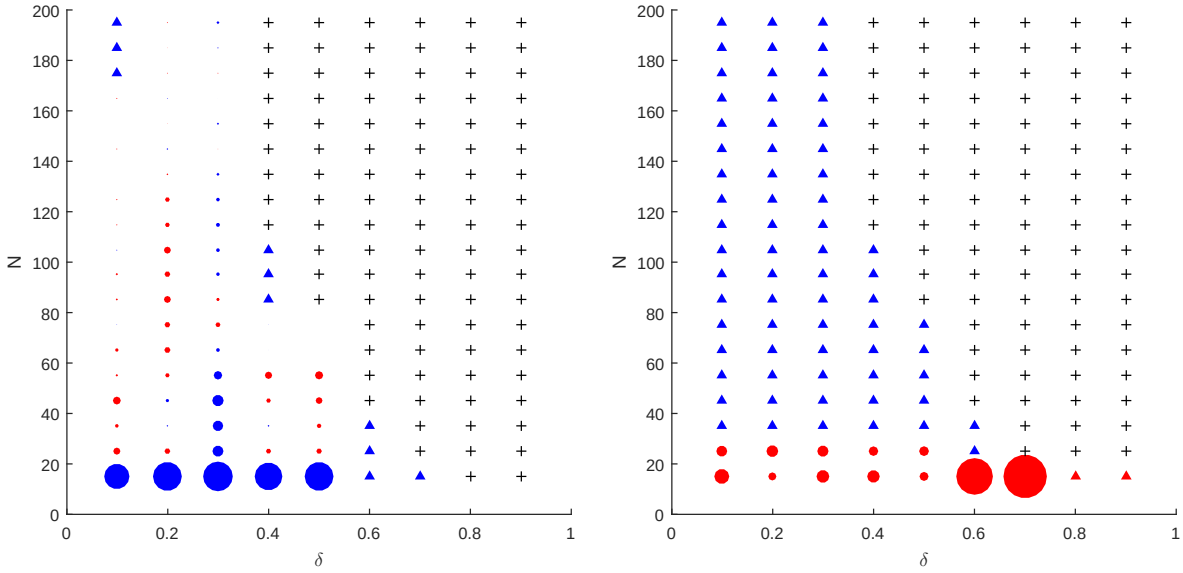
simply because of the lack of the computational time. It seems, however, that for RC heuristic this decrease in the criterial value is more rapid than for PSA.

4.7.4 Numerical study: 4 factors

For the four-dimensional design space, we have performed an analogous study. The model was full quadratic, the criterion D-optimality and the total computational time was increased to 600 seconds (because of the greater size of the problem). We have also reduced the total number of combinations of δ and N, in order to be able to complete the study in a reasonable time.

The results presented in Figure 3.18 show similarities to the three-dimensional examples in Figure 3.17. In Figure 3.18a, Grid algorithm performs well in many cases, but there is still a significant number of combinations of δ and N when it finds no solution at all (in contrast to PSA).

In the implementation of RC heuristic, the best setting turned out to be $L = 3$, which yields $n = 3^4 = 81$ points of the design space. In spite of this being a ridiculously coarse design space grid, RC heuristic has become too slow to compute (in a given time slot of 600 seconds) a permissible design even for many N smaller than 81, see Figure 3.18b.



(a) PSA vs. Grid algorithm

(b) PSA vs. RC heuristic

Figure 3.18: Pairwise comparisons of PSA, Grid algorithm and RC heuristic for various values of δ and N for 4 factors. The color of a circle denotes which algorithm had performed better in the given time of 600 seconds - the blue color stands for PSA, the red for Grid algorithm, event. RC heuristic. The size of a circle designates how well the "winning" algorithm had performed relative to the "losing" one - see the text for more details. The blue triangle was drawn if PSA was the only algorithm of the two to find a solution, the red triangle in the opposite case. The black plus sign signalizes that no permissible design was found by any of the two considered methods. Note that for most of the black plus signs, a permissible design simply does not exist.

Chapter IV

Results, conclusions and outlook

We conclude the thesis with a brief enumeration of the most important results. We also provide a few suggestions for possible future extensions of the ideas presented in this work.

1 Main results

Size- and cost-constrained designs

- We have formulated the optimization problem of size- and cost-constrained designs in its natural form in (2.13). In Proposition 2.1 in Chapter II, we have proved that it is enough to deal with the "equality" problem (2.16) instead.

- For size- and cost-constrained designs, we have proposed an equivalence theorem - Theorem 3.1 in Chapter II, which provides conditions equivalent to the D-optimal size- and cost-constrained designs.

- Theorem 3.2 in Chapter II allows us to compute a lower bound on the efficiency of any permissible size- and cost-constrained design and can be adjusted to "delete" non-supporting points at any stage of the algorithm.

- We have proposed the barycentric multiplicative algorithm for computing D-optimal size- and cost-constrained designs (the so-called S&C algorithm), based on the method introduced in [31]. Moreover, Theorem 4.2 in Chapter II proves its con-

vergence under a mild technical condition (in the sense that the criterial values converge to the optimum).

- We have studied the behaviour of the proposed algorithm under various parameter settings and compared its performance to other selected competing methods. The results presented in Figure 2.6 justify the relevance of S&C algorithm.

Privacy sets

- We have introduced the concept of privacy sets, which represent linear restrictions on the experimental design, enforcing its space-filling properties. This approach is in accordance with the so-called "hard" space-filling methods (in contrast to the more established "soft" space-filling techniques).

- We have proposed a general exchange-type framework for computing designs under space-filling constraints - Privacy Sets Algorithm (PSA) described in Section 2.1 of Chapter III.

- For Bridge designs, we have concretized PSA into an efficient procedure, which significantly outperforms even the state-of-the-art method (see, e.g., Figure 3.5).

- For Minimum-distance designs, we have proposed a specific version of PSA based on the computing of the Voronoi vertices and the power vertices. The main theoretical results are given by Theorems 4.3 and 4.6 in Chapter III, which explain the relation between the multi-dimensional Voronoi and power diagrams and some of the crucial steps of PSA.

- We have demonstrated the performance of PSA specified for both Bridge and Minimum-distance designs relative to the competing techniques in several examples and numerical studies (Section 3.1 and Section 4.7 of Chapter III).

2 Future research

Size- and cost-constrained designs

The approximate D-optimal size- and cost-constrained designs have a tendency to be sparse; that is, they have small support even if the actual size of the problem is large. We

use this property by employing the "deletion" rules, but the sparsity of designs can also be utilized in several other ways.

For instance, for the standard D-optimal design problem, the paper [86] uses a combination of vertex-direction, multiplicative, and vertex-exchange methods (see [8]). In this hybrid algorithm, the vertex-exchange method maintains the support of the solutions to be small, which is to a large extent responsible for the very good efficiency of the algorithm, as a whole.

Another possibility is to use a simplicial decomposition algorithm, which was adapted for computing D-optimal box-constrained designs in [75]. This algorithm is based on alternately solving a linear programming subproblem and a non-linear restricted master problem that finds the maximum of the objective function over the convex hull of a typically small set of permissible points.

With some effort, these approaches could be adapted for solving the size- and cost-constrained problem (2.16).

Privacy sets

The generality of PSA leaves enough space for the future research in many directions. So far, we have specified this technique for Bridge designs and Minimum distance designs, but any other type of constraints that can be viewed from the privacy-sets perspective can be considered as well.

PSA for Minimum-distance designs can be quite straightforwardly adapted to deal with weighted Minimum-distance designs with privacy sets of in general unequal radii given by a function $\delta : \mathcal{X} \to \mathbb{R}_0^+$ (see Definition 4.5 in Chapter III). It would probably require more effort to adjust the algorithm to handle ellipsoid privacy sets (or, equivalently, cuboidal design spaces with in general unequal edge-lengths).

One of the theoretical questions that can arise from Section 4 is identifying the maximal number $N_{\max}$ such that, for a given δ and for every $N \leq N_{\max}$, Assumption 2.1 in Chapter III is satisfied.

Bibliography

[1] Anthony Atkinson, Alexander Donev, and Randall Tobias. *Optimum Experimental Designs, with SAS*. 01 2007.

[2] Franz Aurenhammer. Power diagrams: properties, algorithms and applications. *SIAM Journal on Computing*, 16(1):78–96, 1987.

[3] Franz Aurenhammer. Voronoi diagrams — a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[4] Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi diagrams and Delaunay triangulations*. World Scientific Publishing Company, 2013.

[5] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. MPS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, 2001.

[6] Eva Benková, Radoslav Harman, and Werner G. Müller. Privacy sets for constrained space-filling. *Journal of Statistical Planning and Inference*, 171:1 – 9, 2016.

[7] Dankmar Böhing. On the construction of optimal experimental designs: a penalty approach. *Statistics: A Journal of Theoretical and Applied Statistics*, 12(4):487–495, 1981.

[8] Dankmar Böhning. A vertex-exchange-method in D-optimal design theory. *Metrika*, 33(1):337–347, 1986.

[9] Adrian Bowyer. Computing dirichlet tessellations. *The computer journal*, 24(2):162–166, 1981.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[11] George Casella. *Statistical Design*. Springer New York, New York, 2008.

[12] Paolo Cignoni, Claudio Montani, and Roberto Scopigno. DeWall: A fast divide and conquer Delaunay triangulation algorithm in Ed. *Computer-Aided Design*, 30(5):333–341, 1998.

[13] John H. Conway and Neil J.A. Sloane. *Sphere Packings, Lattices and Groups*, volume 290. 01 1988.

[14] Dennis Cook and Valerii Fedorov. Constrained optimization of experimental design. *Statistics*, 26(2):129–148, 1995.

[15] R. Dennis Cook and Lawrence. A. Thibodeau. Marginally restricted D-optimal designs. *Journal of the American Statistical Association*, 75(370):366–371, 1980.

[16] R. Dennis Cook and Weng Kee Wong. On the equivalence of constrained and compound optimal designs. *Journal of the American Statistical Association*, 89(426):687–692, 1994.

[17] Holger Dette, Andrey Pepelyshev, and Anatoly Zhigljavsky. Improving updating rules in multiplicative algorithms for computing D-optimal designs. *Computational Statistics & Data Analysis*, 53(2):312–320, 2008.

[18] Olivier Devillers. On deletion in Delaunay triangulations. *International Journal of Computational Geometry & Applications*, 12(03):193–205, 2002.

[19] Vladimir Dragalin and Valerii Fedorov. Adaptive designs for dose-finding based on efficacy–toxicity response. *Journal of Statistical Planning and Inference*, 136(6):1800–1823, 2006.

[20] Vladimir Dragalin, Valerii Fedorov, and Yuehui Wu. Adaptive designs for selecting drug combinations based on efficacy–toxicity response. *Journal of Statistical Planning and Inference*, 138(2):352–373, 2008.

[21] Danel Draguljic, Angela M. Dean, and Thomas J. Santner. Noncollapsing space-filling designs for bounded nonrectangular regions. *Technometrics*, 54(2):169–178, 2012.

[22] Gustav Elfving. Optimum allocation in linear regression theory. *The Annals of Mathematical Statistics*, 23(2):255–262, 1952.

[23] Valerii Fedorov and Peter Hackl. *Model-Oriented Design of Experiments*, volume 41. 01 1997.

[24] Valerii V. Fedorov. *Theory of optimal experiments.* Probability and mathematical statistics. Academic Press, 1972.

[25] Valerii V. Fedorov. Optimal design with bounded density: Optimization algorithms of the exchange type. *Journal of Statistical Planning and Inference*, 22(1):1–13, 1989.

[26] Ronald A. Fisher. *The design of experiments. 1935.* Oliver and Boyd, Edinburgh, 1935.

[27] Steven Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2(1-4):153, 1987.

[28] Peter Goos and Bradley Jones. *Optimal design of experiments: a case study approach.* John Wiley & Sons, 2011.

[29] Leonidas J. Guibas, Donald E. Knuth, and Micha Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7(1-6):381–413, 1992.

[30] Linda M. Haines. The application of the annealing algorithm to the construction of exact optimal designs for linear–regression models. *Technometrics*, 29(4):439–447, 1987.

[31] Radoslav Harman. Multiplicative methods for computing D-optimal stratified designs of experiments. *Journal of Statistical Planning and Inference*, 146:82–94, 03 2014.

[32] Radoslav Harman, Alena Bachratá, and Lenka Filová. Construction of efficient experimental designs under multiple resource constraints. *Applied Stochastic Models in Business and Industry*, 32(1):3–17, 2016.

[33] Radoslav Harman and Eva Benková. Barycentric algorithm for computing D-optimal size- and cost-constrained designs of experiments. *Metrika*, 80(2):201–225, 2017.

[34] Radoslav Harman and Lenka Filová. Computing efficient exact designs of experiments using integer quadratic programming. *Computational Statistics & Data Analysis*, 71:1159–1167, 2014.

[35] Radoslav Harman and Luc Pronzato. Improvements on removing nonoptimal support points in D-optimum design algorithms. *Statistics & probability letters*, 77(1):90–94, 2007.

[36] Radoslav Harman and Mária Trnovská. Approximate D-optimal designs of experiments on the convex hull of a finite set of information matrices. *Mathematica Slovaca*, 59(6):693–704, 2009.

[37] Ferran Hurtado, Marc Noy, and Jorge Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, 22(3):333–346, 1999.

[38] Mark E. Johnson, Leslie M. Moore, and Donald Ylvisaker. Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, 26(2):131 – 148, 1990.

[39] Bradley Jones, Rachel T. Silvestrini, Douglas C. Montgomery, and David M. Steinberg. Bridge designs for modeling systems with low noise. *Technometrics*, 57(2):155–163, 2014.

[40] V. Roshan Joseph, Evren Gul, and Shan Ba. Maximum projection designs for computer experiments. *Biometrika*, 102(2):371–380, 2015.

[41] Jack Kiefer. The role of symmetry and approximation in exact design optimality. In *Statistical decision theory and related topics*, pages 109–118. Elsevier, 1971.

[42] Jesús López-Fidalgo and Sandra A. Garcet-Rodríguez. Optimal experimental designs when some independent variables are not subject to control. *Journal of the American Statistical Association*, 99(468):1190–1199, 2004.

[43] Saumen Mandal, Ben Torsney, and Keumhee C. Carriere. Constructing optimal designs with constraints. *Journal of Statistical Planning and Inference*, 128(2):609 – 621, 2005.

[44] Frederick McCollum. Power diagrams. `https://www.mathworks.com/matlabcentral/fileexchange/44385-power-diagrams`. MATLAB Central File Exchange. Retrieved November, 2018.

[45] Michael D. McKay, Richard J. Beckman, and William J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[46] Ruth K. Meyer and Christopher J. Nachtsheim. The coordinate-exchange algorithm for constructing exact optimal experimental designs. *Technometrics*, 37(1):60–69, 1995.

[47] Jaroslava Mikulecká. On a hybrid experimental design. *Kybernetika*, 19(1):1–14, 1983.

[48] Toby J. Mitchell. An algorithm for the construction of "D-optimal" experimental designs. *Technometrics*, 16(2):203–210, 1974.

[49] Max D. Morris and Toby J. Mitchell. Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference*, 43(3):381–402, 1995.

[50] Werner G. Müller, Radoslav Harman, and Eva Benková. Discussion of "Space-filling designs for computer experiments: A review". *Quality Engineering*, 28(1):36–38, 2016.

[51] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. *Spatial tessellations: concepts and applications of Voronoi diagrams*, volume 501. John Wiley & Sons, 2009.

[52] Youjin Park, Douglas C. Montgomery, John W. Fowler, and Connie M. Borror. Cost-constrained G-efficient response surface designs for cuboidal regions. *Quality and Reliability Engineering International*, 22(2):121–139, 2006.

[53] Andrej Pázman. *Foundations of Optimum Experimental Design (Mathematics and its Applications)*. Reidel Publ. Comp., Dodrecht, 1986.

[54] Andrej Pázman and Vladimír Lacko. Prednášky z regresných modelov. *Univerzita Komenského v Bratislave, Bratislava*, 2012.

[55] Matthieu Petelet, Bertrand Iooss, Olivier Asserin, and Alexandre Loredo. Latin hypercube sampling with inequality constraints. *AStA Advances in Statistical Analysis*, 94(4):325–339, 2010.

[56] R. L. Plackett and J. P. Burman. The Design of Optimum Multifactorial Experiments. *Biometrika*, 33(4):305–325, 06 1946.

[57] Luc Pronzato. Penalized optimal designs for dose-finding. *Journal of Statistical Planning and Inference*, 140(1):283–296, 2010.

[58] Luc Pronzato. Minimax and maximin space-filling designs: some properties and methods for construction. *Journal de la Société Française de Statistique*, 158(1):7–36, 2017.

[59] Luc Pronzato and Werner G. Müller. Design of computer experiments: space filling and beyond. *Statistics and Computing*, 22(3):681–701, May 2012.

[60] Luc Pronzato and Andrej Pázman. Design of experiments in nonlinear models. asymptotic normality, optimality criteria and small-sample properties, 2013.

[61] Friedrich Pukelsheim. *Optimal Design of Experiments*. Society for Industrial and Applied Mathematics, 2006.

[62] Friedrich Pukelsheim and Sabine Rieder. Efficient rounding of approximate designs. *Biometrika*, 79(4):763–770, 1992.

[63] Ewaryst Rafajłowicz. Minimum cost experiment design with a prescribed information matrix. *Theory of Probability and its Applications*, 34(2):412–416, 1989.

[64] Guillaume Sagnol and Radoslav Harman. Computing exact D-optimal designs by mixed integer second-order cone programming. *The Annals of Statistics*, 43(5):2198–2224, 2015.

[65] Guillaume Sagnol and Radoslav Harman. Computing exact D-optimal designs by mixed integer second-order cone programming. *Ann. Statist.*, 43(5):2198–2224, 2015.

[66] Samuel D. Silvey, Donald M. Titterington, and Ben Torsney. An algorithm for optimal designs on a design space. *Communications in Statistics - Theory and Methods*, 7(14):1379–1389, 1978.

[67] David Sinclair. S-hull: a fast radial sweep-hull routine for Delaunay triangulation. *arXiv preprint arXiv:1604.01428*, 2016.

[68] Kirstine Smith. On the standard deviations of adjusted and interpolated values of an observed polynomial function and its constants and the guidance they give towards a proper choice of the distribution of observations. *Biometrika*, 12(1/2):1–85, 1918.

[69] Lieven Tack and Martina Vandebroek. Budget constrained run orders in optimum design. *Journal of statistical planning and inference*, 124(1):231–249, 2004.

[70] Ben Torsney. A moment inequality and monotonicity of an algorithm. In Anthony V. Fiacco and Kenneth O. Kortanek, editors, *Semi-Infinite Programming and Applications*, pages 249–260, Berlin, Heidelberg, 1983. Springer Berlin Heidelberg.

[71] Ben Torsney and Saumen Mandal. Construction of constrained optimal designs. In *Optimum Design 2000*, pages 141–152. Springer, 2001.

[72] Ben Torsney and Saumen Mandal. Two classes of multiplicative algorithms for constructing optimizing distributions. *Computational statistics & data analysis*, 51(3):1591–1601, 2006.

[73] Ben Torsney and Raúl Martín-Martín. Multiplicative algorithms for computing optimum designs. *Journal of Statistical Planning and Inference*, 139(12):3947–3961, 2009.

[74] Dariusz Uciński. *Optimal measurement methods for distributed parameter system identification*. CRC Press, 2004.

[75] Dariusz Uciński and Maciej Patan. D-optimal design of a monitoring network for parameter estimation of distributed systems. *Journal of Global Optimization*, 39(2):291–322, 2007.

[76] Martijn van Manen and Dirk Siersma. Power diagrams and their applications. *arXiv preprint math/0508037*, 2005.

[77] Lieven Vandenberghe, Stephen Boyd, and Shao-Po Wu. Determinant maximization with linear matrix inequality constraints. *SIAM Journal on Matrix Analysis and Applications*, 19(2):499–533, 1998.

[78] David F. Watson. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *The computer journal*, 24(2):167–172, 1981.

[79] William J. Welch. Branch-and-bound search for experimental designs based on D optimality and other criteria. *Technometrics*, 24(1):41–48, 1982.

[80] William J. Welch. Computer-aided design of experiments for response estimation. *Technometrics*, 26(3):217–224, 1984.

[81] Natalie Wolchover. A bird's-eye view of nature's hidden order. *Quanta Magazine*, 2016.

[82] Stephen E. Wright, Belle M. Sigal, and A. John Bailer. Workweek optimization of experimental designs: exact designs for variable sampling costs. *Journal of Agricultural, Biological and Environmental Statistics*, 15(4):491–509, 2010.

[83] Chien-Fu Wu and Henry P. Wynn. The convergence of general step-length algorithms for regular optimum design criteria. *The Annals of Statistics*, 6(6):1273 – 1285, 1978.

[84] Henry P. Wynn. The sequential generation of D-optimum experimental designs. *The Annals of Mathematical Statistics*, 41(5):1655 – 1664, 1970.

[85] Yaming Yu. Monotonic convergence of a general algorithm for computing optimal designs. *The Annals of Statistics*, 38(3):1593–1606, 2010.

[86] Yaming Yu. D-optimal designs via a cocktail algorithm. *Statistics and Computing*, 21(4):475–481, 2011.

[87] Maryam Zolghadr and Sergei Zuyev. Optimal design of dilution experiments under volume constraints. *Journal of Agricultural, Biological and Environmental Statistics*, 21(4):663–683, 2016.