DEPARTMENT OF ENGINEERING MANAGEMENT

D- and I-Optimal Design of Mixture Experiments in the Presence of Ingredient Availability Constraints

Utami Syafitri, Bagus Sartono & Peter Goos

DEPARTMENT OF ENGINEERING MANAGEMENT

D- and I-Optimal Design of Mixture Experiments in the Presence of Ingredient Availability Constraints

Utami Syafitri, Bagus Sartono & Peter Goos

RESEARCH PAPER 2015-003
JANUARY 2015

D/2015/1169/003

D- and I-Optimal Design of Mixture Experiments in the Presence of Ingredient Availability Constraints

Utami Syafitri[1,2], Bagus Sartono[2] and Peter Goos[1,3]

[1]Faculty of Applied Economics & StatUa Center for Statistics, Universiteit Antwerpen, Belgium
[2]Department of Statistics, Bogor Agricultural University, Indonesia
[3]Faculty of Bioscience Engineering, Katholieke Universiteit Leuven, Belgium

January 21, 2015

Abstract

Mixture experiments usually involve various constraints on the proportions of the ingredients of the mixture under study. In this paper, inspired by the fact that the available stock of certain ingredients is often limited, we focus on a new type of constraint, which we refer to as an ingredient availability constraint. This type of constraint substantially complicates the search for optimal designs for mixture experiments. One difficulty, for instance, is that the optimal number of experimental runs is not known a priori. We show that the optimal design problem in the presence of ingredient availabililty constraints is a nonlinear multidimensional knapsack problem and propose a variable neighborhood descent algorithm to identify D- and I-optimal designs for mixture experiments in case there is a limited stock of certain ingredients.

Keywords: mixture experiment, nonlinear multidimensional knapsack problem, D-optimality, I-optimality, update formulas, variable neighborhood descent algorithm, V-optimality.

1 Introduction

A mixture experiment is an experiment in which the experimental factors are ingredients of a mixture, and the response depends only on the relative proportions of the ingredients. The levels of the experimental factors therefore all lie between zero and one, and the sum of the levels of all experimental factors is one. These constraints define a simplex-shaped experimental region: in the absence of any other constraints, a mixture experiment involving q ingredients has a $(q-1)$-dimensional simplex as its experimental region. In case there are three ingredients, the experimental region is an equilateral triangle. When there are four ingredients, it is a regular tetrahedron. The textbooks on the design of mixture experiments by Cornell (2002) and Smith (2005) pay much attention to experimental designs which were proposed by Scheffé (1958, 1963) specifically for simplex-shaped experimental regions.

Commonly, at least some of the ingredients in a mixture experiment have lower and/or upper bounds on their proportions. Moreover, there may be multicomponent constraints that impose bounds on linear or nonlinear combinations of the ingredient proportions (see, e.g., Cornell, 2002, Smith, 2005, Atkinson, Donev and Tobias, 2007). In that case, it is necessary to compute mixture experimental designs that are optimal given all constraints on the factor levels. A coordinate-exchange algorithm for the construction of D-optimal mixture designs was proposed by Piepel, Cooley and Jones (2005).

A key feature of all published work on the design of mixture experiments is that it implicitly assumes that the stock of each ingredient is large enough to run every experimental design under consideration. However, in some situations, only a limited stock is available for some of the ingredients. This may render classical designs or optimal designs infeasible. De Ketelaere, Goos and Brijs (2011) discuss a mixture experiment involving flours, and mention that a follow-up experiment was considered but that the remaining stock of certain types of flours was limited and could not be replenished.

Therefore, in this paper, we study the impact of availability constraints on D- and I-optimal designs for mixture experiments. We study D-optimal mixture designs because these have received the largest share of attention in the literature (see, e.g., Kiefer, 1961, Uranisi, 1964, Galil and Kiefer, 1977, Mikaeili, 1989, 1993). We also study I-optimal mixture designs because selecting mixture designs based on the prediction variance was already suggested by Scheffé (1958), and because Hardin and Sloane (1993), Goos and Jones (2011) and Jones and Goos (2012) demonstrated that I-optimal response surface designs usually perform well in terms of D-optimality, whereas D-optimal designs usually do not perform well in terms of I-optimality. On the other hand, few researchers have taken up the challenge to seek I-optimal mixture designs. Exceptions are Lambrakis (1968a,b), Laake (1975), Liu and Neudecker (1995), and Goos and Syafitri (2014) who presented a limited number of results, and Goos, Jones and Syafitri (2013), who provide a literature review on D- and I-optimal mixture designs and propose new I-optimal designs for simplex-shaped experimental regions. None of the published work on D- or I-optimal designs deals with ingredient availability constraints.

A remarkable feature of the problem of designing mixture experiments in the presence of availability constraints is that the optimal number of runs is not necessarily known beforehand. Imagine a mixture experiment involving several flours, where only 1 kg is available of a certain ingredient. In that case, it is possible to perform just one experimental run that utilizes all of the ingredient. However, it is also possible to perform two experimental runs using 0.5 kg each, or one run using 0.2 kg, another run using 0.3 kg and a final run using 0.5 kg. As a result, finding an optimal experimental design in the presence of availability constraints requires determining the optimal number of runs and determining the ingredient proportions at each of the runs.

In this paper, we present a variable neighborhood descent algorithm for finding optimal mixture designs in the presence of availability constraints. Our algorithm exploits the similarity between our optimal mixture design problem and a well-known combinatorial optimization problem in the field of operations research, namely the knapsack problem. More specifically, we consider the optimal mixture design problem to be a nonlinear multi-dimensional knapsack problem.

This paper is organized as follows. In Section 2, we describe the most commonly used models for data from mixture experiments and the D- and I-optimality criteria for selecting designs for mixture experiments. Moreover, we briefly introduce pseudocomponents, simplex-lattice designs and the fraction of design space plot, which will be used throughout the paper. In Section 3, we discuss the similarity between the problem of optimally designing mixture experiments in the presence of availability constraints and the knapsack problem. In Section 4, we propose a variable neighborhood descent (VND) algorithm for the D-optimal and the I-optimal design of mixture experiments involving availability constraints. In Section 5, we describe the working of the algorithm in detail. In Section 6, we present D- and I-optimal designs for a variety of scenarios involving ingredient availability constraints. Finally, in Section 7, we end the paper with a conclusion.

2 Models and designs for mixture experiments

2.1 Models for data from a mixture experiment

The most commonly used models for data from mixture experiments are the Scheffé models. If we denote the response measured by y, the number of ingredients in the experiment by q and the proportions of each of the q ingredients by $x_1, x_2, \ldots, x_q$, then the first-order Scheffé model is given by

$$Y = \sum_{i=1}^{q} \beta_i x_i + \epsilon, \tag{1}$$

while the second-order Scheffé model is given by

$$Y = \sum_{i=1}^{q} \beta_i x_i + \sum_{i=1}^{q-1} \sum_{j=i+1}^{q} \beta_{ij} x_i x_j + \epsilon. \tag{2}$$

In these models, the coefficients β_i represent the expected response when the proportion of the ith ingredient equals 1.0, the parameters β_{ij} represent the nonlinear blending properties of the ith ingredient and the jth ingredient, and ϵ represents the random error. A short-hand notation of the two models is

$$\mathbf{Y} = \mathbf{f}'(\mathbf{x})\boldsymbol{\beta} + \epsilon, \tag{3}$$

where $\mathbf{x}$ is the vector containing the ingredient proportions $x_1, x_2, \ldots, x_q$, $\mathbf{f}(\mathbf{x})$ is the model expansion of $\mathbf{x}$, and $\boldsymbol{\beta}$ is the vector containing the model's coefficients β_i and, for the second-order model, β_{ij}. For the first-order Scheffé model, $\mathbf{f}(\mathbf{x}) = \mathbf{x}$, while, for the second-order Scheffé model, it is $\mathbf{f}(\mathbf{x}) = (\mathbf{x}', x_1x_2, x_1x_3, \ldots, x_{q-1}x_q)'$. The dimension of $\mathbf{f}(\mathbf{x})$ and $\boldsymbol{\beta}$ is denoted by p in the remainder of this paper. It equals q for first-order models and $q(q+1)/2$ for second-order models.

In matrix notation, the models in Equations (1), (2) and (3) can be written as

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\mathbf{Y}$ is the n-dimensional vector of responses, $\mathbf{X}$ represents the $n \times p$ model matrix containing the settings and the model expansions of all experimental factors, i.e. the mixture ingredient proportions $x_1, \ldots, x_q$, for each of the n runs of the experiment, and $\boldsymbol{\epsilon}$ is an n-dimensional vector containing the random errors of the individual runs. We assume that $\boldsymbol{\epsilon}$ is normally distributed with zero mean and variance-covariance matrix $\sigma_\epsilon^2 \mathbf{I}_n$. In that case, the best linear unbiased estimator of the coefficient vector $\boldsymbol{\beta}$ is the ordinary least squares estimator

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}. \tag{4}$$

2.2 Constrained mixture experiments

Mixture experiments are often characterized by the presence of constraints on the ingredient proportions. A simple, common constraint is a lower bound on the proportion of each of the ingredients. In general, this constraint can be written as

$$0 \leq L_i \leq x_i,$$

where L_i represents the lower bound for the proportion of ingredient i. In case lower bounds on individual ingredient proportions are the only constraints, the experimental region remains a regular simplex. Kurotori (1966) suggested transforming the constrained ingredient proportions into coded values called L-pseudocomponents. The coded value of the ith ingredient proportion is

$$x_i^* = \frac{x_i - L_i}{1 - \sum_{i=1}^{q} L_i}.$$

Because the pseudocomponents are linear transformations of the original ingredient proportions, finding a D-optimal or an I-optimal design expressed in terms of pseudocomponents is mathematically equivalent to finding a D-optimal or an I-optimal design expressed in terms of the original ingredient proportions (Atkinson, Donev and Tobias, 2007).

2.3 Simplex-lattice designs for mixture experiments

A $\{q, m\}$ simplex-lattice design consists of all combinations of the proportions $\{0,\ {}^1\!/\!_m, {}^2\!/\!_m, \ldots, 1\}$ for the q ingredients whose proportions sum to one. The simplex-lattice design involves

$$\binom{m+q-1}{m}$$

different combinations or design points. For example, a $\{3, 1\}$ simplex-lattice design for three ingredients involves three design points, $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$. As each design point requires 100% of one ingredient, the design points correspond to the pure components. A $\{3, 2\}$ simplex-lattice design consists of six design points, $(1, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$, $(0, 1/2, 1/2)$, $(1/2, 1/2, 0)$, and $(1/2, 0, 1/2)$. The first three design points are pure components. The last three design points are binary mixtures since they involve nonzero proportions of only two ingredients. In this paper, we subsequenttly present examples in which ternary, quaternary, and quinary mixtures appear in the experimental design too, i.e. where there are three, four, and five nonzero ingredient proportions.

Any $\{q, 1\}$ simplex-lattice design involves the q vertices of the experimental region, whereas any $\{q, 2\}$ simplex-lattice design involves the q vertices and $q(q-1)/2$ binary mixtures involving 50% of one ingredient and 50% of another. These two types of designs were associated with the first- and second-order Scheffé model, respectively (Scheffé, 1958). Simplex-lattice designs can be constructed for any simplex-shaped experimental region, either in the original ingredients (if there are no lower bounds on the ingredient proportions) or in the pseudocomponents (in case there are lower bounds on the ingredient proportions). Note that, if there is a nonzero lower bound on each ingredient proportion, a three-ingredient simplex-lattice design in the pseudocomponents involves the points $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$, but these points are in fact ternary mixtures.

2.4 D-optimality criterion

One of the primary goals of a mixture experiment is to estimate the parameter vector $\boldsymbol{\beta}$. Ideally, we do so with maximum precision. A matrix that quantifies the precision of the ordinary least squares estimator $\hat{\boldsymbol{\beta}}$ in Equation (4) is the variance-covariance matrix

$$\mathrm{cov}(\hat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}'\mathbf{X})^{-1}.$$

The most commmonly used approach to obtain a *small* variance-covariance matrix is to minimize its determinant, or, equivalently, to maximize the determinant of the information matrix

$$\sigma^{-2}(\mathbf{X}'\mathbf{X}) = \sigma^{-2}\sum_{i=1}^{n} n_i \mathbf{f}(\mathbf{x}_i)\mathbf{f}'(\mathbf{x}_i),$$

which is the inverse of the ordinary least squares estimator's variance-covariance matrix. In this expression, n_i is the number of replicates of ith design point. The design which maximizes the determinant of the information matrix is called D-optimal, and the corresponding design selection criterion is referred to as the D-optimality criterion. Note that σ^2 is an irrelevant constant, as far as the search for an optimal design is concerned. Therefore, we can set it to one without loss of generality.

Kiefer (1961) showed that, in the absence of constraints other than the mixture constraint, the $\{q, 1\}$ and $\{q, 2\}$ simplex-lattice designs give the D-optimal design points for first-order and second-order Scheffé models, respectively. For a design with a given number of runs n to be D-optimal, its design points have to be replicated as evenly as possible. The $\{q, 1\}$ and $\{q, 2\}$ simplex-lattice designs expressed in pseudocomponents are also D-optimal in case there are lower bound constraints on the proportions.

When comparing the performance of two designs with model matrices $\mathbf{X}_1$ and $\mathbf{X}_2$ in terms of the D-optimality criterion, we use the relative D-efficiency

$$\left(\frac{|\mathbf{X}_1'\mathbf{X}_1|}{|\mathbf{X}_2'\mathbf{X}_2|}\right)^{1/p},$$

as is customary in the optimal design of experiments literature. A relative D-efficiency larger than one means that the design with model matrix $\mathbf{X}_1$ outperforms the one with model matrix $\mathbf{X}_2$.

2.5 I-optimality criterion

Rather than on the estimation of the parameter vector $\boldsymbol{\beta}$, the I-optimality criterion focuses on precise prediction throughout the entire experimental region. An I-optimal design minimizes the average prediction variance

$$\frac{\int_{S_{q-1}} \sigma^2\mathbf{f}'(\mathbf{x})(\mathbf{X}'\mathbf{X})^{-1}\mathbf{f}(\mathbf{x})d\mathbf{x}}{\int_{S_{q-1}} d\mathbf{x}},$$

which can be rewritten as

$$\frac{1}{\int_{S_{q-1}} d\mathbf{x}}\sigma^2 \mathrm{tr}\left[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{B}\right].$$

In these expressions, S_{q-1} represents the experimental region and

$$\mathbf{B} = \int_{S_{q-1}} \mathbf{f}(\mathbf{x})\mathbf{f}'(\mathbf{x})d\mathbf{x}$$

is the moments matrix.

In case the experimental region is a $(q-1)$-dimensional simplex, the individual elements of the moments matrix can be calculated using the expression

$$\int_{S_{q-1}} x_1^{p_1-1} x_2^{p_2-1} \dots x_q^{p_q-1} dx_1 dx_2 \dots dx_{q-1} = \frac{\prod_{i=1}^{q} \Gamma(p_i)}{\Gamma\left(\sum_{i=1}^{q} p_i\right)},$$

which was given, for instance, in De Groot (1970). The volume of a $(q-1)$-dimensional simplex in the absence of lower bounds on the ingredient proportions is

$$\int_{S_{q-1}} d\mathbf{x} = \frac{1}{\Gamma(q)}.$$

In case the experimental region is not a simplex (due to the presence of upper bound constraints and/or multicomponent constraints), the moments matrix can be approximated by randomly selecting points in the region. For a given design problem, the volume of the experimental region is constant. It is therefore irrelevant when it comes to finding an I-optimal design, as is σ^2. In our algorithm for finding I-optimal designs, we omit these constants. We also reformulate the problem of minimizing the average prediction variance as a maximization problem, i.e. we maximize the negative average prediction variance, so we can use the same algorithm for finding D-optimal designs as for finding I-optimal designs in Sections 3 and 4. Whenever we report I-criterion values, we report the actual average prediction variance.

When comparing the performance of two designs with model matrices $\mathbf{X}_1$ and $\mathbf{X}_2$ in terms of the I-optimality criterion, we use the relative I-efficiency

$$\frac{\text{tr}\left[(\mathbf{X}_2'\mathbf{X}_2)^{-1}\mathbf{B}\right]}{\text{tr}\left[(\mathbf{X}_1'\mathbf{X}_1)^{-1}\mathbf{B}\right]}.$$

This relative efficiency takes a value larger than one if the design with model matrix $\mathbf{X}_1$ outperforms the one with model matrix $\mathbf{X}_2$ in terms of average prediction of variance.

For the first-order Scheffé models, the I-optimal designs are the same as the D-optimal ones. For the second-order Scheffé models, the I-optimal designs differ from the D-optimal designs. One difference is that the number of replicates required for the different design points in the I-optimal design is unequal. Another difference is that the I-optimal designs generally involve ternary mixtures, where three of the ingredients have proportion $1/3$. A review and some new results on I-optimal designs for Scheffé models in the absence of ingredient availability constraints are given in Goos, Jones and Syafitri (2013) and Goos and Syafitri (2014). Note that the I-optimality criterion is sometimes also named the V-optimality criterion.

2.6 Fraction of design space plot

The performance of an experimental design in terms of the prediction variance can be visualized by constructing a fraction of design space (FDS) plot, which shows the cumulative distribution of the predicton variance scaled by σ^2,

$$\mathbf{f}'(\mathbf{x})(\mathbf{X}'\mathbf{X})^{-1}\mathbf{f}(\mathbf{x}),$$

across the entire experimental region or design space. The FDS plot allows us to see the minimum and maximum prediction variance, as well as any percentile of the prediction variance. It therefore provides a more detailed picture of a design's performance in terms of the prediction variance than the average prediction variance (i.e., the I-optimality criterion). Ideally, an experimental design leads to small prediction variances in large portions of the experimental region.

3 The problem

The problem of finding an optimal design for a mixture experiment involving availability constraints is similar to a multi-dimensional knapsack problem. In this section, we discuss the similarities in detail.

A key feature of our approach to solving the optimal mixture design problem is that it involves a set of possible design points, called a candidate set and consisting of feasible combinations of ingredient proportions. In any feasible design for a mixture experiment, each of these candidate design points is either used or not. In case a given candidate point is used in the design, it can be used once or more than once. We denote the cth candidate point by $\mathbf{x}_c = (x_{1c}, x_{2c}, \ldots, x_{qc})$ and the number of times the candidate point appears in the design by z_c, where $z_c \in \{0, 1, 2, \ldots\}$.

The challenge is to identify the values of z_c that result in an optimal value for the D- or I-optimality criterion without violating the availability constraint for each of the ingredients. If we denote the number of candidate points by N, then the number of experimental runs corresponding to a given solution for the z_c values equals

$$n = \sum_{c=1}^{N} z_c.$$

If we denote the availaility of the ith ingredient by R_i and the total amount of the mixture required for one experimental run by λ, the availability constraint for ingredient i can be written as

$$\lambda \sum_{c=1}^{N} x_{ic} z_c \leq R_i.$$

There is no explicit upper bound for the number of experimental runs n, but the total amount of the n mixtures used in the experiment cannot, of course, exceed

$$R = \sum_{i=1}^{q} R_i.$$

An implicit upper bound for n therefore is R/λ. Consequently, R/λ is also an upper bound for the value each z_c can take. Without loss of generality, we set $\lambda = 1$ in the remainder of this paper.

The optimal mixture design problem belongs to the class of nonlinear multidimensional knapsack problems. Knapsack problems have been categorized as difficult or NP-hard optimization problems (Kellerer, Pferschy and Pisinger, 2010). The simplest possible knapsack problem involves maximizing the value of a knapsack by filling it with items chosen from a list of N candidate items. Each of the items c has a certain value v_c and a certain weight w_c. The knapsack problem can then be formulated as follows:

$$\text{Maximize } \sum_{c=1}^{N} v_c z_c$$

subject to

$$\sum_{c=1}^{N} w_c z_c \leq C,$$

and

$$z_c \in \{0, 1\},$$

where C denotes the capacity of the knapsack. In this formulation, the decision variable z_c takes the value 1 if item c is selected and 0 otherwise. As a consequence of the fact that z_c can only take the values 0 or 1, the problem is called binary. The problem is unidimensional because there is only one capacity constraint, which imposes an upper bound on the total weight of all selected items. The problem is linear because both the objective function and the capacity constraints involve linear combinations of the z_c values.

Several extensions of the unidimensional knapsack problem have been described in the literature. In one extension, the decision variables z_c can take integer values other than 0 or 1. If there is an upper bound for the z_c values, as is the case in the mixture design problem, the problem is called bounded. In another extension of the basic knapsack problem, the objective function is quadratic (Gallo, Hammer and Simeoni, 1980, Billionnet and Calmels, 1996, Pisinger, 2007) or another nonlinear function of the decision variables.

The literature on knapsack problems also involves articles dealing with multiple linear knapsack constraints. These knapsack problems are usually referred to as multidimensional knapsack problems. Chu and Beasley (1998) provide an overview of much of the work on multidimensional knapsack problems with a linear ojective function. Djerdjour, Mathur and Salkin (1988), Quadri, Soutif and Tolla (2009) and Wang, Kochenberger and Glover (2012) discuss various techniques for solving a quadratic knapsack problem involving multiple knapsack constraints. Our mixture design problem also involves multiple linear knapsack constraints, one for each ingredient. Therefore, the mixture design problem belongs to the class of multidimensional knapsack problems.

The original knapsack problem involves a linear objective function and a linear constraint. Morin and Marsten (1976) define a nonlinear knapsack problem involving both a nonlinear objective function and nonlinear constraints. Their nonlinear constraints and objective function are sums of nonlinear functions of individual decision variables z_c. Therefore, their problem is called separable: the contribution of a certain item c to the objective function and to the left-hand side of each of the knapsack constraints does not depend on which other items are selected.

Our mixture design problem is also a nonlinear knapsack problem because its objective functions are nonlinear functions of the design points selected. In other words, the D- and I-optimality criteria are nonlinear functions of the decision variables z_c. For the D-optimality criterion, the objective function can be written as

$$\text{Maximize det}\left\{\sum_{c=1}^{N} z_c \mathbf{f}(\mathbf{x}_c)\mathbf{f}'(\mathbf{x}_c)\right\},$$

where the sum between curly braces produces the information matrix $\mathbf{X}'\mathbf{X}$ corresponding to the selected experimental design. For the I-optimality criterion, the objective function can be formulated as

$$\text{Maximize } - \text{ tr}\left[\left\{\sum_{c=1}^{N} z_c \mathbf{f}(\mathbf{x}_c)\mathbf{f}'(\mathbf{x}_c)\right\}^{-1}\mathbf{B}\right].$$

A key feature of the two nonlinear objective functions is that they are nonseparable, which means that the objective function cannot be rewritten as a sum of univariate functions $g(z_c)$. As a result, the contribution of one or more replicates of a design point to the objective function depends on the other points that have been selected in the design. In a review paper, Bretthauer and Shetty (2002) state that few papers have addressed these kinds of knapsack problems. The papers that did study nonlinear objective functions focus on quadratic knapsack problems, which have a much simpler objective function than the optimal mixture design problem.

Finally, it is important to point out that the same design point can be included more than once in any given mixture design. Due to the fact that there is random error when recording the response at every experimental test, replicating experimental tests is useful and can be optimal. In our knapsack problem, the decison variables z_c can therefore be larger than one. Given that the total availability of the ingredients, R, forms an upper bound for each z_c, the problem is bounded.

As a conclusion, the problem of finding an optimal design for mixture experiments involving availability constraints is a bounded nonlinear, nonseparable multidimensional knapsack problem. As a result thereof, many solution approaches presented in the literature for knapsack problems cannot be used. This is because the approaches for the simplest knapsack problems exploit the linear nature of the objective function and the constraints, those for quadratic knapsack problems also exploit the specific nature of the objective function, and those for nonlinear knapsack problems focus on the subclass of separable nonlinear knapsack problems.

We use a metaheuristic approach because such an approach has been used successfully to tackle various NP-hard problems. More specifically, we utilize a variable neighborhood descent algorithm in order to find D- and I-optimal designs for mixture experiments involving availability constraints.

4 Variable neighborhood descent algorithm

4.1 Main idea

Variable neighborhood search (VNS) was introduced by Mladenović and Hansen (1997) as an improvement over local-search-based algorithms for combinatorial optimization. In local-search-based algorithms, the search for a good solution is done by iteratively making one small change (called a move) to the current solution s. All solutions s' that can be reached in one single move starting from a given solution s form the neighborhood of that solution, denoted by $N(s)$. A solution whose neighborhood does not contain any better solution is called a local optimum with respect to the neighborhood. Many metaheuristics have been developed, each using a different strategy to escape from these local optima and hopefully reach a globally optimal solution.

Unlike local-search-based algorithms, VNS systematically explores different neighborhood structures (i.e., neighborhoods defined by different types of moves). The main idea of VNS is that a solution that is a local optimum relative to a certain neighborhood structure not necessarily is a local optimum relative to another neighborhood structure. For this reason, escaping from a locally optimal solution can be done by changing the neighborhood structure. VNS has been successfully applied to a wide variety of optimization problems such as vehicle routing (Kytöjoki, Nuortio, Bräysy and Gendreau, 2007), project scheduling (Fleszar and Hindi, 2004), automatic discovery of theorems (Caporossi and Hansen, 2004), graph coloring (Avanthay, Hertz and Zufferey, 2003), and the synthesis of radar polyphase codes (Mladenović, Petrović, Kovačević-Vujčić and Čangalović, 2003). In the field of experimental design, VNS has been used for the determination of optimal run orders for the design points of standard experimental designs in the presence of serially correlated responses (Garroi, Goos and Sörensen, 2009) and to assign treatments to whole plots in orthogonal split-plot designs (Sartono, Goos and Schoen, 2015). In both cases, solving the optimal experimental design problem at hand came down to finding an optimal permutation. This is very different from the optimal design problem considered in this paper, where the optimal number of experimental runs needs to be found, as well as the optimal design points and the number of times each design point is used.

Most implementations of VNS algorithms explore the different neighborhoods in a sequential fashion, starting with the neighborhood that includes the fewest solutions and ending with the neighborhood that contains the largest number of solutions. Larger neighborhoods are only explored when the current solution is a local optimum with respect to all smaller neighborhoods, to save computing time.

4.2 Our algorithm

Our algorithm uses the deterministic variant of VNS, named variable neighborhood descent (VND). In the first step, a starting design is created. In the second step, different neighborhoods are explored to improve the starting design, so as to turn it into a solution that is a local optimum with respect to all neighborhoods. In this section, we first discuss the input to the algorithm. Next, we present the different neighborhoods we utilize, as well as a detailed description (including pseudocode) of the algorithm.

4.2.1 Input

Key input to our algorithm is the set of mixtures that can be used in the experiment (i.e., the possible design points). In the optimal experimental design literature, this set is usually referred to as the candidate set. We start the construction of the candidate set by considering all points of the $\{q, h\}$ simplex-lattice design, where q denotes the number of ingredients in the mixture and h is a nonzero integer. In case there are lower and upper bounds for the ingredient proportions, and/or other constraints involving more than one ingredient, the simplex-lattice points that do not satisfy the constraints have to be dropped. The remaining points then form the final candidate set.

Ideally, the candidate set is as large as possible, implying that the value of h should be as large as possible. However, for a mixture experiment to be practically feasible, h should not be excessively large. Moreover, there is a decreasing marginal utility for increasing the value of h. In other words, the extent to which the

D- and I-optimality of a design can be improved by increasing h by one unit becomes smaller and smaller for larger values of h. In our examples in the next sections, we set h to 200 for problems involving two ingredients and to 20 for problems involving three or more ingredients.

Besides the candidate set, which we call $\mathcal{C}$ in the remainder of this paper, the input to our algorithm consists of the model specification (first-order or second-order) and the availabilities $R_1, R_2, \ldots, R_q$ of the q ingredients.

4.2.2 Starting design

Our algorithm starts by randomly generating an initial design. This is done by randomly selecting points from the candidate set $\mathcal{C}$ until at least one of the ingredient availabilities is exceeded. The final feasible design produced by this procedure is the starting design for the VND procedure. The procedure to obtain the initial design is described in Algorithm 1, where $\mathcal{S}$ represents the starting design.

Algorithm 1 Pseudocode for generating a feasible starting design $\mathcal{S}$

```
1: Set $i \leftarrow 1$
2: $\mathcal{S} = \varnothing$
3: repeat
4:     Randomly choose a candidate point from $\mathcal{C}$ and name it $\mathbf{x}_i$
5:     $\mathcal{S} = \mathcal{S} \cup \mathbf{x}_i$
6:     Set $i \leftarrow i + 1$
7: until $\mathcal{S}$ violates one or more of the availability constraints
8: $\mathcal{S} = \mathcal{S} \setminus \mathbf{x}_{i-1}$
9: STOP
```

The number of design points in the starting design produced by Algorithm 1 is random. The maximum number of points in the starting design depends on the ingredient availabilities and on the lower bounds for each of the ingredients in the mixture.

4.2.3 The variable neighborhood descent

Our algorithm improves the starting design by iteratively making small changes to the design. The algorithm considers four types of changes or moves, each of which defines a neighborhood. The first type of move involves adding a point from the candidate set to the design. Therefore, the neighborhood N_0 is the set of all designs that can be obtained from a design by adding a point from the candidate set. The point added does not need to be different from the points in the current design. The second type of move involves replacing a point from the current design by a point from the candidate set. This move type defines neighborhood N_1. The third move type involves replacing a design point with two points from the candidate set and defines neighborhood N_2. The final type of move, which determines neighborhood N_3, involves the replacement of two points from the design by two points from the candidate set. Each of the four move types in our algorithm has been used on various occasions in the literature on the construction of optimal experimental designs. A unique feature of our algorithm is that it combines all these move types to tackle a novel problem in the optimal design of mixture experiments. The move types which define the neighborhoods N_0, N_1, N_2 and N_3 were used by Wynn (1972), Fedorov (1972), Mitchell (1974) and Vuchkov and Donev (1985), among

Table 1: Neighborhood structures N_k for the VND algorithm

N_k	Size	Description
N_0	N	Add a candidate point to the current design
N_1	$n_c \times N$	Remove a current design point and add a candidate point
N_2	$n_c \times N^2$	Remove a current design point and add two candidate points
N_3	$0.5n_c(n_c - 1) \times N^2$	Remove two current design points and add two candidate points

others. In our algorithm, we only consider feasible solutions (i.e., mixture designs that do not violate the availability constraints). Therefore, the neighborhoods in our algorithm only involve feasible designs.

Neighborhood N_0 is the smallest of the four neighborhoods, while neighborhood N_3 is the largest. Denoting the number of points in the design under evaluation by n_c, these two neighborhoods contain at most N and $0.5n_c(n_c - 1) \times N^2$ solutions, respectively. Neighborhoods N_1 and N_2 contain at most $n_c \times N$ and $n_c \times N^2$ solutions, respectively. An overview of the four neighborhoods and their maximum sizes is shown in Table 1.

Our algorithm explores the four neighborhoods, starting with the smallest neighborhood N_0 and ending with the largest one, N_3. This means that the algorithm starts by seeking designs in neighborhood N_0 that have a better D- or I-optimality criterion value than the current design. As soon as such an improved design has been found, the neighborhood N_0 of that newly found design is constructed and explored to attempt to find an even better design in terms of the D- or I-optimality criterion. At some point in time, this procedure will result in a design for which the neighborhood N_0 does not contain any better alternative. That design then is a local optimum with respect to neighborhood N_0. It might, however, not be a locally optimal design with respect to neighborhood N_1. Therefore, the next step is to explore the neighborhood N_1 of that design until a better one is found. As soon as a better design has been obtained, the algorithm switches back to the smallest neighborhood N_0 and attempts to find a better design in that neighborhood. If no better design can be found in neighborhood N_1, then the algorithm switches to neighborhood N_2. Finally, if neighborhood N_2 does not contain any better design, neighborhood N_3 is explored. This process is repeated until a design is obtained that is a local optimum with respect to all of the neighborhoods N_0-N_3.

Each time an improved solution is found in one of the larger neighborhoods, the algorithm returns to the smallest neighborhood N_0. This way, the algorithm avoids having to explore the largest neighborhoods oftentimes, thereby limiting its computing time. Note that, when the number of experimental runs in the current design is maximal (because all the available resources have been used), the neighborhoods N_0 and N_2 are skipped because it is then no longer possible to add experimental runs to the design. The structure of the VND algorithm is shown schematically in Figure 1.

Our algorithm is a *first-improvement algorithm*, since it does not evaluate all alternative designs contained within a neighborhood. Instead, it evaluates the alternatives contained within the neighborhood of a given design one by one, until a better design is found. The search process is then re-centered around the newly obtained better design, and neighborhood N_0 of that newly obtained design is investigated. By using a first-improvement algorithm, we follow the recommendation by Hansen and Mladenović (2006), who state that a first-improvement algorithm is better and faster than a best-improvement algorithm when the initial solution is chosen at random.

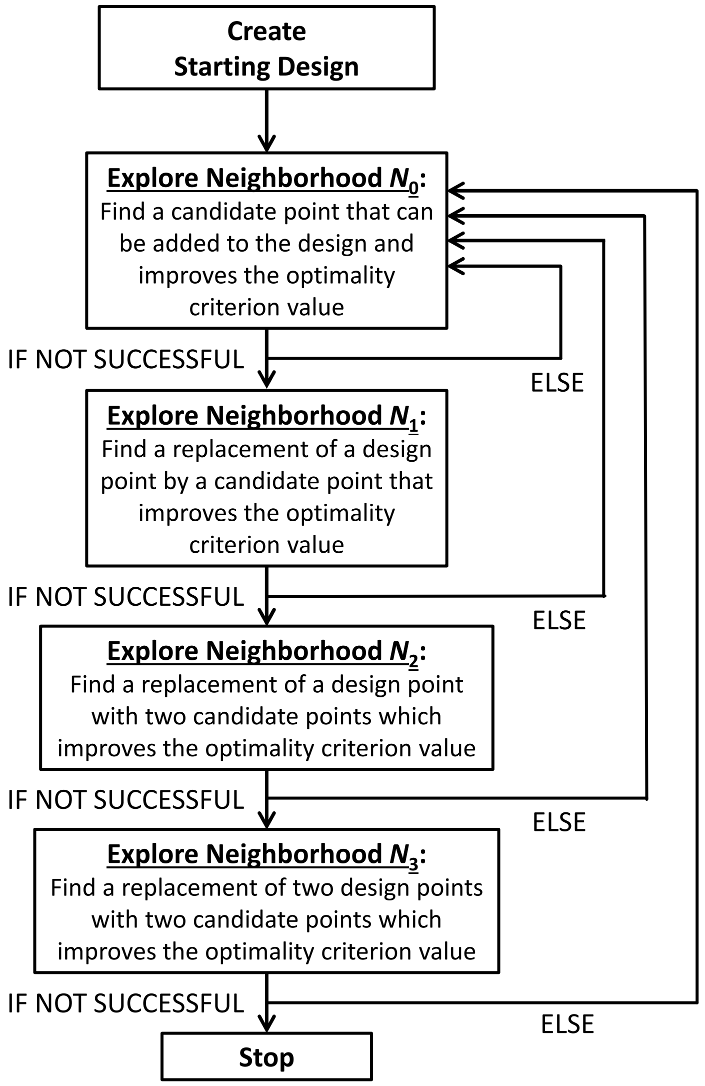

Create Starting Design
Explore Neighborhood N_0: Find a candidate point that can be added to the design and improves the optimality criterion value
IF NOT SUCCESSFUL
ELSE
Explore Neighborhood N_1: Find a replacement of a design point by a candidate point that improves the optimality criterion value
IF NOT SUCCESSFUL
ELSE
Explore Neighborhood N_2: Find a replacement of a design point with two candidate points which improves the optimality criterion value
IF NOT SUCCESSFUL
ELSE
Explore Neighborhood N_3: Find a replacement of two design points with two candidate points which improves the optimality criterion value
IF NOT SUCCESSFUL
ELSE
Stop

Figure 1: Flow chart of one execution of the variable neighborhood descent (VND) algorithm for finding a D- or I-optimal mixture design in the presence of ingredient availability constraints.

Algorithm 2 Pseudocode of the VND algorithm for finding an optimal design

```
1: $objmax \leftarrow -\infty$
2: $k \leftarrow 0$
3: repeat
4:     Generate a random initial design $\mathcal{S}$ using Algorithm 1.
5:     $y \leftarrow obj(\mathcal{S})$
6:     $t \leftarrow 0$
7:     repeat
8:         repeat
9:             Randomly select a design $\mathcal{S}_{\text{new}} \in N_t(\mathcal{S})$
10:            if $obj(\mathcal{S}_{\text{new}}) > y$ then
11:                $\mathcal{S} \leftarrow \mathcal{S}_{\text{new}}$
12:                $y \leftarrow obj(\mathcal{S}_{\text{new}})$
13:                Return to Line 6
14:            else
15:                $N_t(\mathcal{S}) \leftarrow N_t(\mathcal{S}) \setminus \{\mathcal{S}_{\text{new}}\}$
16:            end if
17:        until $N_t(\mathcal{S}) = \varnothing$
18:        $t \leftarrow t + 1$
19:    until $t = 3$
20:    if $obj(\mathcal{S}) > objmax$ then
21:        $\mathcal{S}_{\text{max}} \leftarrow \mathcal{S}$
22:        $objmax \leftarrow obj(\mathcal{S}_{\text{max}})$
23:    end if
24:    $k \leftarrow k + 1$
25: until $k = k_{\text{max}}$
26: Return $\mathcal{S}_{\text{max}}$.
```

In order to increase the likelihood of finding a globally optimal mixture design, we repeat the entire search a prespecified number of times, each time starting from a different randomly generated initial solution. The pseudocode outline of the VND algorithm for generating D- or I-optimal designs is provided in Algorithm 2. In the pseudocode, *objmax* represents the overall best objective function value found by the algorithm, and $\mathcal{S}_{\text{max}}$ represents the corresponding mixture design solution. The parameter k_{max} denotes the maximum number of iterations of the algorithm, while k represents the current iteration.

We implemented the VND algorithm in the SAS procedure iml. We extracted the moments matrices required for the computation of I-optimal designs from the software package JMP.

4.2.4 Computational issues

In order to reduce the computational time of our algorithm, we used updating formulas for evaluating the D-optimality criterion and the I-optimality criterion for each design considered in its course. There is a rich history of using update formulas in construction algorithms for D- and I-optimal designs (see, e.g., Meyer and Nachtsheim, 1995, Goos and Vandebroek, 2004, Arnouts and Goos, 2010). The update formula required for evaluating the D-optimality criterion value after adding a point to the design is well-known and given in, for instance, Meyer and Nachtsheim (1995) and Goos (2002). This update formula is important for exploring neighborhood N_0 in a computationally efficient way. Similarly, the update formula for evaluating the impact of a replacement of one design point by one point from the candidate set is well-known and given in Goos

(2002). This update formula is important for exploring neighborhood N_1. We did not find update formulas for efficiently exploring neighborhoods N_2 and N_3 in the literature. We derive the required formulas in the appendix, where we also provide a formula for updating the inverse of the information matrix for each of the neighborhoods. This is useful when evaluating the I-optimality criterion value after a change in the design.

Using the update formulas requires the information matrix, $\mathbf{X}'\mathbf{X}$, to be non-singular. It is possible, however, that the number of distinct design points in the starting design is smaller than the number of parameters p in the Scheffé model under consideration. In that case, the information matrix is singular. We circumvent this problem by replacing the information matrix $\mathbf{X}'\mathbf{X}$ by $\mathbf{X}'\mathbf{X} + \omega\mathbf{I}_p$, where ω is a small positive number, for as long the current design is singular. This approach is common in the optimal experimental design literature (Atkinson, Donev and Tobias, 2007).

5 The algorithm working

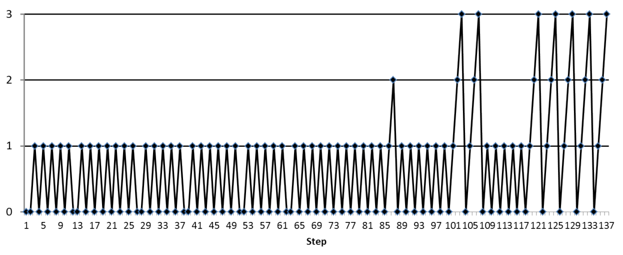
In this section, we demonstrate the usefulness of each of the neighborhoods of the VND algorithm. In the example we use for the demonstration, each experimental run requires at least 0.3 kg of ingredient 1 and 0.2 kg of ingredient 3. The available stock of ingredient 1 is 10.2 kg, the available stock of ingredient 2 is 4 kg, and the available stock of ingredient 3 is 4.9 kg. The total stock is 19.1 kg, so that the maximum number of runs in the experiment is 19, assuming that every experimental run requires a mixture of 1 kg. The candidate set for this problem involves 66 points, i.e. $N = 66$. We ran the VND algorithm on a standard CPU (Intel(R) Core(TM) i7 processor, 2.93 Ghz, 4 GB) using 30 iterations, i.e. using $k_{\max} = 30$.

5.1 D-optimal design

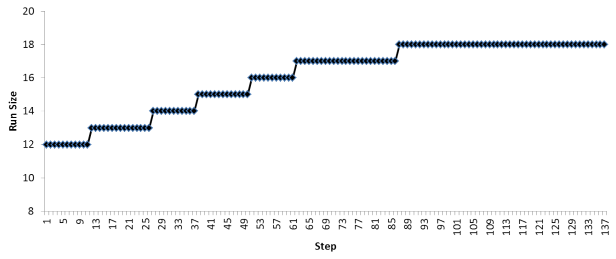
The best design found by the VND algorithm in terms of D-optimality was obtained in the tenth iteration. The number of runs of the initial design in that iteration was equal to 11. Next, the VND algorithm iteratively improved this initial design until it returned an 18-run design. Figure 2 shows how the VND algorithm moved from one neighborhood to the other to improve the design step by step, along with its impact on the number of runs and the D-efficiency of each intermediate design relative to the D-optimal design, and the computation time in seconds.

Figure 2(a) visualizes how often the neighborhoods N_0, N_1, N_2 and N_3 are visited by the VND algorithm. The horizontal axis shows that the VND algorithm uses 136 steps to move from the initial design to the final design. The figure shows dots at four different levels. There is a dot on the horizontal axis each time neighborhood N_0 is visited. There is a dot at the next higher level each time neighborhood N_1 is visited. Similarly, there is a dot at the next to highest level each time neighborhood N_2 is visited, and a dot at the highest level each time neighborhood N_3 is visited.

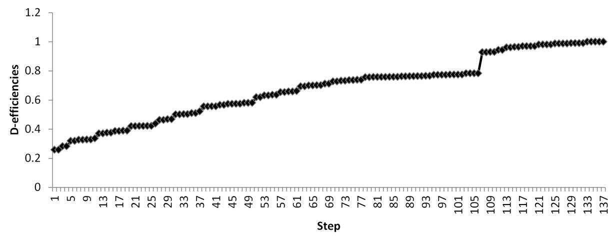
In its first step, the algorithm visits neighborhood N_0 and identifies a better design in that neighborhood. This causes the number of runs to increase by one unit, from 11 to 12. After this improvement to the design, neighborhood N_0 is revisited. The two successive visits of N_0 are shown in Figure 2(a) by means of the two dots on the horizontal axis at the extreme left. Because the second visit of N_0 did not yield an improved design, the VND algorithm moved to neighborhood N_1, where it was able to find an improved design with the same number of runs, 12. This causes the algorithm to return to neighborhood N_0 of the new design. Because this did not lead to an improvement in the design, the VND algorithm continued with neighborhood



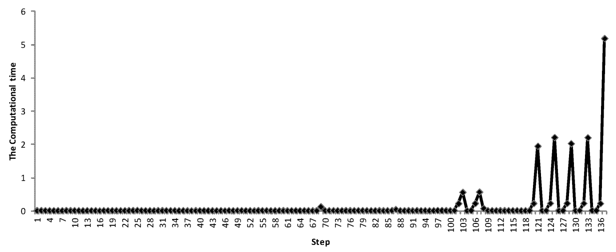
(a) Different steps of the VND algorithm



(b) Evolution of the number of runs in the course of the VND algorithm



(c) Increase in D-efficiency in the course of the VND algorithm



(d) Execution time for each neighborhood

Figure 2: Working of the VND algorithm when constructing a D-optimal design.

N_1, again with success. Figure 2(a) shows that the alternating visits to the first two neighborhoods continue until step 12. During that step, the visit of neighborhood N_0 leads to an improved design with 13 runs, after which it is neighborhood N_1 again that repeatedly produces improved designs. The figure shows that the majority of the design improvements was found by visiting neighborhood N_1, but at steps 27, 38, 51 and 62, a visit of neighborhood N_0 results in a better design, each time with one extra experimental run.

The next remarkable event in the course of the VND algorithm occurs in steps 85 and 86, when neither N_0 nor N_1 allow the algorithm to find an improved design. The algorithm therefore explores neighborhood N_2 in step 87. This does result in an improvement of the design, after which the algorithm returns to neighborhoods N_0 and N_1, until step 100. The design produced by the VND algorithm at that step cannot be improved by any of the neighborhoods N_0, N_1 and N_2, which is why neighborhood N_3 is visited. In the final steps of the algorithm, it is visiting the neighborhoods N_1 and N_3 that results in improved designs. The algorithm's final step is a visit to neighborhood N_3, which does not produce any better design and causes the VND algorithm to terminate its tenth iteration.

Figure 2(b) shows the evolution of the number of experimental runs in the course of the design generation by the VND algorithm. In the first step of the algorithm, the number of runs immediately increases by one, from 11 (the number of runs in the initial design) to 12 (due to the fact that one run is added after visiting neighborhood N_0 for the first time). The number of runs remains 12 until step 12, at which another visit of neighborhood N_0 results in an improved design with 13 runs. Additional successful visits of neighborhood N_0 occur in steps 27, 38, 51 and 62, after which a 17-run design is obtained and the number of runs remains stable until step 87. During that step, the algorithm's first visit of neighborhood N_2 leads to the final increase in number of runs, from 17 to 18.

Figure 2(c) shows the increase in D-efficiency in the course of the VND algorithm. The D-efficiency gradually increases until it reaches the value 1 when the D-optimal design is found. The graph shows that the D-efficiency increases after a successful visit to a neighborhood and remains stable whenever a visit to a neighborhood does not result in an improvement of the design. In the final four steps, the D-efficiency is 1 because these steps did not lead to a design improvement.

Finally, Figure 2(d) visualizes the computation time absorbed by the 136 steps of the VND algorithm. The computation time for visiting the neighborhoods N_0 and N_1 is negligible compared to that used when visiting neighborhood N_3 after step 100. By far the largest computation time is due to the exploration of neighborhood N_3 in the final steps of the algorithm. This is because N_3 is the largest neighborhood, and because evaluating the quality of a solution in that neighborhood is computationally more demanding than evaluating the quality of solutions in the other neighborhoods. This can be seen from the update formulas in the appendix. Note also that the computation time required by neighborhood N_3 varies from step to step, because our algorithm is a first-improvement algorithm. This implies that the VND algorithm returns to neighborhood N_0 as soon as an improved solution has been identified in N_3. Figure 2(d) shows that the first few visits to N_3 absorb little computing time, because it was relatively easy to find an improved solution in N_3 in the earlier stages of the algorithm. The computation time absorbed by N_3, however, increases in the final stages of the VND algorithm, indicating that it becomes increasingly difficult to find a better solution. The final visit to neighborhood N_3 takes longest because all solutions in the neighborhood are evaluated during that visit, to confirm that the final solution is a local optimum with respect to N_3. The total computation time for the tenth iteration of the VND algorithm shown in Figure 2 was 21.717 seconds.

5.2 I-optimal design

The best design found by the VND algorithm in terms of I-optimality was produced in the first iteration. The number of runs of the initial design was equal to 13. Next, the VND algorithm improved this initial design until it returned a 15-run design. Figure 3 shows how the VND algorithm moved from one neighborhood to the other to improve the initial design step by step, along with its impact on the number of runs in the design and the I-efficiency relative to the I-optimal design, and the computation time in seconds.

Figure 3(a) shows that the most important neighborhood was N_1, while neighborhood N_0 did not lead to a single design improvement in this iteration of the VND algorithm. Neighborhoods N_2 and N_3 allowed the algorithm to identify an improved design twice and four times, respectively. In the final visit to N_3, no better design could be found, so that the final design is locally optimal with respect to N_3. Obviously, the final design is also locally optimal with respect to neighborhoods N_0, N_1 and N_2. Otherwise, the VND algorithm would not have visited N_3 in its final step.

Figure 3(b) shows that the number of runs in the design remains stable at 13 until step 78 of the VND algorithm. In that step, an improved design is found in neighborhood N_2, which causes the number of experimental runs to increase by one unit. Similarly, the number of runs increases from 14 to 15 in step 94 due to another improved solution found in neighborhood N_2. During all other steps of the VND algorithm, the number of runs does not change, so that the final design involves 15 runs.

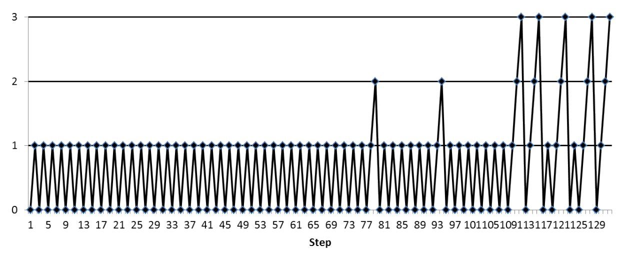
Figure 3(c) shows the increase in I-efficiency in the course of the algorithm. The I-efficiency gradually increases and reaches the value 1 when the I-optimal design is found. The graph shows that the I-efficiency goes up after each successful visit to a neighborhood and remains stable whenever a visit to a neighborhood does not result in an improvement of the design.

Figure 3(d) shows computation time absorbed by the visits to the different neighborhoods. Again, it is only the exploration of neighborhood N_3 which uses a substantial amount of computation time. In particular, it is the final visit to N_3 which is time consuming. This is again due to the fact that all solutions in that neighborhood need to be evaluated to confirm the optimality of the final design with respect to N_3.

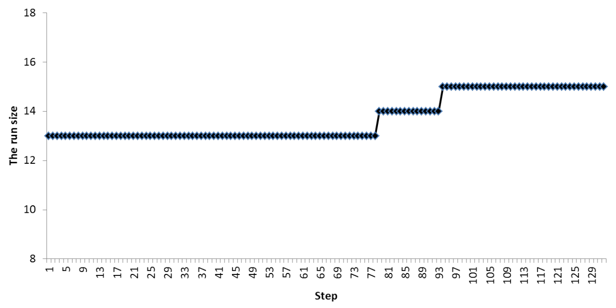
In total, the computation time for the VND iteration shown in Figure 3 was 13.213 seconds. In general, the computation time for generating I-optimal designs is longer than that for generating D-optimal designs. This is due to the fact that evaluating any possible solution to an I-optimal design problem requires the computation of the inverse of the information matrix, while this is not the case for evaluating possible solutions to a D-optimal design problem. This can be seen from the update formulas in the appendix.

6 The optimal designs

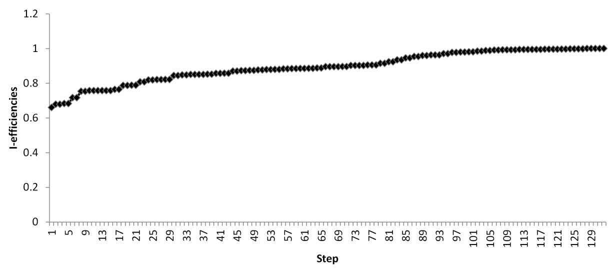
In this section, we discuss the results for various optimal experimental design problems. For each problem, we computed a D-optimal design as well as an I-optimal design. The problems studied correspond to the scenarios listed in Table 2. The scenarios involve different numbers of ingredients q, different lower bounds L_i on the ingredient proportions, different availabilities R_i of the ingredients, and first- and second-order Scheffé models.



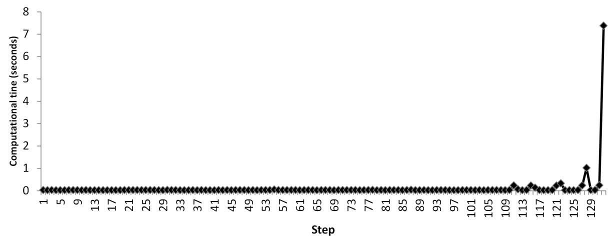
(a) Different steps of the VND algorithm



(b) Evolution of the number of runs in the course of the VND algorithm



(c) Increase in I-efficiency in the course of the VND algorithm



(d) Execution time for each neighborhood

Figure 3: Working of the VND algorithm when constructing a I-optimal design.

Table 2: Twelve scenarios with different numbers of ingredients (q), lower bounds (L_i), availabilities (R_i) and model types.

q	L_i	R_i	Order of Model	Scenario
2	0.25, 0.5	2.5, 4.5	First Second	2.1 2.2
3	0, 0, 0	1.5, 3, 3	First Second	3.1.1 3.1.2
	0, 0, 0	4, 4, 5	First Second	3.2.1 3.2.2
	0.3, 0, 0.2	10.2, 4, 4.9	First Second	3.3.1 3.3.2
	0.1, 0.2, 0.1	2.5, 4, 10	First Second	3.4.1* 3.4.2*
4	0.2, 0.1, 0.1, 0.2	2.5, 6, 3, 7	First Second	4.1 4.2
6	0.05, 0.1, 0.1, 0.1, 0.2, 0.2	4, 4, 5, 5, 8, 16	First Second	6.1 6.2

* The scenarios marked with an asterisk involve additional constraints.

6.1 Two ingredients

The first design problem involves two ingredients. Each experimental run requires at least 0.25 kg of the first ingredient and at least 0.5 kg of the second ingredient. The available stock of the two ingredients is 2.5 kg and 4.5 kg, respectively.

When the interest is in a first-order model (Scenario 2.1), the D-optimal design and the I-optimal design coincide, and involve four runs at $(x_1, x_2) = (0.25, 0.75)$ and three runs at $(x_1, x_2) = (0.5, 0.5)$. A feature of the coinciding designs is that they are unbalanced: the first design point is replicated more often than the latter.

When the interest is in a second-order model (Scenario 2.2), the D-optimal design and the I-optimal design differ. The D-optimal design involves three distinct design points: it has three runs at $(x_1, x_2) = (0.25, 0.75)$, two runs at $(x_1, x_2) = (0.375, 0.625)$ and two runs at $(x_1, x_2) = (0.5, 0.5)$. The D-optimality criterion value of the D-optimal design equals 0.000183, while its I-optimality criterion value is 0.3778.

The I-optimal design involves four distinct design points rather than three. It has two runs at $(x_1, x_2) = (0.25, 0.75)$, three runs at $(x_1, x_2) = (0.355, 0.645)$, one run at $(x_1, x_2) = (0.435, 0.565)$ and one run at $(x_1, x_2) = (0.5, 0.5)$. The I-optimality criterion value of the I-optimal design is 0.330893. The I-efficiency of the D-optimal design relative to the I-optimal one is therefore 87.59%. It shows that the I-optimal design is better than the D-optimal design in terms of prediction. The D-efficiency of the I-optimal design relative to the D-optimal one is 87.15%, as the D-criterion value for the I-optimal design is 0.0001.

The fact that the I-optimal design does not involve the design point $(x_1, x_2) = (0.375, 0.625)$, which is right in the middle between the points $(x_1, x_2) = (0.25, 0.75)$ (using the smallest possible proportion of the first ingredient) and $(x_1, x_2) = (0.5, 0.5)$ (using the smallest possible proportion of the second ingredient), is surprising. This is because the simplex-lattice design, which is usually associated with second-order Scheffé models, does involve the point $(x_1, x_2) = (0.375, 0.625)$ for the constrained experimental region

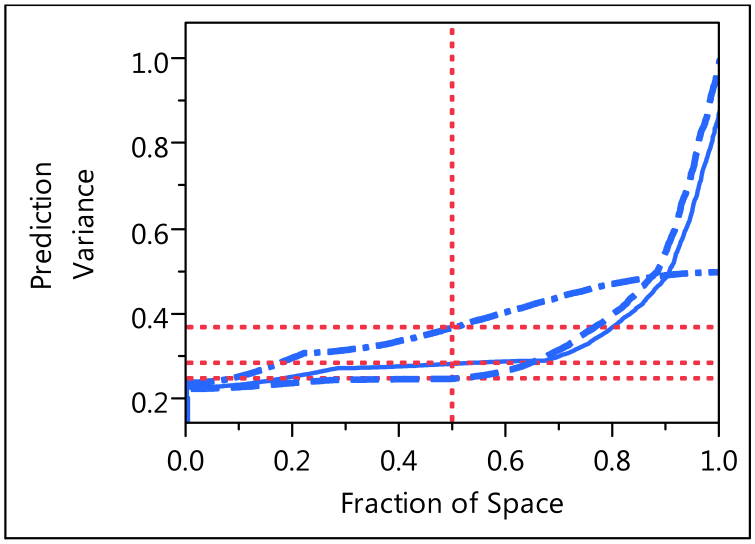


Figure 4: FDS plot comparing the D-optimal design and two I-optimal design for Scenario 2.2. The solid line represents the four-point I-optimal design. The dashed line represents the three-point I-optimal design, and the dash-dotted line represents the D-optimal design.

under consideration. We therefore investigated what the I-optimal design would be if we restricted our attention to the points $(x_1, x_2) = (0.25, 0.75)$, $(x_1, x_2) = (0.375, 0.625)$ and $(x_1, x_2) = (0.5, 0.5)$ on the constrained design region. The I-optimal design would then involve two runs at $(x_1, x_2) = (0.25, 0.75)$, four runs at $(x_1, x_2) = (0.375, 0.625)$ and one run at $(x_1, x_2) = (0.5, 0.5)$. The I-optimality criterion value of that design amounts to 0.3316, so that the I-efficiency of the three-point design relative to the four-point design is 99.79%. The loss in I-efficiency caused by the restriction to the points of the simplex-lattice design is therefore minor in Scenario 2.2.

Figure 4 visualizes the performance of the D-optimal design and the four-point and three-point I-optimal designs for the second-order model by means of a FDS plot. The plot shows that the I-optimal designs produce smaller prediction variances than the D-optimal design for about 90% of the experimental region. The two I-optimal designs exhibit a similar performance.

6.2 Three ingredients

6.2.1 Three unconstrained ingredients

Suppose that we have three ingredients and that there are no lower bounds on the ingredients proportions to be used. The first scenarios we consider (Scenarios 3.1.1 and 3.1.2) assume that the available stock of ingredient 1 is 1.5 kg, the stock of the ingredient 2 is 3 kg, and the stock of the ingredient 3 is also 3 kg. The next scenarios (Scenarios 3.2.1 and 3.2.2) assume that the available stocks are 4 kg for ingredients 1 and 2, and 5 kg for ingredient 3.

Assuming a first-order model in Scenario 3.1.1 results in a D-optimal design that has three distinct design points and two I-optimal designs that have four distinct design points. The D- and I-optimal designs all involve seven runs. The D-optimal design and the I-optimal designs are shown in Figure 5. In the figure, the design points' labels represent the number of times these points appear in the optimal design. Figure 5(b) shows the I-optimal design involving the design point $(x_1, x_2, x_3) = (1, 0, 0)$, the binary mixture $(x_1, x_2, x_3) = (0.5, 0.5, 0)$, two replicates of the design point $(x_1, x_2, x_3) = (0, 1, 0)$ and three replicates of the design point $(x_1, x_2, x_3) = (0, 0, 1)$. The alternative I-optimal design, shown in Figure 5(c), switches the roles of x_2 and x_3, and thus has one run at the design points $(x_1, x_2, x_3) = (1, 0, 0)$ and $(x_1, x_2, x_3) = (0.5, 0, 0.5)$, two replicates of the design point $(x_1, x_2, x_3) = (0, 0, 1)$ and three replicates of the design point $(x_1, x_2, x_3) = (0, 1, 0)$. The I-efficiency of the D-optimal design relative to the I-optimal ones is 90.91%, while the D-efficiency of the I-optimal designs relative to the D-optimal one is 97.14%.

The result that the D- and I-optimal designs for Scenario 3.1.1 differ is surprising, because D- and I-optimal designs for first-order models are identical for many optimal experimental design problems in the absence of ingredient availability constraints. It is also interesting to point out that the D-optimal design does not utilize the entire stock of the most scarce ingredient (ingredient 1), whereas the I-optimal designs do.

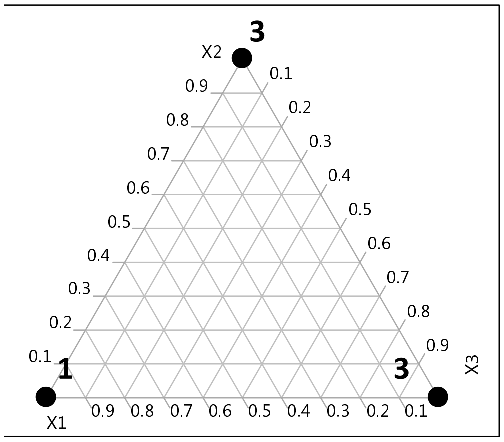
For Scenario 3.2.1, where a first-order model is assumed too, the D- and I-optimal designs coincide and involve 13 runs. Figure 6 shows the design that is both D- and I-optimal. The design involves three distinct design points, the multiplicities of which correspond to the three ingredient availabilities, 4, 4 and 5.

Even though they both involve six distinct design points, the D- and I-optimal designs for the second-order model in Scenario 3.1.2 are different from each other. This can be seen from Figures 7(a) and 7(b), which show the D-optimal design and the I-optimal design, respectively. Unlike the D-optimal design, the I-optimal design does not involve the design point $(x_1, x_2, x_3) = (1, 0, 0)$. Instead, it involves a ternary mixture, i.e. the design point $(x_1, x_2, x_3) = (0.8, 0.1, 0.1)$ which uses all three ingredients. Additionally, the I-optimal design involves the design points $(x_1, x_2, x_3) = (0.35, 0.65, 0)$ and $(x_1, x_2, x_3) = (0.35, 0, 0.65)$, which are not included in the D-optimal design (which includes the binary mixtures $(x_1, x_2, x_3) = (0.25, 0.75, 0)$ and $(x_1, x_2, x_3) = (0.25, 0, 0.75)$ instead). In any case, neither the D-optimal design nor the I-optimal design use the six points of the $\{3, 2\}$ simplex-lattice design. It is also interesting to point out that both optimal designs require 1.5 kg of the first ingredient, 2.75 kg of the second ingredient, and 2.75 kg of the third ingredient. So, both design utilize the entire available stock of the first ingredient, which has the smallest availability. The I-optimality criterion value of the D-optimal design is 0.9247 whereas that of the I-optimal design is 0.6700. Therefore, the I-efficiency of the D-optimal design is only 72.46% compared to the I-optimal design. The D-efficiency of the I-optimal design relative to the D-optimal one is 93.42%.

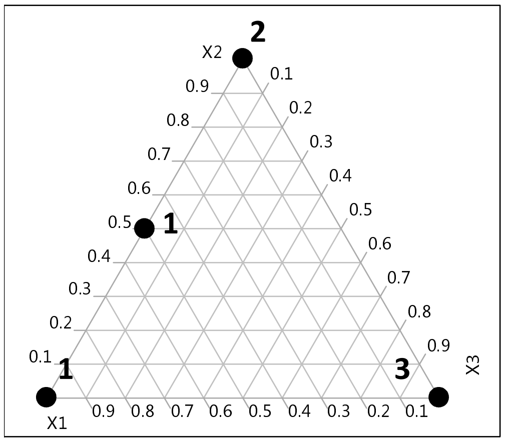
In Scenario 3.2.2, the D-optimal design involves the six points of the $\{3, 2\}$ simplex-lattice design, as shown in Figure 8(a). The design has two replicates at the midpoints of the edges of the experimental region, two replicates at the vertices $(x_1, x_2, x_3) = (1, 0, 0)$ and $(x_1, x_2, x_3) = (0, 1, 0)$, and three replicates at the vertex $(x_1, x_2, x_3) = (0, 0, 1)$. The D-optimal designs' I-optimality criterion value equals 0.3111.

The I-optimal design for Scenario 3.2.2 has eight distinct points, six of which correspond to the points of the $\{3, 2\}$ simplex-lattice design. The two additional points involve ternary mixtures, $(x_1, x_2, x_3) = (0.45, 0.05, 0.5)$ and $(x_1, x_2, x_3) = (0.05, 0.45, 0.5)$. The design is shown in Figure 8(b).

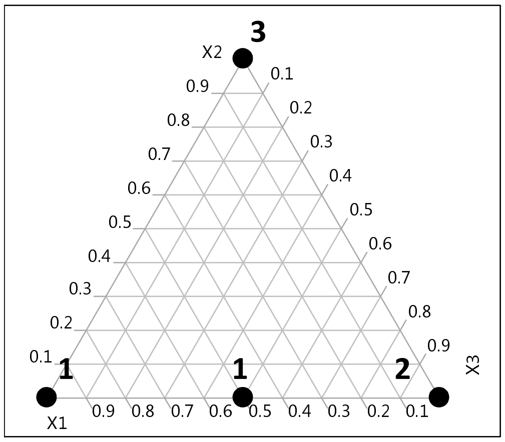
The I-optimality criterion value of the I-optimal design is 0.2603, so that the D-optimal design is only 83.68% I-efficient compared to the I-optimal design. The I-optimal design is 89.07% D-efficient. Furthermore, each optimal design requires 4 kg of the first ingredient, 4 kg of the second ingredient, and 5 kg of the third ingredient. The I-optimal design puts substantially less weight on the vertices of the experimental region



(a) D-optimal design



(b) First I-optimal design



(c) Second I-optimal design

Figure 5: The D-optimal design and the I-optimal designs for Scenario 3.1.1.

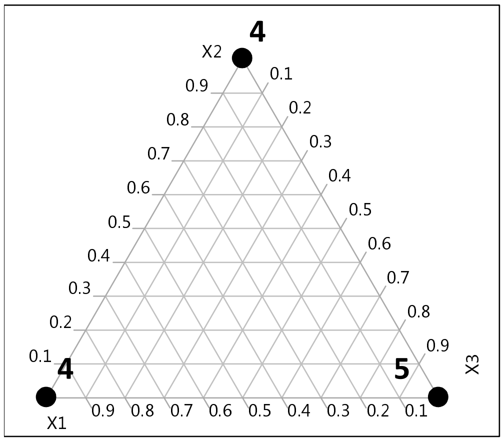


Figure 6: D- and I-optimal design for Scenario 3.2.1.

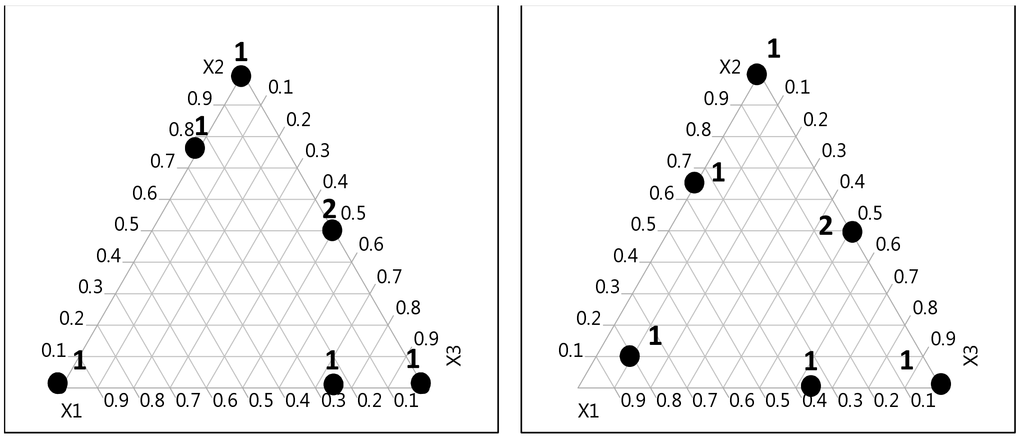

(a) D-optimal design (b) I-optimal design

Figure 7: D- and I-optimal designs for Scenario 3.1.2.

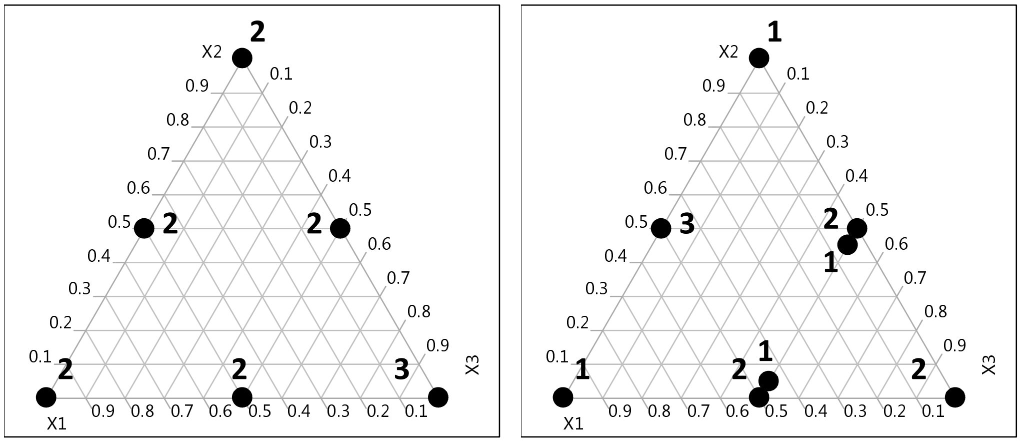

(a) D-optimal design (b) I-optimal design

Figure 8: D- and I-optimal designs for Scenario 3.2.2.

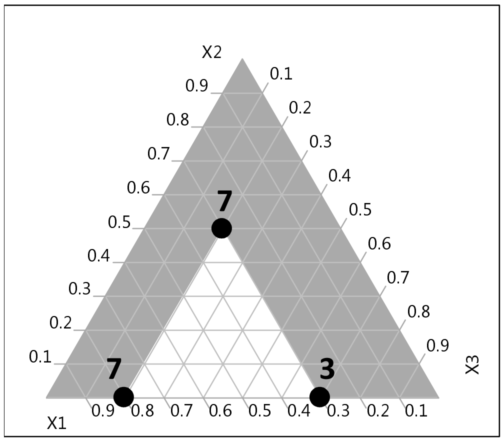


Figure 9: D- and I-optimal design for Scenario 3.3.1

than the D-optimal design. This is in line with the literature on D- and I-optimal mixture designs in the absence of ingredient availability constraints.

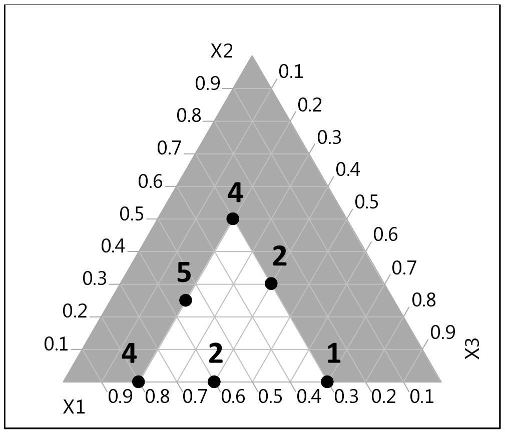
6.2.2 Three ingredients with lower bounds

Figure 9 shows the D- and I-optimal designs for Scenario 3.3.1, involving a first-order model, lower bounds of 0.3 kg and 0.2 kg for ingredient 1 and ingredient 3, respectively, and availabilities of 10.2 kg, 4 kg and 4.9 kg for ingredients 1, 2 and 3, respectively. The white triangle in Figure 9 shows the constrained experimental region, which takes into account the lower bounds for ingredients 1 and 3. The larger grey triangle represents the experimental region in the event there are no lower bounds.

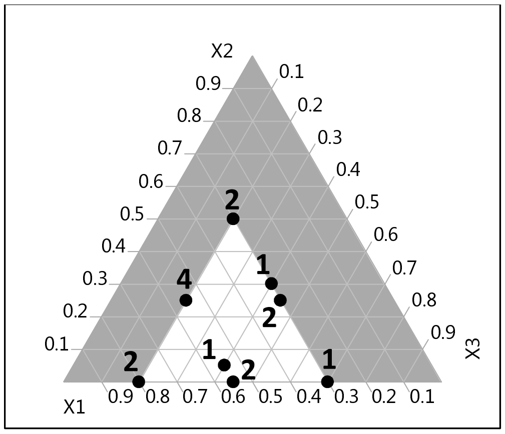
The D- and I-optimal designs for Scenario 3.3.1 coincide and consist of 17 runs at three distinct design points. The three design points are the vertices of the constrained design region, $(x_1, x_2, x_3) = (0.8, 0, 0.2)$, $(x_1, x_2, x_3) = (0.3, 0.5, 0.2)$ and $(x_1, x_2, x_3) = (0.3, 0, 0.7)$. The first two of these points are replicated seven times, while the last design point is used three times only.

A remarkable fact about the D- and I-optimal design for Scenario 3.3.1 is that it requires only 3.5 kg of the most scarce ingredient, i.e. ingredient 2, which has an availability of 4 kg. Ingredient 3, which has an availability of 4.9 kg, is the only ingredient whose entire stock is used. This is counterintuitive at first sight. However, ingredient 2 does not have a lower bound on its proportion, which implies that many experimental runs can be performed without emptying the stock of that ingredient. Ingredient 3, however, does have a lower bound of 0.2 on its proportion. This implies that every single experimental run requires some of ingredient 3. For this reason, the limited availability of ingredient 3 is more problematic than that of ingredient 2. In general, it is the ingredients for which the availability is R_i small and the lower bound L_i is large that will limit the number of runs that can be conducted, and, hence, limit the information content of the experimental design.

The D-optimal design for Scenario 3.3.2 (which assumes a second-order model) differs from the I-optimal design. The I-optimal design consists of 15 runs at 8 distinct design points, whereas the D-optimal design involves 18 runs at 6 distinct design points. The D-optimal design is shown in Figure 10(a), while the



(a) D-optimal design



(b) I-optimal design

Figure 10: Optimal designs for Scenario 3.3.2

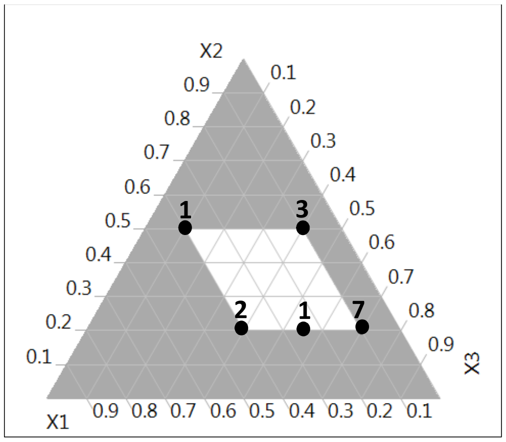
I-optimal design is shown in Figure 10(b). The D-optimal design has more replicates at two of the three vertices of the experimental region than the I-optimal design. The I-optimal design involves a larger number of design points other than the vertices. It therefore puts less emphasis on the vertices of the experimental region than the D-optimal design. The I-efficiency of the D-optimal design relative to the I-optimal design is 88.12%. The D-efficiency of the I-optimal design is 81.93%.

6.2.3 Three ingredients with lower and upper bounds

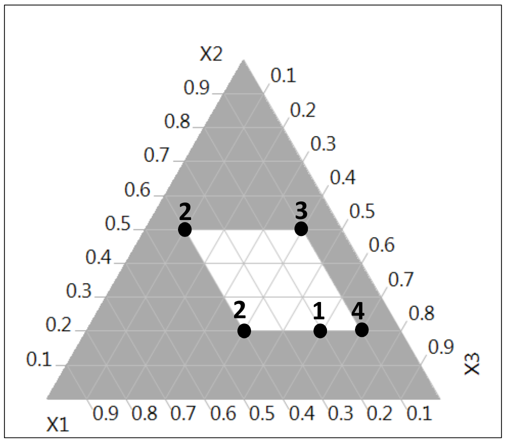
Scenario 3.4.1 and 3.4.2 involve lower and upper bounds on the ingredient proportions. The proportion of the first ingredient has to lie between 0.1 and 0.4. The proportion of the second ingredient has to lie between 0.2 and 0.5. Finally, the proportion of the third ingredient has to lie between 0.1 and 0.7. The resulting constrained design region is a parallelogram, as shown by the white area in Figures 11–14. The available stock of ingredients 1, 2 and 3 is 2.5 kg, 4 kg, and 10 kg, respectively. We based the candidate set for our algorithm on the {3,20} simplex-lattice design. After removing the points that did not satisfy the lower and upper bounds, 49 candidate points remained.

Figure 11(a) shows the D-optimal design for the first-order model (Scenario 3.4.1). Four design points correspond to the vertices, while another design point lies in the middle of an edge. In total, the design involves five distinct design points and 14 runs. The number of replicates differs from design point to design point. The D-optimal design has unbalanced replicates because the stock of the third ingredient is substantially larger than that of the two other ingredients. The design requires 2.45 kg of the first ingredient, 4 kg of the second ingredient, and 7.55 kg of the third ingredient. The D-optimality criterion value of the design is 0.7695, while its I-optimality criterion value of the design is 0.1781.

The I-optimal design for the first-order model in Scenario 3.4.1 differs from the D-optimal design. Figure 11(b) shows the composition of the I-optimal design. The design consists of five distinct design points and 12 runs only. Four design points correspond to the vertices and one other design point lies on an edge, namely the point $(x_1, x_2, x_3) = (0.2, 0.2, 0.6)$. In the I-optimal design, the replicates are distributed more evenly across the design points than in the D-optimal design. The I-optimality criterion value of the



(a) The original D-optimal design



(b) The original I-optimal design

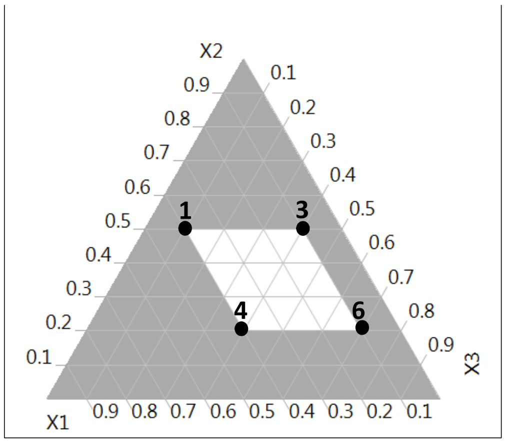
Figure 11: Original D- and I-optimal designs for Scenario 3.4.1.

I-optimal design is 0.1543, while the D-optimality criterion value of the design is 0.7182. The D-efficiency of the I-optimal design compared to the D-optimal design therefore is 97.73%, while the I-efficiency of the D-optimal design compared to the I-optimal design is 86.68%.

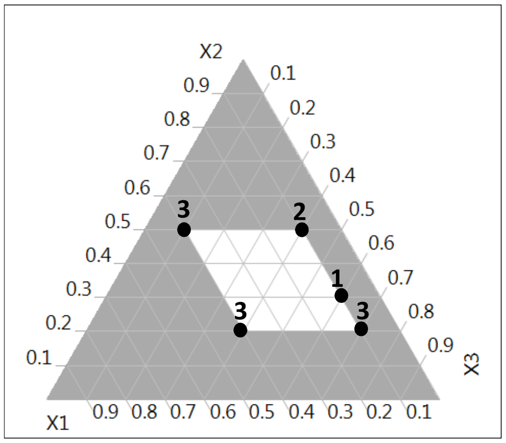
The fact that the I-optimal design in Figure 11(b) uses the entire stock of the first ingredient might inspire the experimenters to order an additional amount of that ingredient. In case an extra 0.5 kg is ordered, the total availability of ingredient 1 becomes 3 kg. Figure 12(a) shows the composition of the corresponding new D-optimal design and Figure 12(b) shows the composition of the new I-optimal design. The new D-optimal design involves four distinct design points, while the new I-optimal design involves five distinct points. The replicates in the new I-optimal design are more evenly spread than in the new D-optimal design. The D-efficiency of the new I-optimal design compared to the new D-optimal design is 91.76%, whereas the I-efficiency of the D-optimal design compared to the I-optimal design is 93.44%.

The D-optimal design for the second-order model in Scenario 3.4.2 is shown in Figure 13(a) and involves seven distinct points and 13 runs. Four design points are vertices, two design points lie in the middle of an edge and one design point lies in the middle of the experimental region. The bottom right vertex has the largest number of replicates of all design points. The D-optimality criterion value of the design is 1.488E-9 and its I-optimality criterion value is 0.5079. The design requires 2.5 kg of ingredient 1, 3.95 kg of ingredient 2, and 6.55 kg of ingredient 3.

Unlike the D-optimal design, the I-optimal design for the second-order model in Scenario 3.4.2 involves 13 runs and nine distinct design points. Figure 13(b) shows the composition of the I-optimal design. From an optimal design of experiments perspective, a few things are unusual about the I-optimal design. First, the top left vertex of the experimental region, with coordinates $(x_1, x_2, x_3) = (0.4, 0.5, 0.1)$, does not appear in the design. Instead, the point $(x_1, x_2, x_3) = (0.4, 0.45, 0.15)$ is a design point. One point, $(x_1, x_2, x_3) = (0.25, 0.3, 0.45)$, is right next to the overall centroid of the parallelogram-shaped region, and another point, $(x_1, x_2, x_3) = (0.2, 0.2, 0.6)$, is to the right of the middle of the experimental region's lower edge. The I-optimal design requires 2.5 kg of the first ingredient, 3.95 kg of the second ingredient, and 5.55 kg of the third ingredient.

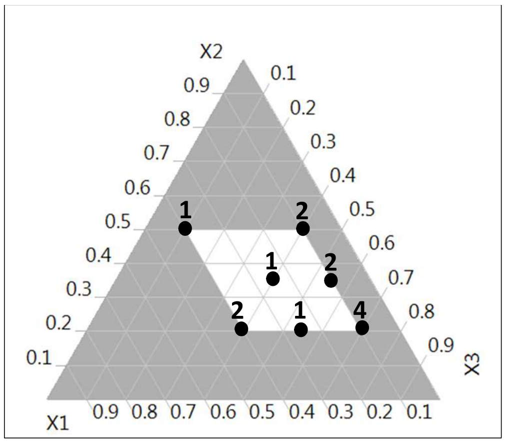


(a) The new D-optimal design

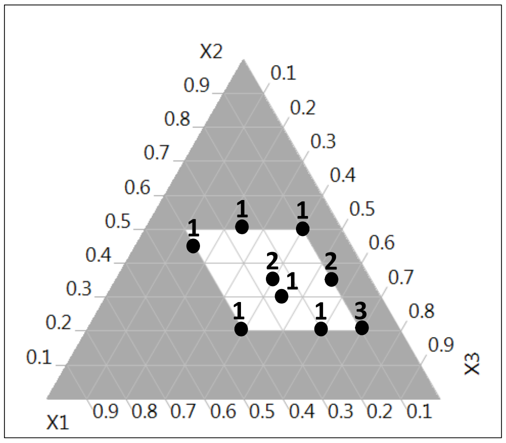


(b) The new I-optimal design

Figure 12: New D- and I-optimal designs for Scenario 3.4.1 in case the availability of the first ingredient is increased.

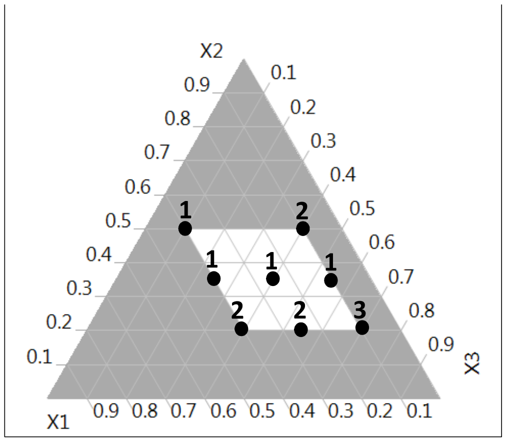


(a) The original D-optimal design

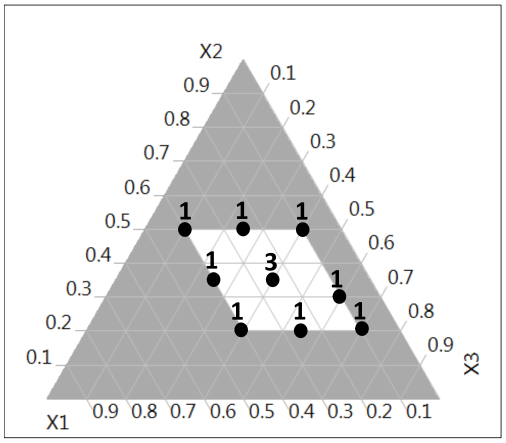


(b) The original I-optimal design

Figure 13: Original D- and I- optimal designs for Scenario 3.4.2.



(a) The new D-optimal design



(b) The new I-optimal design

Figure 14: New D- and I-optimal designs for Scenario 3.4.2 in case the availability of the first ingredient is increased.

The I-optimality criterion value of the design is 0.3492 and its D-optimality criterion value is 7.1720E-10. For the second-order model, the D-efficiency of the I-optimal design compared to the D-optimal one therefore is 88.55%, whereas the I-efficiency of the D-optimal design compared to the I-optimal one is 68.74%.

Increasing the availability of ingredient 1 by 0.5 kg leads to a new D-optimal design consisting of 13 runs with eight distinct design points. As a result, there is an additional design point in the new D-optimal design. Figure 14(a) shows the composition of the new D-optimal design. The design requires 2.95 kg of ingredient 1, 3.95 kg of ingredient 2, and 6.1 kg of ingredient 3. The D-optimality criterion value of the design is 2.3016E-09 and its I-optimality criterion value is 0.4233. The new D-optimal design is 7.54% more D-efficient than the original D-optimal design.

The new I-optimal design for the modified Scenario 3.4.2 involves 12 runs and nine distinct design points, and is shown in Figure 14(b). The number of distinct design points is the same as in the original I-optimal design in Figure 13(b), but the locations of four of the nine design points differ. One change is that the top left vertex of the design region, with coordinates $(x_1, x_2, x_3) = (0.4, 0.5, 0.1)$, now does appear in the design. The new I-optimal design requires 3 kg of the first ingredient, 4 kg of the second ingredient, and 5 kg of the third ingredient. So, the design exhausts the entire stocks of the first and the second ingredient.

The I-optimality criterion value of the new I-optimal design is 0.3101, whereas the D-optimality criterion value is 1.066E-9. As a result, the I-efficiency of the original I-optimal design relative to the new I-optimal design is 88.80%. Clearly, the new I-optimal design is substantially better than the previous design in terms of average variance of prediction. The fraction of design space plot in Figure 15 shows that the new I-optimal design (represented by the dashed line) is superior to the original I-optimal design (represented by the solid line) in terms of prediction variance in most of the experimental region. The median prediction variances for the two designs are visualized by means of the horizontal dashed lines and equal 0.2705 for the new I-optimal design and 0.2900 for the original I-optimal design. Furthermore, the D-efficiency of the new I-optimal design compared to the new D-optimal design is 87.96% and the I-efficiency of the new D-optimal design compared to the new I-optimal design is 73.26%.

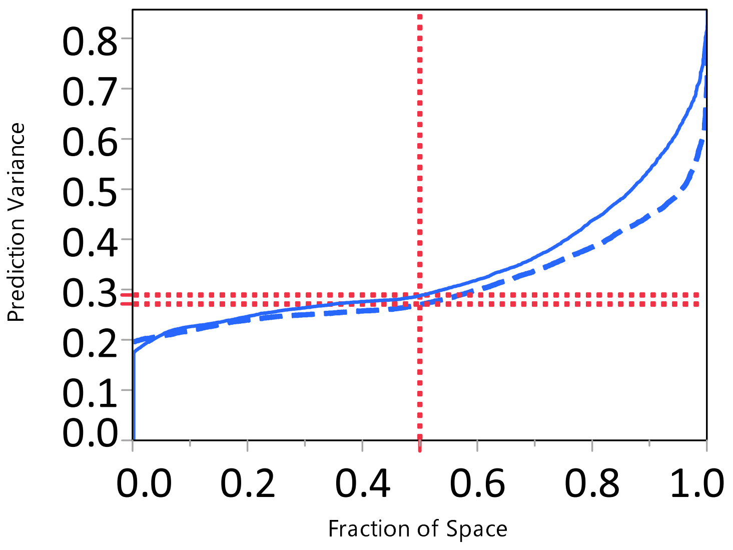


Figure 15: Fraction of design space plot comparing the original and the new I-optimal design for Scenario 3.4.2. The solid line represents the original I-optimal design, while the dashed line represents the new I-optimal design after increasing the availability of ingredient 1.

6.3 Four ingredients

Scenarios 4.1 and 4.2 involve four ingredients with availabilities 2.5 kg, 6 kg, 3 kg and 7 kg. Every experimental run requires at least 0.2 kg of ingredient 1, 0.1 kg of ingredient 2, 0.1 kg of ingredient 3, and 0.2 kg of ingredient 4.

For Scenario 4.1, in which a first-order model is assumed, the I-optimal design is slightly different from the D-optimal design, even though both designs have the same number of runs, 10. All the runs of the D-optimal design are performed at the vertices of the experimental region, while the I-optimal design involves one design point that is not a vertex. This can be seen by comparing Tables 3 and 4, which show the D- and I-optimal designs, respectively. The tables show the proportions to be used of the original ingredients, as well as the proportions of the pseudocomponents. The former are useful to verify that the availability constraints are satisfied, while the latter are useful to evaluate the geometry of the designs. Whenever the proportion of a pseudocomponent equals one, the corresponding design point is a vertex. That is the case for all points in the D-optimal design and for all but one point of the I-optimal design. The I-optimal design point that is not a vertex lies on one of the edges of the tetrahedron-shaped constrained experimental region.

The I-optimality criterion value of the D-optimal design is 0.2, whereas that of the I-optimal design is 0.19457. Therefore, the I-efficiency of the D-optimal design is 97.29% compared to the I-optimal design. The D-efficiency of the I-optimal design is also 97.29%.

As for Scenario 4.1, the D- and I-optimal designs differ for Scenario 4.2, where a second-order model is assumed. The designs, which each involve ten runs, are shown in Tables 5 and 6. Remarkably, the two designs only involve three of the four vertices of the experimental region. The missing vertex is the point $(x_1, x_2, x_3, x_4) = (0.6, 0.1, 0.1, 0.2)$, which corresponds to a mixture with the maximum proportion of x_1

Table 3: D-optimal design for Scenario 4.1

Number	Original proportion				Pseudocomponents				Replicates	Design point
	x_1	x_2	x_3	x_4	x_1^*	x_2^*	x_3^*	x_4^*		
1	0.2	0.1	0.1	0.6	0	0	0	1	3	Vertex
2	0.2	0.1	0.5	0.2	0	0	1	0	3	Vertex
3	0.2	0.5	0.1	0.2	0	1	0	0	3	Vertex
4	0.6	0.1	0.1	0.2	1	0	0	0	1	Vertex
Total	2.4	2.2	2.2	3.2					10	
Availability	2.5	6	3	7					18.5	

Table 4: I-optimal design for Scenario 4.1

Number	Original proportion				Pseudocomponents				Replicates	Design point
	x_1	x_2	x_3	x_4	x_1^*	x_2^*	x_3^*	x_4^*		
1	0.2	0.1	0.1	0.6	0	0	0	1	3	Vertex
2	0.2	0.1	0.5	0.2	0	0	1	0	2	Vertex
3	0.2	0.5	0.1	0.2	0	1	0	0	3	Vertex
4	0.3	0.1	0.4	0.2	0.25	0	0.75	0	1	Edge
5	0.6	0.1	0.1	0.2	1	0	0	0	1	Vertex
Total	2.5	2.2	2.1	3.2					10	
Availability	2.5	6	3	7					18.5	

possible and the minimum proportions of x_2, x_3 and x_4 possible. This vertex is missing in both designs because x_1 is the most scarce ingredient and, in addition, it has the largest lower bound for its proportion. The remaining seven points of the D-optimal design all lie on the edges of the tetrahedron-shaped experimental region. The I-optimal design only has four design points that lie on an edge, and also involves two points that lie on a face of the experimental region and one point in the region's interior.

The I-optimality criterion value of the D-optimal design is 1.5568, whereas it is 1.0817 for the I-optimal design. Hence, the I-efficiency of the D-optimal design is only 69.48% compared to the I-optimal design. The D-efficiency of the I-optimal design is 91.03%.

Note that the availabilities in Scenario 4.2 only just suffice to create a design with ten runs, which is the minimum required number of runs to estimate the second-order model in the four ingredients. Also, the ten runs are conducted at ten distinct design points, to ensure estimability of the second-order model. Therefore, there are no degrees of freedom left to estimate the error variance.

The FDS plot in Figure 16 shows that the I-optimal design produces better prediction variances than the D-optimal design in about 80% of the experimental region. A remarkable fact is that the maximum prediction variance is much larger than the average prediction variance, both for the D- and for the I-optimal design. This is due to the fact that neither design includes a run at the vertex $(x_1, x_2, x_3, x_4) = (0.6, 0.1, 0.1, 0.2)$, and, as a result of that, the prediction variance in that point is large. This large prediction variance is a direct result from the limited availability of ingredient 1, which makes the use of design points involving large quantities of that ingredient impossible.

The large maximum prediction variances of the D- and I-optimal designs and the fact that their number

Table 5: D-optimal design for Scenario 4.2

Number	Original proportion				Pseudocomponents				Design point
	x_1	x_2	x_3	x_4	x_1^*	x_2^*	x_3^*	x_4^*	
1	0.2	0.1	0.1	0.6	0	0	0	1	Vertex
2	0.2	0.1	0.3	0.4	0	0	0.5	0.5	Edge
3	0.2	0.1	0.5	0.2	0	0	1	0	Vertex
4	0.2	0.3	0.1	0.4	0	0.5	0	0.5	Edge
5	0.2	0.3	0.3	0.2	0	0.5	0.5	0	Edge
6	0.2	0.5	0.1	0.2	0	1	0	0	Vertex
7	0.25	0.1	0.1	0.55	0.125	0	0	0.875	Edge
8	0.3	0.1	0.4	0.2	0.25	0	0.75	0	Edge
9	0.3	0.4	0.1	0.2	0.25	0.75	0	0	Edge
10	0.45	0.1	0.1	0.35	0.625	0	0	0.375	Edge
Total	2.5	2.1	2.1	3.3					10
Availability	2.5	6	3	7					18.5

Table 6: I-optimal design for Scenario 4.2

Number	Original proportion				Pseudocomponents				Design point
	x_1	x_2	x_3	x_4	x_1^*	x_2^*	x_3^*	x_4^*	
1	0.2	0.1	0.1	0.6	0	0	0	1	Vertex
2	0.2	0.1	0.3	0.4	0	0	0.5	0.5	Edge
3	0.2	0.1	0.5	0.2	0	0	1	0	Vertex
4	0.2	0.3	0.1	0.4	0	0.5	0	0.5	Edge
5	0.2	0.3	0.3	0.2	0	0.5	0.5	0	Edge
6	0.2	0.5	0.1	0.2	0	1	0	0	Vertex
7	0.25	0.1	0.1	0.55	0.125	0	0	0.875	Edge
8	0.3	0.1	0.35	0.25	0.25	0	0.625	0.125	Face
9	0.3	0.35	0.1	0.25	0.25	0.625	0	0.125	Face
10	0.45	0.15	0.15	0.25	0.625	0.125	0.125	0.125	Interior
Total	2.5	2.3	2.1	3.1					10
Availability	2.5	6	3	7					18.5

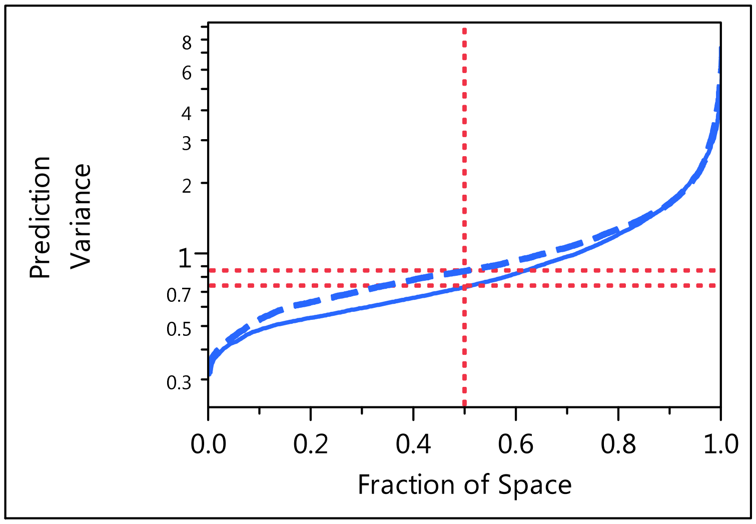


Figure 16: FDS plot for the D- and I-optimal designs in Scenario 4.2. The dashed line represents the D-optimal design, while the solid line represents the I-optimal design.

of runs equals the number of model parameters are two weaknesses, which are due to the limited availability of ingredient 1. As a matter of fact, ingredient 1 is the only ingredient whose stock is entirely used by the designs. An experimenter facing such a situation might therefore consider acquiring some additional stock of the first ingredient. In order to decide on an appropriate amount of stock to acquire, we recommend a sensitivity study in which the total availability of the first ingredient is gradually increased to study the impact on the resulting D- and I-optimal designs, the corresponding average prediction variances and the corresponding maximum prediction variances. In case several ingredients have a limited stock, the sensitivity study should involve a gradual increase of the availabilities of more than one ingredient.

Suppose, as an illustration, that 2 additional kilograms of the first ingredient and 1.5 additional kilograms of the third ingredient are acquired, resulting in total availabilities of 4.5 kg for the first ingredient, 6 kg for the second ingredient, 4.5 kg for the third ingredient, and 7 kg for the fourth ingredient. The I-optimal design we obtained for the increased availabilities is shown in Table 7. The design consists of 14 distinct design points and 17 runs. Despite the substantially larger available stock of ingredient 1, the new I-optimal design does not involve the vertex $(x_1, x_2, x_3, x_4) = (0.6, 0.1, 0.1, 0.2)$. So, rather than using one run that uses the maximum amount of ingredient 1, it is better, from an I-optimality criterion point of view, to use design points which use a smaller proportion of ingredient 1. The three other vertices of the design region are included in the design once. Ten runs are conducted at one of the edges of the experimental region, and the four final runs are performed at one of the experimental region's faces. The new I-optimal design requires 4.5 kg of the first ingredient, 3.6 kg of the second ingredient, 3.6 kg of the third ingredient, and 5.3 kg of the fourth ingredient. So, the design uses the complete additional stock of the first ingredient and some of the extra stock of the third ingredient.

The absence of the vertex $(x_1, x_2, x_3, x_4) = (0.6, 0.1, 0.1, 0.2)$ in the new I-optimal design results in a larger prediction variance at that point when compared to other parts of the experimental region. When compared to the original I-optimal design, the new design, however, leads to a maximum prediction variance

Table 7: I-optimal design involving four ingredients for Scenario 4.2, based on the new ingredient availabities.

Number	Original proportion				Pseudocomponents				Design point
	x_1	x_2	x_3	x_4	x_1^*	x_2^*	x_3^*	x_4^*	
1	0.2	0.1	0.1	0.6	0	0	0	1	Vertex
2	0.2	0.1	0.5	0.2	0	0	1	0	Vertex
3	0.2	0.5	0.1	0.2	0	1	0	0	Vertex
4	0.2	0.1	0.3	0.4	0	0	0.5	0.5	Edge
5	0.2	0.1	0.3	0.4	0	0	0.5	0.5	Edge
6	0.2	0.3	0.1	0.4	0	0.5	0	0.5	Edge
7	0.2	0.3	0.1	0.4	0	0.5	0	0.5	Edge
8	0.2	0.3	0.3	0.2	0	0.5	0.5	0	Edge
9	0.2	0.3	0.3	0.2	0	0.5	0.5	0	Edge
10	0.35	0.1	0.1	0.45	0.375	0	0	0.625	Edge
11	0.35	0.1	0.35	0.2	0.375	0	0.625	0	Edge
12	0.35	0.35	0.1	0.2	0.375	0.625	0	0	Edge
13	0.55	0.1	0.1	0.25	0.875	0	0	0.125	Edge
14	0.2	0.25	0.25	0.3	0	0.375	0.375	0.25	Face
15	0.3	0.1	0.25	0.35	0.25	0	0.375	0.375	Face
16	0.3	0.25	0.1	0.35	0.25	0.375	0	0.375	Face
17	0.3	0.25	0.25	0.2	0.25	0.375	0.375	0	Face
Total	4.5	3.6	3.6	5.3					
Availability	4.5	6	4.5	7					

that is considerably smaller at that vertex. For the original design, the prediction variance amounts to 17.84 at that point, while it is 2.33 for the new design. The average variance of prediction is 0.3090 for the new I-optimal design, compared to 1.0818 for the original I-optimal design. Figure 17 compares the fraction of design space plots for the two I-optimum designs in Scenario 4.2. Clearly, the new design (represented by the dashed line) performs much better than the original design (represented by the solid line). It is clear that the additional availability of certain ingredients has a major impact on the maximum, the median and the average prediction variance.

Unlike the original I-optimal design in Table 6, the new I-optimal design in Table 7 does allow the estimation of the error variance, and, because three design points are used twice, even a lack-of-fit test.

6.4 Six ingredients

The final scenarios we consider in this paper are Scenarios 6.1 and 6.2, involving six ingredients. The minimum proportions for the six ingredients at each experimental run are 0.05, 0.1, 0.1, 0.1, 0.2 and 0.2. The available stocks for the six ingredients are 4, 4, 5, 5, 8 and 16 kg.

For Scenario 6.1, the D- and I-optimal designs differ in the number of distinct design points: the D-optimal design involves the six vertices of the experimental region only, whereas the I-optimal design uses a design point at an edge of the experimental region in addition to the six vertices. The two designs also differ in the number of runs: the D-optimal design has 35 runs, while the I-optimal design has 32 runs. The optimal design points of the two designs are shown in Tables 8 and 9, along with the number of replicates at each of them. The largest number of replicates for a given vertex in the D-optimal design is nine, while

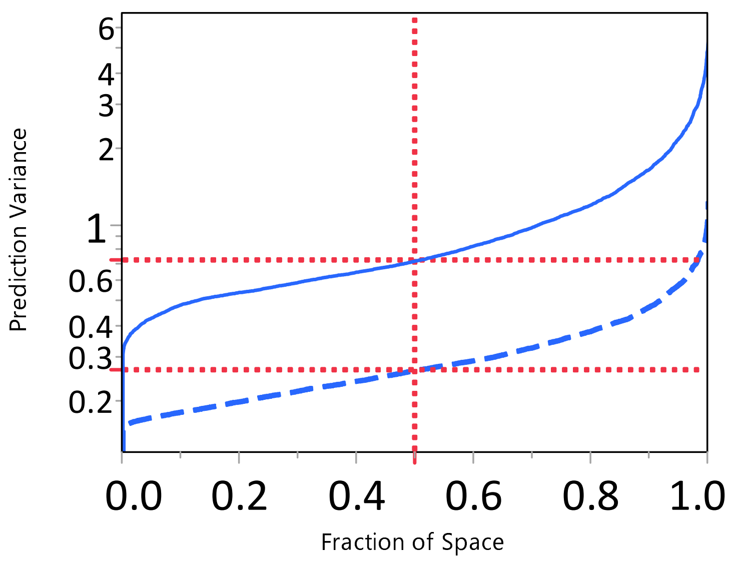


Figure 17: Fraction of design space plot comparing the original and the new I-optimal design for Scenario 4.2. The solid line represents the original I-optimal design, while the dashed line represents the new I-optimal design exploiting the increased availabilities of ingredients 1 and 3.

this number is only seven for the I-optimal design. The smallest number of replicates of a given vertex in the D-optimal design is two, while this number is three in the I-optimal design. As a result, the I-optimal design is more balanced in the number of replicates than the D-optimal design.

The I-optimal design is more efficient than the D-optimal design in terms of prediction, even though the I-optimal design has a smaller number of runs than the D-optimal design. The I-efficiency of the D-optimal design relative to the I-optimal one is 90.77%. The D-efficiency of the I-optimal design is 97.17%.

Figure 18 shows a FDS plot comparing the predictive performance of the D-optimal design with that of the I-optimal design. It can be seen that the performance of the two designs is similar for about 50% of the experimental region. However, the D-optimal design leads to larger prediction variances in many cases.

Table 8: D-optimal design for Scenario 6.1

Number	Original proportion						Pseudocomponents						Repl.
	x_1	x_2	x_3	x_4	x_5	x_6	x_1^*	x_2^*	x_3^*	x_4^*	x_5^*	x_6^*	
1	0.05	0.1	0.1	0.1	0.2	0.45	0	0	0	0	0	1	9
2	0.05	0.1	0.1	0.1	0.45	0.2	0	0	0	0	1	0	4
3	0.05	0.1	0.1	0.35	0.2	0.2	0	0	0	1	0	0	6
4	0.05	0.1	0.35	0.1	0.2	0.2	0	0	1	0	0	0	6
5	0.05	0.35	0.1	0.1	0.2	0.2	0	1	0	0	0	0	2
6	0.3	0.1	0.1	0.1	0.2	0.2	1	0	0	0	0	0	8
Total	3.75	4	5	5	8	9.25							35
Availability	4	4	5	5	8	16							42

Table 9: I-optimal design for Scenario 6.1

Number	Original proportion						Pseudocomponents						Repl.
	x_1	x_2	x_3	x_4	x_5	x_6	x_1^*	x_2^*	x_3^*	x_4^*	x_5^*	x_6^*	
1	0.05	0.1	0.1	0.1	0.2	0.45	0	0	0	0	0	1	6
2	0.05	0.1	0.1	0.1	0.45	0.2	0	0	0	0	1	0	6
3	0.05	0.1	0.1	0.35	0.2	0.2	0	0	0	1	0	0	7
4	0.05	0.1	0.35	0.1	0.2	0.2	0	0	1	0	0	0	5
5	0.05	0.35	0.1	0.1	0.2	0.2	0	1	0	0	0	0	3
6	0.05	0.15	0.3	0.1	0.2	0.2	0	0.8	0.2	0	0	0	1
7	0.3	0.1	0.1	0.1	0.2	0.2	1	0	0	0	0	0	5
Total	2.85	4	4.65	4.7	7.9	7.9							32
Availability	4	4	5	5	8	16							42

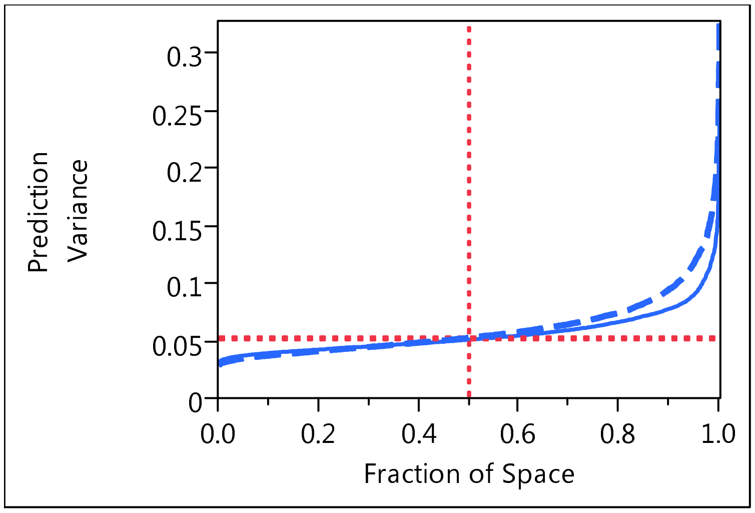


Figure 18: FDS plot comparing the D-optimal design with the I-optimal design for Scenario 6.1. The dashed line represents the D-optimal design, while the solid line represents the I-optimal design.

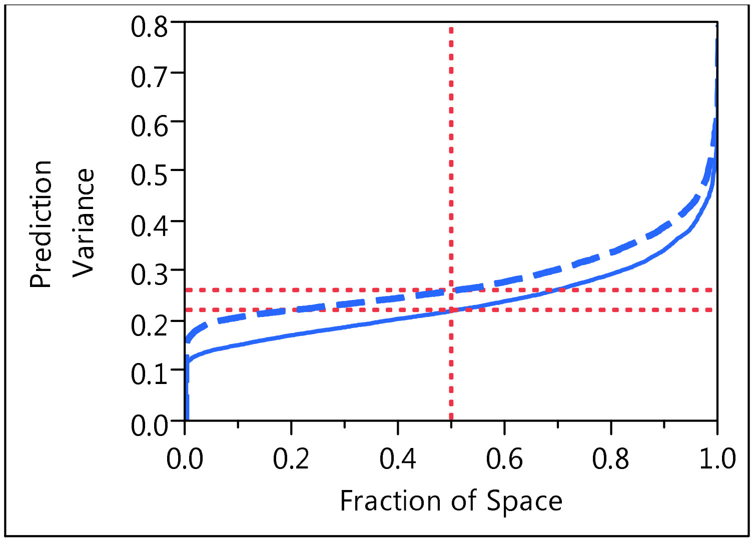


Figure 19: FDS plot comparing the D- and I-optimal designs for Scenario 6.2. The dashed line represents the D-optimal design, while the solid line represents the I-optimal design.

The D-optimal design for Scenario 6.2 has 32 runs, while the I-optimal design only involves 31 runs. The D-optimal design, shown in Table 10, only consists of points that either are vertices or lie on the edges of the experimental region. One of the vertices, the first one, is duplicated in that design. The I-optimal design, shown in Table 11, also involves design points that are not vertices and do not lie on the edges of the experimental region. The design involves six design points that correspond to the vertices of the experimental design, 17 design points that lie on an edge, five design points that lie on a face, and three other design points. None of the design points are replicated in the I-optimal design.

The I-efficiency of D-optimal design is 84.99% relative to the I-optimal one, whereas the D-efficiency of the I-optimal design is only 67.88%. Figure 19 compares the predictive performance of the D- and I-optimal designs by means of a FDS plot. The curve for the I-optimal design lies substantially lower than that for the D-optimal design, over nearly the entire design region, indicating a superior performance of the I-optimal design in terms of prediction variance.

7 Conclusion

In this article, we introduced a new type of problem in the field of optimal design of experiments, i.e. a problem involving limited availabilities of ingredients in a mixture. To solve instances of the new optimal design problem, we used a new variable neighborhood descent algorithm. The algorithm has the advantage that it generally does not get stuck in local optima as often as simpler algorithms that are commonly used in optimal experimental design (such as point-exchange and coordinate-exchange algorithms). Moreover, the algorithm is more intuitive than genetic algorithms and simulated annealing, for instance.

We demonstrated the potential of the variable neighborhood descent algorithm to generate D- and I-optimal designs by means of several examples. The resulting optimal designs are very different from the

Table 10: D-optimal design for Scenario 6.2

Number	Original ingredients						Pseudocomponents						Design point
	x_1	x_2	x_3	x_4	x_5	x_6	x_1^*	x_2^*	x_3^*	x_4^*	x_5^*	x_6^*	
1	0.05	0.1	0.1	0.1	0.2	0.45	0	0	0	0	0	1	Vertex
2	0.05	0.1	0.1	0.1	0.2	0.45	0	0	0	0	0	1	Vertex
3	0.05	0.1	0.1	0.1	0.3	0.35	0	0	0	0	0.4	0.6	Edge
4	0.05	0.1	0.1	0.1	0.35	0.3	0	0	0	0	0.6	0.4	Edge
5	0.05	0.1	0.1	0.1	0.45	0.2	0	0	0	0	1	0	Vertex
6	0.05	0.1	0.1	0.2	0.2	0.35	0	0	0	0.4	0	0.6	Edge
7	0.05	0.1	0.1	0.2	0.35	0.2	0	0	0	0.4	0.6	0	Edge
8	0.05	0.1	0.1	0.25	0.2	0.3	0	0	0	0.6	0	0.4	Edge
9	0.05	0.1	0.1	0.25	0.3	0.2	0	0	0	0.6	0.4	0	Edge
10	0.05	0.1	0.1	0.35	0.2	0.2	0	0	0	1	0	0	Vertex
11	0.05	0.1	0.2	0.1	0.2	0.35	0	0	0.4	0	0	0.6	Edge
12	0.05	0.1	0.2	0.1	0.35	0.2	0	0	0.4	0	0.6	0	Edge
13	0.05	0.1	0.2	0.25	0.2	0.2	0	0	0.4	0.6	0	0	Edge
14	0.05	0.1	0.25	0.1	0.2	0.3	0	0	0.6	0	0	0.4	Edge
15	0.05	0.1	0.25	0.1	0.3	0.2	0	0	0.6	0	0.4	0	Edge
16	0.05	0.1	0.25	0.2	0.2	0.2	0	0	0.6	0.4	0	0	Edge
17	0.05	0.1	0.35	0.1	0.2	0.2	0	0	1	0	0	0	Vertex
18	0.05	0.2	0.1	0.1	0.2	0.35	0	0.4	0	0	0	0.6	Edge
19	0.05	0.2	0.1	0.1	0.35	0.2	0	0.4	0	0	0.6	0	Edge
20	0.05	0.2	0.1	0.25	0.2	0.2	0	0.4	0	0.6	0	0	Edge
21	0.05	0.25	0.2	0.1	0.2	0.2	0	0.6	0.4	0	0	0	Edge
22	0.05	0.35	0.1	0.1	0.2	0.2	0	1	0	0	0	0	Vertex
23	0.15	0.1	0.1	0.1	0.2	0.35	0.4	0	0	0	0	0.6	Edge
24	0.15	0.1	0.1	0.1	0.35	0.2	0.4	0	0	0	0.6	0	Edge
25	0.15	0.1	0.1	0.25	0.2	0.2	0.4	0	0	0.6	0	0	Edge
26	0.15	0.1	0.25	0.1	0.2	0.2	0.4	0	0.6	0	0	0	Edge
27	0.2	0.1	0.1	0.1	0.2	0.3	0.6	0	0	0	0	0.4	Edge
28	0.2	0.1	0.1	0.1	0.3	0.2	0.6	0	0	0	0.4	0	Edge
29	0.2	0.1	0.1	0.2	0.2	0.2	0.6	0	0	0.4	0	0	Edge
30	0.2	0.1	0.2	0.1	0.2	0.2	0.6	0	0.4	0	0	0	Edge
31	0.2	0.2	0.1	0.1	0.2	0.2	0.6	0.4	0	0	0	0	Edge
32	0.3	0.1	0.1	0.1	0.2	0.2	1	0	0	0	0	0	Vertex
Total	3	4	4.55	4.6	7.8	8.05							32
Availability	4	4	5	5	8	16							42

Table 11: I-optimal design for Scenario 6.2

Number	Original ingredients						Pseudocomponents						Design point
	x_1	x_2	x_3	x_4	x_5	x_6	x_1^*	x_2^*	x_3^*	x_4^*	x_5^*	x_6^*	
1	0.05	0.1	0.1	0.1	0.2	0.45	0	0	0	0	0	1	Vertex
2	0.05	0.1	0.1	0.1	0.3	0.35	0	0	0	0	0.4	0.6	Edge
3	0.05	0.1	0.1	0.1	0.45	0.2	0	0	0	0	1	0	Vertex
4	0.05	0.1	0.1	0.2	0.25	0.3	0	0	0	0.4	0.2	0.4	Edge
5	0.05	0.1	0.1	0.2	0.35	0.2	0	0	0	0.4	0.6	0	Edge
6	0.05	0.1	0.1	0.25	0.2	0.3	0	0	0	0.6	0	0.4	Edge
7	0.05	0.1	0.1	0.35	0.2	0.2	0	0	0	1	0	0	Vertex
8	0.05	0.1	0.15	0.1	0.3	0.3	0	0	0.2	0	0.4	0.4	Edge
9	0.05	0.1	0.2	0.1	0.2	0.35	0	0	0.4	0	0	0.6	Edge
10	0.05	0.1	0.2	0.2	0.25	0.2	0	0	0.4	0.4	0.2	0	Face
11	0.05	0.1	0.25	0.1	0.3	0.2	0	0	0.6	0	0.4	0	Edge
12	0.05	0.1	0.25	0.2	0.2	0.2	0	0	0.6	0.4	0	0	Edge
13	0.05	0.1	0.35	0.1	0.2	0.2	0	0	1	0	0	0	Vertex
14	0.05	0.15	0.15	0.15	0.25	0.25	0	0.2	0.2	0.2	0.2	0.2	Other
15	0.05	0.2	0.1	0.1	0.2	0.35	0	0.4	0	0	0	0.6	Edge
16	0.05	0.2	0.1	0.1	0.35	0.2	0	0.4	0	0	0.6	0	Edge
17	0.05	0.2	0.1	0.25	0.2	0.2	0	0.4	0	0.6	0	0	Edge
18	0.05	0.2	0.25	0.1	0.2	0.2	0	0.4	0.6	0	0	0	Edge
19	0.05	0.35	0.1	0.1	0.2	0.2	0	1	0	0	0	0	Vertex
20	0.1	0.1	0.2	0.1	0.2	0.3	0.2	0	0.4	0	0	0.4	Face
21	0.1	0.15	0.15	0.15	0.2	0.25	0.2	0.2	0.2	0.2	0	0.2	Other
22	0.15	0.1	0.1	0.1	0.35	0.2	0.4	0	0	0	0.6	0	Edge
23	0.15	0.1	0.1	0.2	0.2	0.25	0.4	0	0	0.4	0	0.2	Face
24	0.15	0.1	0.1	0.2	0.25	0.2	0.4	0	0	0.4	0.2	0	Face
25	0.15	0.1	0.15	0.2	0.2	0.2	0.4	0	0.2	0.4	0	0	Face
26	0.15	0.1	0.2	0.1	0.25	0.2	0.4	0	0.4	0	0.2	0	Face
27	0.15	0.15	0.1	0.1	0.25	0.25	0.4	0.2	0	0	0.2	0.2	Other
28	0.2	0.1	0.1	0.1	0.2	0.3	0.6	0	0	0	0	0.4	Edge
29	0.2	0.1	0.2	0.1	0.2	0.2	0.6	0	0.4	0	0	0	Edge
30	0.2	0.2	0.1	0.1	0.2	0.2	0.6	0.4	0	0	0	0	Edge
31	0.3	0.1	0.1	0.1	0.2	0.2	1	0	0	0	0	0	Vertex
Total	2.95	4	4.5	4.45	7.5	7.6							31
Availability	4	4	5	5	8	16							42

optimal designs constructed in the literature under the assumption that there are no restrictions on the ingredient usage. For instance, the new designs often involve unequal replication of the different design points. Design points which use a large proportion of a scarce ingredient are generally replicated less frequently than points which use a small proportion of that ingredient. In some cases, experimenters may want to explore alternative designs that replicate the design points as evenly as possible across the experimental region. This may require sacrificing several runs involving a little bit of one scarce ingredient in exchange for one run that uses a lot of the scarce ingredient. A systematic exploration of this approach would be a useful topic for future research.

Another feature of optimal mixture designs in the presence of ingredient availability constraints is that the distribution of design points across the experimental region depends on the lower bounds of the ingredients' proportions. This is not the case in the absence of ingredient availability constraints, where the distribution of points is entirely driven by the shape of the experimental region and not by the exact location of the lower bounds on the proportions.

In most cases studied in this paper, the D-optimal design perfomed more poorly in terms of I-optimality than the I-optimal design performed in terms of D-optimality. This is in line with the results of Hardin and Sloane (1993), Goos and Jones (2011) and Jones and Goos (2012). In one case, however, the performance of the I-optimal design in terms of the D-optimality criterion was much poorer than the performance of the D-optimal design in terms of the I-optimality criterion. This demonstrates that the presence of ingredient availability constraints results in a substantially different optimal design problem, where insights acquired in the absence of ingredient availability constraints may no longer apply.

Acknowledgement

The first author was financially supported by the General Directorat of Higher Education of the Indonesian Ministry of National Education.

Appendix. Update formulas for the four neighborhoods

The update formulas we used for the determinant and the inverse of the information matrix $\mathbf{M}$ can all be derived from the general formulas

$$|\mathbf{M}^*| = |\mathbf{M}|\,|\mathbf{D}|\,|\mathbf{D}^{-1} + \mathbf{U}\mathbf{M}^{-1}\mathbf{U}'| \tag{5}$$

and

$$\mathbf{M}^{*-1} = \mathbf{M}^{-1} - \mathbf{M}^{-1}\mathbf{U}'(\mathbf{D}^{-1} + \mathbf{U}\mathbf{M}^{-1}\mathbf{U}')^{-1}\mathbf{U}\mathbf{M}^{-1}, \tag{6}$$

where

$$\mathbf{M}^* = \mathbf{M} + \mathbf{U}'\mathbf{D}\mathbf{U}$$

is the information matrix resulting from a change to the design.

Neighborhood N_0

In neighborhood N_0, a candidate point, say $\mathbf{x}_c$, is added to the design. In that case, $\mathbf{D} = \mathbf{D}^{-1} = |\mathbf{D}| = 1$ and $\mathbf{U} = \mathbf{f}'(\mathbf{x}_c)$, so that

$$|\mathbf{M}^*| = |\mathbf{M}|(1 + \mathbf{f}'(\mathbf{x}_c)\mathbf{M}^{-1}\mathbf{f}(\mathbf{x_c}))$$

and

$$\mathbf{M}^{*-1} = \mathbf{M}^{-1} - \frac{\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_c)\mathbf{f}'(\mathbf{x}_c)\mathbf{M}^{-1}}{1 + \mathbf{f}'(\mathbf{x}_c)\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_c)}.$$

These update formulas are well known in the design of experiments literature (see, for instance, Goos (2002)).

Neighborhood N_1

In neighborhood N_1, a design point, say $\mathbf{x}_d$, is replaced by a candidate point, say $\mathbf{x}_c$. In that case,

$$\mathbf{D} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

and

$$\mathbf{U} = \begin{bmatrix} \mathbf{f}'(\mathbf{x}_c) \\ \mathbf{f}'(\mathbf{x}_d) \end{bmatrix}.$$

The resulting update formula for the determinant is also well known (see, for instance, Goos (2002)), and is given by

$$|\mathbf{M}^*| = |\mathbf{M}| \left[\{1 + \mathbf{f}'(\mathbf{x}_c)\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_c)\}\{1 - \mathbf{f}'(\mathbf{x}_d)\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_d)\} + \{1 + \mathbf{f}'(\mathbf{x}_c)\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_d)\}^2\right].$$

Note that $\mathbf{D}^{-1} = \mathbf{D}$ and $|\mathbf{D}| = -1$ for all design changes studied in neighborhood N_1, which allows a simplification of Equations (5) and (6).

Neighborhood N_2

In neighborhood N_2, a design point, say $\mathbf{x}_d$, is replaced by two candidate points, say $\mathbf{x}_{c1}$ and $\mathbf{x}_{c2}$. In that case,

$$\mathbf{U} = \begin{bmatrix} \mathbf{f}'(\mathbf{x}_d) \\ \mathbf{f}'(\mathbf{x}_{c1}) \\ \mathbf{f}'(\mathbf{x}_{c2}) \end{bmatrix}$$

and

$$\mathbf{D} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Note that, here too, $\mathbf{D}^{-1} = \mathbf{D}$ and $|\mathbf{D}| = -1$.

Neighborhood N_3

In neighborhood N_3, two design points, say $\mathbf{x}_{d1}$ and $\mathbf{x}_{d2}$, are replaced by two candidate points, say $\mathbf{x}_{c1}$ and $\mathbf{x}_{c2}$. In that case,

$$\mathbf{U} = \begin{bmatrix} \mathbf{f}'(\mathbf{x}_{d1}) \\ \mathbf{f}'(\mathbf{x}_{d2}) \\ \mathbf{f}'(\mathbf{x}_{c1}) \\ \mathbf{f}'(\mathbf{x}_{c2}) \end{bmatrix}$$

and

$$\mathbf{D} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Note that $\mathbf{D}^{-1} = \mathbf{D}$ and that $|\mathbf{D}| = 1$ for all design changes studied in neighborhood N_3.

References

Arnouts, H. and Goos, P. (2010). Update formulas for split-plot and block designs, *Computational Statistics and Data Analysis* **54**: 3381–3391.

Atkinson, A. C., Donev, A. N. and Tobias, R. D. (2007). *Optimum Experimental Designs, with SAS*, Oxford: Oxford University Press.

Avanthay, C., Hertz, A. and Zufferey, N. (2003). A variable neighborhood search for graph coloring, *European Journal of Operational Research* **151**: 379–388.

Billionnet, A. and Calmels, F. (1996). Linear programming for the 0-1 quadratic knapsack problem, *European Journal of Operational Research* **92**: 310–325.

Bretthauer, K. and Shetty, B. (2002). The nonlinear knapsack problem — algorithms and applications, *European Journal of Operational Research* **138**: 459–472.

Caporossi, G. and Hansen, P. (2004). Variable neighborhood search for extremal graphs. 5. three ways to automate finding conjectures, *Discrete Mathematics* **276**: 81–94.

Chu, P. C. and Beasley, J. E. (1998). A genetic algorithm for the multidimensional knapsack problem, *Journal of Heuristics* **4**: 63–86.

Cornell, J. A. (2002). *Experiments with Mixtures: Designs, Models, and the Analysis of Mixture Data*, New York: Wiley.

De Groot, M. (1970). *Optimal Statistical Decisions*, New York: McGraw Hill.

De Ketelaere, B., Goos, P. and Brijs, K. (2011). Prespecified factor level combinations in the optimal design of mixture-process variable experiments, *Food Quality and Preference* **22**: 661–670.

Djerdjour, M., Mathur, K. and Salkin, H. M. (1988). A surrogate relaxation based algorithm for a general quadratic multi-dimensional knapsack problem, *Operations Research Letters* **7**: 253–258.

Fedorov, V. V. (1972). *Theory of Optimal Experiments*, New York: Academic Press.

Fleszar, K. and Hindi, K. S. (2004). Solving the resource-constrained project scheduling problem by a variable neighbourhood search, *European Journal of Operational Research* **155**: 402–413.

Galil, Z. and Kiefer, J. (1977). Comparison of simplex designs for quadratic mixture models, *Technometrics* **19**: 445–453.

Gallo, G., Hammer, P. L. and Simeone, B. (1980). Quadratic knapsack problems, *Mathematical Programming Studies* **12**: 132–149.

Garroi, J.-J., Goos, P. and Sörensen, K. (2009). A variable-neighbourhood search algorithm for finding optimal run orders in the presence of serial correlation, *Journal of Statistical Planning and Inference* **139**: 30–44.

Goos, P. (2002). *The Optimal Design of Blocked and Split-plot Experiments*, New York: Springer.

Goos, P. and Jones, B. (2011). *Design of Experiments: A Case-Study Approach*, New York: Wiley.

Goos, P., Jones, B. and Syafitri, U. (2013). I-optimal mixture designs, *Working paper 033*, Faculty of Applied Economics, University of Antwerp.

Goos, P. and Syafitri, U. (2014). V-optimal mixture designs for the qth degree model, *Chemometrics and Intelligent Laboratory Systems* **136**: 173–178.

Goos, P. and Vandebroek, M. (2004). Outperforming completely randomized designs, *Journal of Quality Technology* **36**: 12–26.

Hansen, P. and Mladenović, N. (2006). First vs. best improvement: an empirical study, *Discrete Applied Mathematics* **154**: 802–817.

Hardin, R. H. and Sloane, N. J. A. (1993). A new approach to construction of optimal designs, *Journal of Statistical Planning and Inference* **37**: 339–369.

Jones, B. and Goos, P. (2012). I-optimal versus D-optimal split-plot response-surface designs, *Journal of Quality Technology* **44**: 85–101.

Kellerer, H., Pferschy, U. and Pisinger, D. (2010). *Knapsack Problems*, Heidelberg: Springer.

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

Kurotori, I. (1966). Experiments with mixtures of components having lower bounds, *Industrial Quality Control* **22**: 592–596.

Kytöjoki, J., Nuortio, T., Bräysy, O. and Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems, *Computers and Operations Research* **34**: 2743–2757.

Laake, P. (1975). On the optimal allocation of observation in experiments with mixtures, *Scandinavian Journal of Statistics* **2**: 153–157.

Lambrakis, D. (1968a). Experiments with mixtures: A generalization of the simplex-lattice design, *Journal of Royal Statistical Society, Ser. B* **30**(1): 123–136.

Lambrakis, D. (1968b). Experiments with p-component mixtures, *Journal of Royal Statistical Society, Ser. B* **30**(1): 137–144.

Liu, S. and Neudecker, H. (1995). A V-optimal design for Scheffé's polynomial model, *Statistics and Probability Letters* **23**: 253–258.

Meyer, R. K. and Nachtsheim, C. J. (1995). The coordinate-exchange algorithm for constructing exact optimal experimental designs, *Technometrics* **37**: 60–69.

Mikaeili, F. (1989). D-optimum design for cubic without 3-way effect on the simplex, *Journal of Statistical Planning and Inference* **21**: 107–115.

Mikaeili, F. (1993). D-optimum design for full cubic on q-simplex, *Journal of Statistical Planning and Inference* **35**: 121–130.

Mitchell, T. J. (1974). An algorithm for the construction of D-optimal experimental designs, *Technometrics* **16**: 203–210.

Mladenović, M. and Hansen, P. (1997). Variable neighborhood search, *Computers and Operations Research* **24**: 1097–1100.

Mladenović, N., Petrović, J., Kovačević-Vujčić, V. and Čangalović, M. (2003). Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search, *European Journal of Operational Research* **151**: 389–399.

Morin, T. L. and Marsten, R. E. (1976). An algorithm for nonlinear knapsack problems, *Management Science* **22**: 1147–1158.

Piepel, G. F., Cooley, S. K. and Jones, B. (2005). Construction of a 21-component layered mixture experiment design using a new mixture coordinate-exchange algorithm, *Quality Engineering* **17**: 579–594.

Pisinger, D. (2007). The quadratic knapsack problem—a survey, *Discrete Applied Mathematics* **155**: 623—648.

Quadri, D., Soutif, E. and Tolla, P. (2009). Exact solution method to solve large scale integer quadratic multidimensional knapsack problems, *Journal of Combinatorial Optimization* **17**: 157–167.

Sartono, B., Goos, P. and Schoen, E. (2015). Constructing general orthogonal fractional factorial split-plot designs, *Technometrics* **57**: To appear.

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

Scheffé, H. (1963). The simplex-centroid design for experiments with mixtures, *Journal of the Royal Statistical Society, Ser. B* **25**: 235–263.

Smith, W. (2005). *Experimental Design for Formulation*, Philadelphia : Siam.

Uranisi, H. (1964). Optimal design for the special cubic regression model on the q-simplex, *Mathematical Report 1*, Kyushu University, General Education Department.

Vuchkov, I. and Donev, A. (1985). D-optimal designs generation for large number of variables, *45th Session of the International Statistical Institute*, Amsterdam, August 12-22. contributed paper.

Wang, H., Kochenberger, G. and Glover, F. (2012). A computational study on the quadratic knapsack problem with multiple constraints, *Computers and Operations Research* **39**: 3–11.

Wynn, H. P. (1972). Results in the theory and construction of D-optimum experimental designs, *Journal of the Royal Statistical Society, Ser. B* **34**: 133–147.