A nonlinear multidimensional knapsack problem in the optimal design of mixture experiments

Journal Pre-proof

A Nonlinear Multidimensional Knapsack Problem in the Optimal Design of Mixture Experiments

P. Goos, U. Syafitri, B. Sartono, A.R. Vazquez

PII: S0377-2217(19)30673-3
DOI: https://doi.org/10.1016/j.ejor.2019.08.020
Reference: EOR 15995

To appear in: *European Journal of Operational Research*

Received date: 28 June 2018
Accepted date: 9 August 2019

Please cite this article as: P. Goos, U. Syafitri, B. Sartono, A.R. Vazquez, A Nonlinear Multidimensional Knapsack Problem in the Optimal Design of Mixture Experiments, *European Journal of Operational Research* (2019), doi: https://doi.org/10.1016/j.ejor.2019.08.020

Highlights

- We define a novel type of nonseparable multidimensional knapsack problem
- The new knapsack problem arises in the statistical design of mixture experiments
- We propose a variable descent algorithm to compute D- and I-optimal designs
- We also propose a mixed integer nonlinear programming formulation
- We study designs for first-order and second-order Scheffé regression models

A Nonlinear Multidimensional Knapsack Problem in the Optimal Design of Mixture Experiments

P. Goos[a,b], U. Syafitri[c], B. Sartono[c], A. R. Vazquez[a]

[a] *Faculty of Bioscience Engineering & Leuven Statistics Research Centre (LStat), KU Leuven, Belgium*
[b] *Faculty of Business and Economics & StatUa Center for Statistics, Universiteit Antwerpen, Belgium*
[c] *Department of Statistics, Bogor Agricultural University, Indonesia*

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. The resulting optimal experimental design problem belongs to the class of nonlinear nonseparable multidimensional knapsack problems. We present a variable neighborhood search algorithm as well as a mixed integer nonlinear programming approach to tackle the problem to identify D- and I-optimal designs for mixture experiments when there is a limited stock of certain ingredients, and we show that the variable neighborhood descent algorithm is highly competitive in terms of solution quality and computing time.

Keywords: metaheuristics; nonlinear multidimensional knapsack problem; mixed integer nonlinear programming (MINLP); variable neighborhood search (VNS) algorithm; D- and I-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 for every test or run of the experiment. 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. When 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 specific for simplex-shaped experimental regions. These experimental designs were proposed by Scheffé (1958, 1963).

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 et al., 2007). In that case, it is common to compute mixture experimental designs that are optimal, in terms of precision of model estimation or precision of prediction, given all constraints on the factor levels. The point-exchange algorithm described in Atkinson et al. (2007) allows the construction of D-optimal experimental designs for mixture experiments. A coordinate-exchange algorithm for the construction of D-optimal mixture designs was proposed by Piepel et al. (2005), to deal with a problem involving many ingredients and multiple multicomponent constraints.

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 et al. (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 flour for a follow-up experiment was limited and could not be replenished.

Therefore, in this paper, we study the impact of ingredient 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 prediction variances was already suggested by Scheffé (1958), and because Hardin and Sloane (1993), Goos and Jones (2011) and Jones and Goos (2012b) 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. In the literature, few researchers have taken up the challenge to seek I-optimal mixture designs. Exceptions are Lambrakis (1968b,a); Laake (1975) and Liu and Neudecker (1995), who presented a limited number of theoretical results, and Goos and Syafitri (2014) and Goos et al. (2016), who provide literature reviews on D- and I-optimal mixture designs and present new I-optimal designs for simplex-shaped experimental regions in the absence of ingredient availability constraints.

A remarkable feature of the problem of designing mixture experiments in the presence of ingredient 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 ingredient availability constraints requires determining the optimal number of runs and determining the ingredient proportions at each of the runs.

In this paper, we first study a heuristic approach, involving a predefined set of candidate tests or runs and a variable neighborhood search (VNS) algorithm, for finding optimal mixture designs in the presence of ingredient availability constraints. This approach 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 with ingredient availability constraints to be a nonlinear nonseparable multidimensional knapsack problem. In the second part of the paper, we investigate a mixed integer nonlinear programming (MINLP) approach for our optimal mixture design problem, and use it as a benchmark for evaluating the performance of the heuristic approach.

The contributions of this paper are fourfold. First, we introduce a practically relevant nonlinear nonseparable multidimensional knapsack problem in the operations research literature. Second, we present an algorithm that produces high-quality solutions in a reasonable amount of computing time. Third, we present a novel MINLP formulation for the problem. Fourth, we help to bridge the gap between, on the one hand, statistics and design of experiments, and, on the other hand, operations research, in two ways: (i) we draw the attention of the operations research community to the existence of challenging optimization problems in statistics and design of experiments, and (ii) we demonstrate that a state-of-the-art metaheuristic and MINLP can be used to tackle optimization problems in statistics.

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 simplex-lattice designs. In Section 3, we discuss the similarity between the problem of optimally designing mixture experiments in the presence of ingredient availability constraints and the knapsack problem. In Section 4, we propose a VNS algorithm for the D- and I-optimal design of mixture experiments involving ingredient availability constraints. In Section 5, we describe the working of the VNS algorithm for two instances. In Section 6, we present D- and I-optimal designs produced by the VNS algorithm for several scenarios involving ingredient availability constraints. In Section 7, we introduce a MINLP formulation for our problem and use a state-of-the-art optimization solver to benchmark the performance of our heuristic approach. Finally, in Section 8, we end the paper with a discussion and a conclusion.

2. Models and designs for mixture experiments

Mixture experiments are intended to study the relation between the ingredient proportions and a property of interest. In this section, we first describe how that relation is generally modeled using linear regression techniques. Next, we discuss which experimental design approaches can be used to collect the data required to build sensible models.

2.1. Models for data from mixture experiments

The most commonly used regression 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

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

where $\mathbf{x}$ is the q-dimensional 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 values and the model expansions of all mixture ingredient proportions $x_1, \ldots, x_q$ for each of the n runs or tests 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 $\sigma_\epsilon^2 \mathbf{I}_n$, where $\mathbf{I}_n$ is the $n \times n$ identity matrix. In that case, the best linear unbiased estimator of the coefficient vector $\boldsymbol{\beta}$ is the ordinary least squares estimator

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

A specific feature of the regression models used for mixture experiments is that they do not

involve an intercept or constant. This is due to the fact that

$$\sum_{i=1}^{q} x_i = 1, \tag{5}$$

for each of the runs of a mixture experiment. This equality is generally referred to as the mixture constraint.

2.2. Simplex-lattice designs for mixture experiments

A $\{q, h\}$ simplex-lattice design for a mixture experiment consists of all combinations of the proportions $\{0, 1/h, 2/h, \ldots, 1\}$ for the q ingredients whose proportions sum to one. 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 present examples in which not just pure components and binary mixtures, but also ternary and quaternary mixtures appear in the optimal experimental design (i.e., mixtures with three or four nonzero ingredient proportions).

2.3. D-optimality criterion

One of the primary goals of a mixture experiment is to estimate the coefficient 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

$$\text{cov}(\hat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}'\mathbf{X})^{-1}. \tag{6}$$

The most commonly used approach to obtain a *small* variance-covariance matrix is to minimize its determinant, or, equivalently, to maximize the determinant of the information matrix $\sigma^{-2}(\mathbf{X}'\mathbf{X})$, which is the inverse of the ordinary least squares estimator's variance-covariance matrix. A design that maximizes the determinant of the information matrix, or, equivalently, the logarithm of the determinant of the information matrix, is called D-optimal, and the corresponding design selection criterion is referred to as the D-optimality criterion (Goos and Jones, 2011). 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.

In the absence of constraints other than the mixture constraint in Equation (5), the $\{q, 1\}$ and $\{q, 2\}$ simplex-lattice designs give the D-optimal design points for first-order and second-order Scheffé models, respectively (Kiefer, 1961). For a design with a given number of runs n to be D-optimal in the absence of ingredient availability constraints, its design points have to be replicated as evenly as possible. The $\{q, 1\}$ and $\{q, 2\}$ simplex-lattice designs remain D-optimal

when there are lower bound constraints on the proportions and the designs are rescaled to fit in the constrained experimental region.

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

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

A relative D-efficiency larger than one means that the design with model matrix $\mathbf{X}_1$ outperforms the one with model matrix $\mathbf{X}_2$.

2.4. I-optimality criterion

Rather than on the estimation of the coefficient 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

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

where S_{q-1} represents the experimental region, $\text{tr}\left[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{B}\right]$ denotes the trace of the matrix $(\mathbf{X}'\mathbf{X})^{-1}\mathbf{B}$, and

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

is the moments matrix (Goos et al., 2016). The denominator in the expressions for the average prediction variance is the volume of the experimental region.

The individual elements of the moments matrix $\mathbf{B}$ are all of the form

$$\int_{S_{q-1}} x_1^{p_1-1} x_2^{p_2-1} \dots x_q^{p_q-1} dx_1 dx_2 \dots dx_{q-1},$$

where each p_i is an integer larger than or equal to one. When the experimental region is a $(q-1)$-dimensional simplex, the elements of $\mathbf{B}$ 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 these expressions, the notation $\Gamma()$ refers to the gamma function. When the experimental region is not a simplex (due to the presence of upper bound constraints and/or multicomponent

constraints), the moments matrix $\mathbf{B}$ 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 that we can describe both the D-optimal design problem and the I-optimal design problem as maximization problems in the remainder of the paper. 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 for a simplex-shaped experimental region are the same as the D-optimal ones when there are no ingredient availability constraints. 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 proportions of 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 et al. (2016).

2.5. Existing approaches for computing D- and I-optimal designs

The vast majority of the algorithms on optimal design of experiments are heuristic and thus cannot guarantee the optimality of the designs produced. However, a limited amount of work on exact approaches has been done as well. Exact approaches do guarantee the optimality of the designs produced, unless the optimization is stopped prematurely due to a computing time limit. In this section, we provide a concise overview of the literature on heuristic and exact approaches for finding optimal experimental designs. In this overview, we ignore the part of the literature that deals with so-called approximate or continuous optimal experimental designs, because these designs are generally not feasible in practice.

2.5.1. Heuristic approaches

As can be seen from Chaper 12 in Atkinson et al. (2007), the vast majority of the algorithms for computing optimal experimental designs is heuristic and requires the specification of a candidate set of test combinations or design points. These algorithms are generally referred

to as point-exchange algorithms, because they iteratively improve the experimental design by exchanging its points with points from the candidate set. Atkinson et al. (2007) indicate that usually a coarse grid in the experimental region is used as the set of candidate design points.

Meyer and Nachtsheim (1995) presented a candidate-set-free heuristic algorithm to compute optimal experimental designs. Piepel et al. (2005) developed a variant of this algorithm to deal with mixture experiments. These algorithms are generally referred to as coordinate-exchange algorithms in the literature, because they improve the experimental design one coordinate at a time. Modern coordinate-exchange algorithms (Jones and Goos, 2012a; Ruseckaite et al., 2017; Huang et al., 2019) do no longer discretize the range of the experimental factors, since a one-dimensional continuous optimization algorithm such as the one from Brent (1973) can be used to determine optimal values for each coordinate.

The candidate-set-free approach is especially attractive for highly complex problems involving many experimental factors and many constraints on the levels of the factors, because a high dimensionality of the experimental region requires large candidate lists. Large candidate lists slow down the working of point-exchange algorithms, but this is compensated by the fact that point-exchange algorithms explore a broader neighborhood of any given solution. Another advantage of point-exchange algorithms is that the experimenter can specify a list of feasible test combinations as the set of candidates. This is in contrast with candidate-set-free algorithms which often produce test combinations that are practically infeasible, especially when they involve a continuous optimizer.

As demonstrated in the textbooks on mixture experiments by Cornell (2002) and Smith (2005), most mixture experiments involve no more than eight ingredients. Because we focus on mixture experiments in this article, the dimensionality of the problems to be solved is generally limited. As a result, there is no need to resort to a candidate-set-free algorithm and we can safely use an algorithm based on a candidate set. The challenge then is to come up with an algorithm that selects points from the candidate set such that they optimize the D- or I-optimality criterion, while making sure that the ingredient availability constraints are not violated.

Technically speaking, the point-exchange and coordinate-exchange algorithms in the optimal experimental design literature belong to the class of local search algorithms. A local search algorithm starts from a candidate solution and then iteratively moves to a neighboring solution. In a local search algorithm, typically, only one kind of neighboring solution is considered. In this paper, we go beyond such an algorithm, by developing a variable neighborhood search or VNS algorithm, which considers multiple kinds of neighboring solutions. We then evaluate the effectiveness of our VNS algorithm by comparing it to an exact approach.

2.5.2. Exact approaches

Exact optimization methods have rarely been used in optimal experimental design, especially when the goal is to determine the optimal test combinations or design points. One exception is the recent work of Duarte et al. (2019), who formulated the problems of finding D- and A-optimal experimental designs as MINLPs. A-optimal experimental designs minimize the trace

of the variance-covariance matrix in Equation (6), i.e., tr $\left[\sigma^2(\mathbf{X}'\mathbf{X})^{-1}\right]$.

The MINLP formulations of Duarte et al. (2019) do not require a candidate set. Instead, the coordinates of the design points are specified as continuous decision variables in the MINLP formulations. The MINLP formulations can accommodate a wide range of linear and nonlinear regression models, as well as constraints on the optimal design search. For example, Duarte et al. (2019) demonstrated that their MINLP formulations can be used to find D- and A-optimal designs for mixture experiments. However, their formulations cannot be used to compute optimal mixture designs in the presence of ingredient availability constraints. This is because, in their formulations, the total number of design points and the number of runs are considered given. As we explained in Section 1, the optimal number of runs and the optimal number of design points in a mixture experiment are not known in advance when ingredient availability constraints are present. In Section 7, we adapt the MINLP formulations of Duarte et al. (2019) to deal with ingredient availability constraints and to compute I-optimal designs instead of A-optimal designs. These MINLP formulations serve as a benchmark for the VNS algorithm we present.

3. The problem

When using a candidate set of test combinations or design points, the problem of finding an optimal design for a mixture experiment involving ingredient availability constraints is a multidimensional nonlinear nonseparable 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 combinations of feasible ingredient proportions. In any feasible design for a mixture experiment, each of these candidate design points is either used or not. When 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 ingredient availability constraints for each of the ingredients. If we denote the number of candidate points by N, then the number of experimental runs or observations corresponding to a given solution for the z_c values equals

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

If we denote the availability 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

$$\lambda \sum_{c=1}^{N} x_{ic} z_c \leq R_i. \tag{7}$$

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

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

An implicit upper bound for n therefore is $\lfloor R/\lambda \rfloor$, where $\lfloor a \rfloor$ denotes the largest integer smaller than or equal to a. Consequently, $\lfloor R/\lambda \rfloor$ 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 involving ingredient availability constraints 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 basic knapsack problem is called binary. The problem is unidimensional because there is only one capacity constraint, which imposes a maximum value on the total weight of all selected items. The problem is linear because both the objective function and the capacity constraint involve linear combinations of the z_c values.

Several extensions of the 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 (Becker and Buriol, 2019). If there is an upper bound for z_c, as is the case in the mixture design problem, the problem is called bounded. Other extensions of the basic knapsack problem involve multiple knapsacks (Dell'Amico et al., 2019) or allow fractional values for the decision variables z_c (Malaguti et al., 2019). In yet other extensions of the basic knapsack problem, the objective function is quadratic or another nonlinear function of the decision variables (Gallo et al., 1980; Billionnet and Calmels, 1996; Pisinger, 2007; Schauer, 2016).

The literature on knapsack problems also involves work dealing with multiple linear knapsack constraints. These knapsack problems are usually referred to as multidimensional knapsack

problems. An overview of much of the work on multidimensional knapsack problems with a linear ojective function is provided by Chu and Beasley (1998). Djerdjour et al. (1988), Quadri et al. (2009) and Wang et al. (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. A nonlinear knapsack problem involving both a nonlinear objective function and nonlinear constraints is defined by Morin and Marsten (1976). 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. Algorithms for solving nonlinear separable knapsack problems have been proposed by D'Ambrosio and Martello (2011) and D'Ambrosio et al. (2018).

Our mixture design problems for D- and I-optimality are also nonlinear knapsack problems because their 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 or separable nonlinear knapsack problems, which have a much simpler objective function than the optimal mixture design problem we study here.

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 decision 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, when starting from a candidate set of design points, our problem of finding optimal designs for mixture experiments involving ingredient 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 VNS algorithm in order to find D- and I-optimal designs for mixture experiments involving ingredient availability constraints.

4. Variable neighborhood search 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 to try to 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 et al., 2007), project scheduling (Fleszar and Hindi, 2004), automatic discovery of theorems (Caporossi and Hansen, 2004), graph coloring (Avanthay et al., 2003), the synthesis of radar polyphase codes (Mladenović et al., 2003), and the molecular distance geometry problem (Liberti et al., 2009). In the field of experimental design, VNS has been used to determine optimal run orders for the design points of standard experimental designs in the presence of serially correlated responses (Garroi et al., 2009), to assign treatments to whole plots in orthogonal split-plot designs (Sartono et al., 2015) and to optimize the concatenation of two orthogonal experimental designs (Vazquez et al., 2019). In each of these three cases, solving the optimal experimental design problem at hand involved the identification of an optimal permutation. This is very different

from the optimal design problem we consider here, because we need to find the optimal number of experimental runs, 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

In the first step of our VNS algorithm, 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. When 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. We denote the number of points in the candidate set by N.

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 20. This implies that we use large candidate sets and thus large values of N, since the $\{q, h\}$ simplex-lattice design involves

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

different combinations or design points.

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 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 VNS procedure. The construction of the initial design is described in Algorithm 1, where $\mathcal{S}$ represents the starting design.

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

Set $i \leftarrow 1$
$\mathcal{S} = \varnothing$
repeat
    Randomly choose a candidate point from $\mathcal{C}$ and name it $\mathbf{x}_i$
    $\mathcal{S} = \mathcal{S} \cup \mathbf{x}_i$
    Set $i \leftarrow i + 1$
until $\mathcal{S}$ violates one or more of the ingredient availability constraints
$\mathcal{S} = \mathcal{S} \setminus \mathbf{x}_{i-1}$
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 search algorithm

Our VNS 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 have 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 others. In our algorithm, we only consider feasible solutions (i.e., mixture designs that do not violate the ingredient 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

Table 1: Neighborhood structures N_k for the VNS 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	$n_c(n_c - 1) \times N^2$	Remove two current design points and add two candidate points

largest. Denoting the number of points in the design under evaluation by n_c, these two neighborhoods contain at most N and $n_c(n_c - 1) \times N^2$ alternative designs, respectively. Neighborhoods N_1 and N_2 contain at most $n_c \times N$ and $n_c \times N^2$ alternative designs, respectively. An overview of the four neighborhoods and their 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 from N_1, 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 design 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 empty. In that case, these neighborhoods are skipped during the execution of the VNS algorithm.

We use *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, in a random order, 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 2 Pseudocode of the VNS 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: $\mathcal{S}_{\max} \leftarrow \mathcal{S}$
22: $objmax \leftarrow obj(\mathcal{S}_{\max})$
23: **end if**
24: $k \leftarrow k + 1$
25: **until** $k = k_{\max}$
26: Return $\mathcal{S}_{\max}$.

algorithm is better and faster than a best-improvement algorithm when the initial solution is chosen at random.

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 VNS 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 $\mathcal{S}_{\max}$ 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 VNS algorithm in the procedure IML in SAS v9.2. We extracted the moments matrices required for the computation of I-optimal designs from the statistical software package JMP v13.

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 a 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 too 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 therefore 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, because the I-optimality criterion is a function of the inverse of the information matrix.

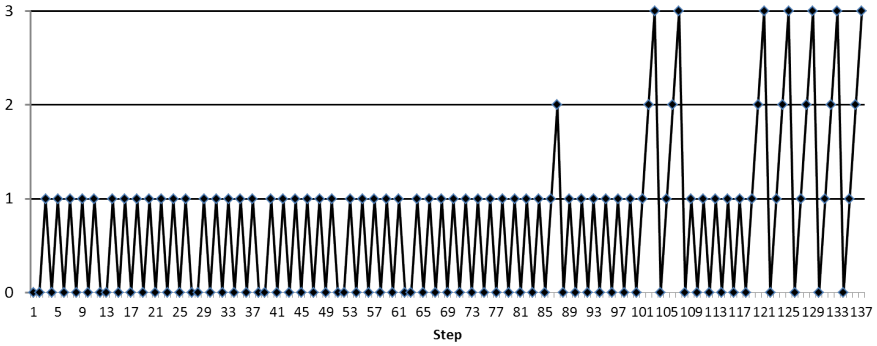
Using the update formulas requires the information matrix, $\mathbf{X}'\mathbf{X}$, to be non-singular. It is possible, however, that the number of distinct design points in the starting design is smaller than the number of parameters p in the Scheffé model under consideration. In that case, the information 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 (Goos, 2002).

5. The working of the variable neighborhood search algorithm

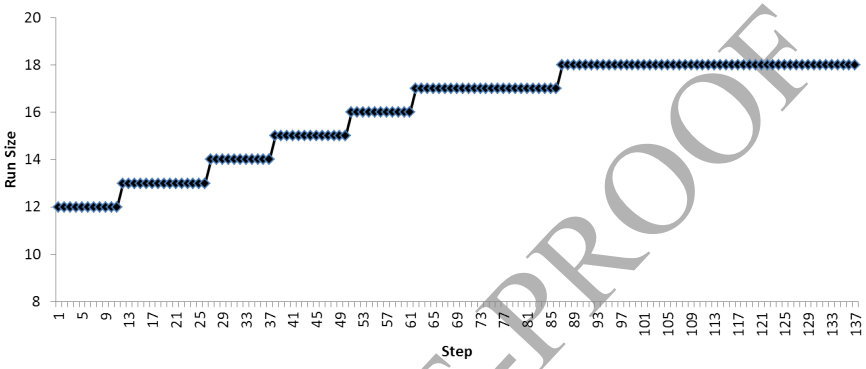
In this section, we demonstrate the usefulness of each of the neighborhoods of the VNS algorithm using a problem instance in which each experimental run requires at least 0.3 kg of ingredient 1 and 0.2 kg of ingredient 3. The available stock of ingredient 1 is 10.2 kg, the available stock of ingredient 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, in which a second-order Scheffé model is assumed and which is labeled Scenario 3.3.2 in Section 6, involves 66 points, i.e. $N = 66$. We ran the VNS algorithm 30 times, i.e. using $k_{\max} = 30$.

5.1. D-optimal design

The best design found by the VNS algorithm in terms of D-optimality was obtained in the tenth iteration. The number of runs of the initial design produced by Algorithm 1 in that iteration was equal to 11. Next, the VNS algorithm iteratively improved this initial design until it returned an 18-run design. Figure 1 shows how the VNS algorithm moved from one neighborhood to the



(a) Different steps of the VNS algorithm



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

Figure 1: Working of the VNS algorithm when constructing a D-optimal design for a 3-ingredient mixture experiment in which the availabilities of the three ingredients are 10.2 kg, 4 kg and 4.9 kg, the lower bounds on the proportions are 0.3, 0 and 0.2, and a second-order Scheffé model is considered.

other to improve the design step by step, along with its impact on the number of runs in the design.

Figure 1(a) visualizes how often the neighborhoods N_0, N_1, N_2 and N_3 were visited by the VNS algorithm. The horizontal axis shows that the VNS algorithm used 136 steps to move from the initial design to the final design, while the vertical axis shows which neighborhood was visited at each step. In the figure, a dot is associated with each visit to a neighborhood. There is a dot on the horizontal axis each time neighborhood N_0 is visited. There is a dot at level 1 each time neighborhood N_1 is visited. Similarly, there is a dot at level 2 each time neighborhood N_2 is visited, and a dot at level 3 each time neighborhood N_3 is visited.

In its first step, the algorithm visited neighborhood N_0 and identified a better design in that neighborhood. This caused the number of runs to increase by one unit, from 11 to 12. After this improvement to the design, neighborhood N_0 was revisited. The two successive visits of N_0 are shown in Figure 1(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 VNS algorithm moved to neighborhood N_1, where it was able to find an improved design with the same number of runs, 12. This caused the algorithm to return to neighborhood N_0. Because this did not lead to an improvement in the design, the VNS algorithm continued with neighborhood N_1, again with success. Figure 1(a) shows that the alternating visits to the first two neighborhoods continued until step 12. During that step, the visit of neighborhood N_0 led to an improved design with 13 runs, after which it was again neighborhood N_1 that repeatedly produced 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 resulted in a better design, each time with one extra experimental run.

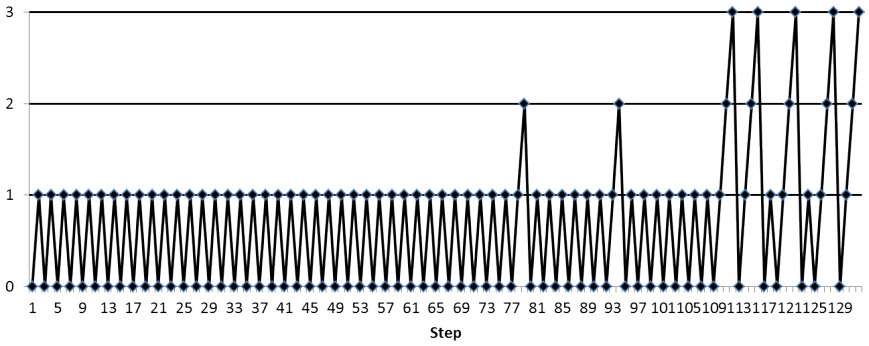
The next remarkable event in the course of the VNS algorithm occurred in steps 85 and 86, when neither N_0 nor N_1 allowed the algorithm to find an improved design. The algorithm therefore explored neighborhood N_2 in step 87. This did result in an improvement of the design, after which the algorithm returned to neighborhoods N_0 and N_1, until step 100. The design produced by the VNS algorithm at that step could not be improved by any of the neighborhoods N_0, N_1 and N_2, which is why neighborhood N_3 was visited. In the final steps of the algorithm, it was visiting the neighborhoods N_1 and N_3 that resulted in improved designs. The algorithm's final step was a visit to neighborhood N_3, which did not produce any better design and caused the VNS algorithm to terminate its tenth iteration.

5.2. I-optimal design

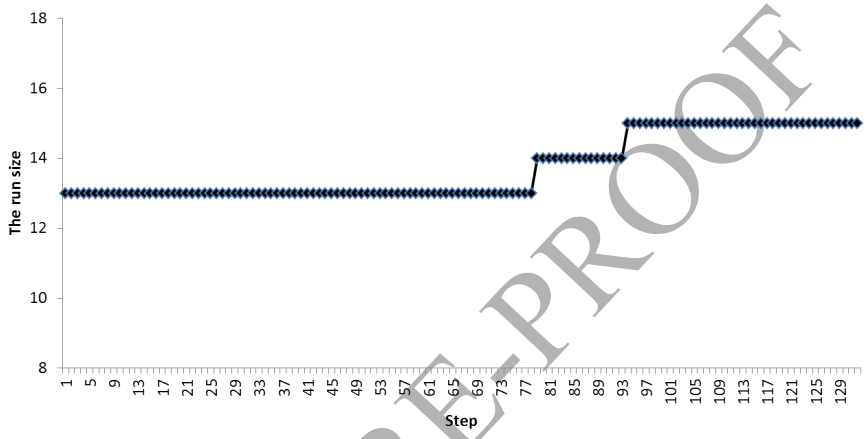
The best design found by the VNS algorithm in terms of I-optimality was found in the first iteration. The number of runs of the initial design produced by Algorithm 1 in that iteration was equal to 13. Next, the VNS algorithm improved this initial design until it returned a 15-run design. Figure 2 shows how the VNS 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.

Figure 2(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 VNS 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 VNS algorithm would not have visited N_3 in its final step.

Figure 2(b) shows that the number of runs in the design remains stable at 13 until step 78 of the VNS 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 design found in neighborhood N_2. During all other steps of the VNS algorithm, the number of runs does not change, so that the final design involves 15 runs.



(a) Different steps of the VNS algorithm



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

Figure 2: Working of the VNS algorithm when constructing an I-optimal design for a 3-ingredient mixture experiment in which the availabilities of the three ingredients are 10.2 kg, 4 kg and 4.9 kg, the lower bounds on the proportions are 0.3, 0 and 0.2, and a second-order Scheffé model is considered.

6. Results from the variable neighborhood search algorithm

In this section, we describe the results produced by our VNS algorithm for ten instances. The instances differ in the numbers of ingredients, the lower bounds on the ingredient proportions, the availabilities of the ingredients, and the models. We consider both first- and second-order Scheffé models. For each instance, we optimized the D- and I-optimality criterion using the VNS algorithm. Following the tradition in the optimal experimental design literature, we refer to the resulting designs as optimal designs, even though there is no guarantee that the designs produced by the VNS algorithm are indeed optimal.

6.1. Three unconstrained ingredients

Suppose that we have three ingredients and that there are no lower bounds on the ingredient proportions to be used. The first scenarios we consider (Scenarios 3.1.1 and 3.1.2) assume that

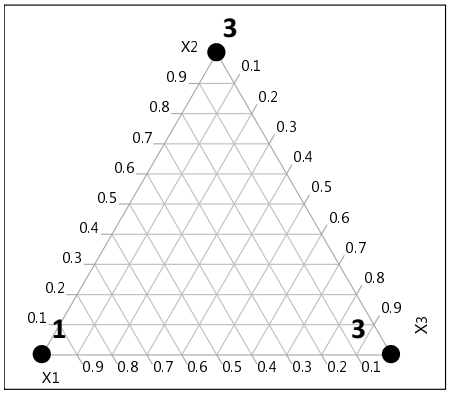
the available stock of ingredient 1 is 1.5 kg, the stock of ingredient 2 is 3 kg, and the stock of ingredient 3 is also 3 kg. The next scenarios (Scenarios 3.2.1 and 3.2.2) assume that the available stocks are 4 kg for ingredients 1 and 2, and 5 kg for ingredient 3. In all these scenarios, the number of points in the candidate set, N, was 253.

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 experimental runs. The D-optimal design and the I-optimal designs are shown in Figure 3. In the figure, the design points' labels represent the number of times these points appear in the optimal design. Figure 3(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 3(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%.

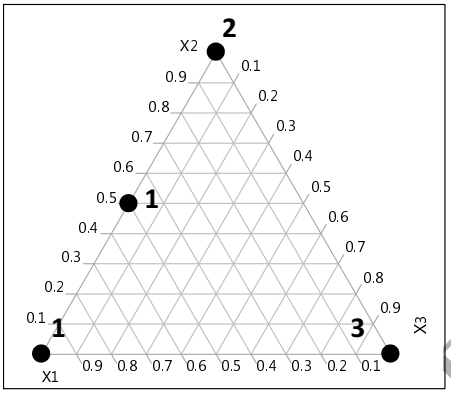
The result that the D- and I-optimal designs for Scenario 3.1.1 differ is surprising, because D- and I-optimal designs for first-order models are identical for many optimal experimental design problems in the absence of ingredient availability constraints. It is also interesting to point out that the 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 4 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.

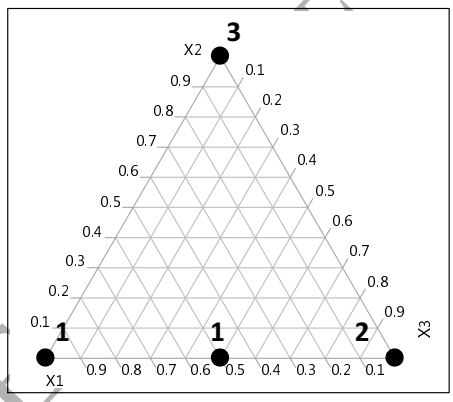
Even though they both involve six distinct design points, the D- and I-optimal designs for the second-order model in Scenario 3.1.2 are different from each other. This can be seen from Figures 5(a) and 5(b), which show the D-optimal design and the I-optimal design, respectively. Unlike the D-optimal 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. The presence of the ingredient availability constraints thus have a major impact on the optimal experimental 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 designs utilize the entire available



(a) D-optimal design



(b) First I-optimal design



(c) Second I-optimal design

Figure 3: The D-optimal design and the I-optimal designs for Scenario 3.1.1 (in which the availabilities of the three ingredients are 1.5 kg, 3 kg and 3 kg, there are no lower bounds on the proportions and a first-order Scheffé model is considered).

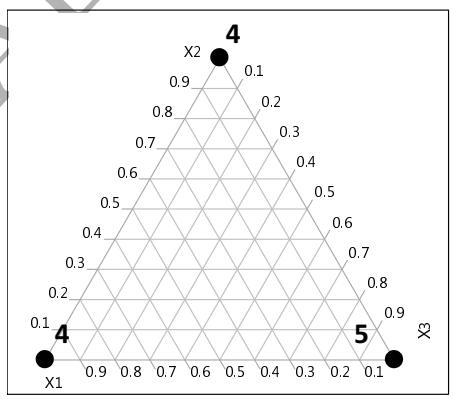
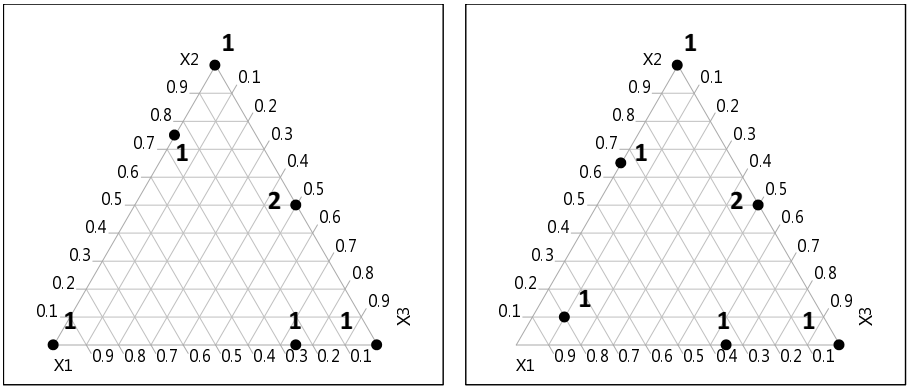


Figure 4: D- and I-optimal design for Scenario 3.2.1 (in which the availabilities of the three ingredients are 4 kg, 4 kg and 5 kg, there are no lower bounds on the proportions and a first-order Scheffé model is considered).



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

Figure 5: D- and I-optimal designs for Scenario 3.1.2 (in which the availabilities of the three ingredients are 1.5 kg, 3 kg and 3 kg, there are no lower bounds on the proportions and a second-order Scheffé model is considered).

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%.

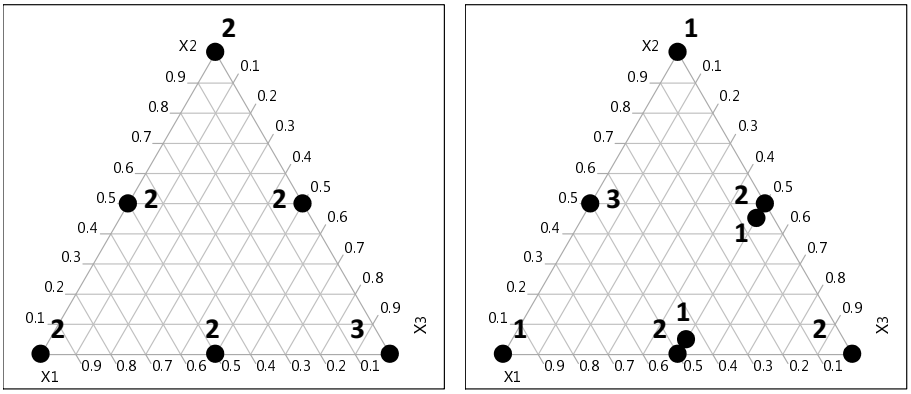
In Scenario 3.2.2, the D-optimal design involves the six points of the $\{3, 2\}$ simplex-lattice design, as shown in Figure 6(a). The design has two replicates at the midpoints of the edges of the experimental region, two replicates at the vertices $(x_1, x_2, x_3) = (1, 0, 0)$ and $(x_1, x_2, x_3) = (0, 1, 0)$, and three replicates at the vertex $(x_1, x_2, x_3) = (0, 0, 1)$. The D-optimal design's 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 6(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 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 (Laake, 1975; Goos et al., 2016).

6.2. Three constrained ingredients

Figure 7 shows the D- and I-optimal design for Scenario 3.3.1, involving a first-order model, lower bounds of 0.3 and 0.2 for ingredient 1 and ingredient 3, respectively, availabilities of 10.2



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

Figure 6: D- and I-optimal designs for Scenario 3.2.2 (in which the availabilities of the three ingredients are 4 kg, 4 kg and 5 kg, there are no lower bounds on the proportions and a second-order Scheffé model is considered).

kg, 4 kg and 4.9 kg for ingredients 1, 2 and 3, respectively, and 66 candidate points. The white triangle in Figure 7 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 would be 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 experimental 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 kg 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 R_i is 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 8(a), while the I-optimal design is shown in Figure 8(b). The D-optimal design has more

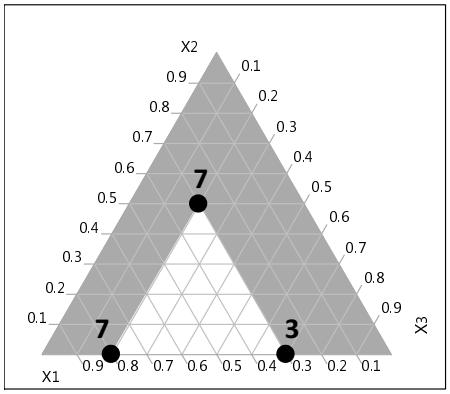
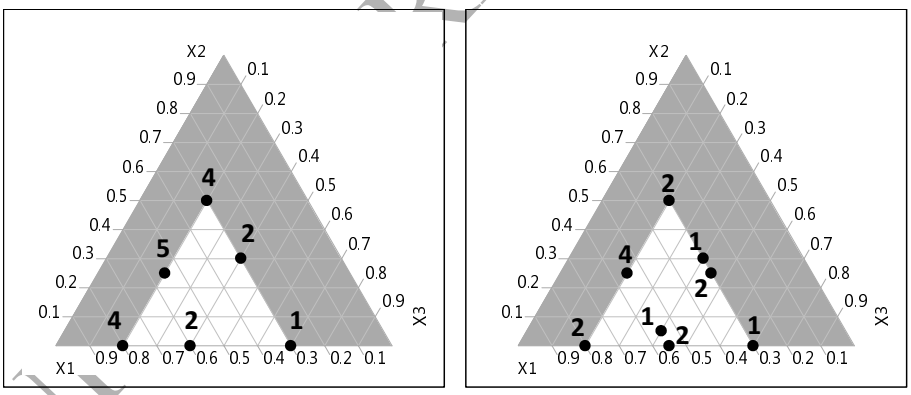


Figure 7: D- and I-optimal design for Scenario 3.3.1 (in which the availabilities of the three ingredients are 10.2 kg, 4 kg and 4.9 kg, the lower bounds on the proportions are 0.3, 0 and 0.2, and a first-order Scheffé model is considered).



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

Figure 8: Optimal designs for Scenario 3.3.2 (in which the availabilities of the three ingredients are 10.2 kg, 4 kg and 4.9 kg, the lower bounds on the proportions are 0.3, 0 and 0.2, and a second-order Scheffé model is considered).

Table 2: D-optimal design for Scenario 4.1 (in which the availabilities of the four ingredients are 2.5 kg, 6 kg, 3 kg and 7 kg, the lower bounds on the proportions are 0.2, 0.1, 0.1 and 0.2, and a first-order Scheffé model is considered).

Number	x_1	x_2	x_3	x_4	Multiplicity
1	0.2	0.1	0.1	0.6	3
2	0.2	0.1	0.5	0.2	3
3	0.2	0.5	0.1	0.2	3
4	0.6	0.1	0.1	0.2	1
Total	2.4	2.2	2.2	3.2	10
Availability	2.5	6	3	7	18.5

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% relative to the D-optimal design.

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. In the two scenarios with four ingredients, the number of points in the candidate set, N, was 165.

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. 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, with ingredient proportions $(x_1, x_2, x_3, x_4) = (0.3, 0.1, 0.4, 0.2)$, that is not a vertex. This can be seen by comparing Table 2 and Table 3, which show the D- and I-optimal designs, respectively. The last column of each of the two tables shows the multiplicity of each design point.

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% relative to the D-optimal design.

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 4 and 5. 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

Table 3: I-optimal design for Scenario 4.1 (in which the availabilities of the four ingredients are 2.5 kg, 6 kg, 3 kg and 7 kg, the lower bounds on the proportions are 0.2, 0.1, 0.1 and 0.2, and a first-order Scheffé model is considered).

Number	x_1	x_2	x_3	x_4	Multiplicity
1	0.2	0.1	0.1	0.6	3
2	0.2	0.1	0.5	0.2	2
3	0.2	0.5	0.1	0.2	3
4	0.3	0.1	0.4	0.2	1
5	0.6	0.1	0.1	0.2	1
Total	2.5	2.2	2.1	3.2	10
Availability	2.5	6	3	7	18.5

Table 4: D-optimal design for Scenario 4.2 (in which the availabilities of the four ingredients are 2.5 kg, 6 kg, 3 kg and 7 kg, the lower bounds on the proportions are 0.2, 0.1, 0.1 and 0.2, and a second-order Scheffé model is considered).

Number	x_1	x_2	x_3	x_4
1	0.2	0.1	0.1	0.6
2	0.2	0.1	0.3	0.4
3	0.2	0.1	0.5	0.2
4	0.2	0.3	0.1	0.4
5	0.2	0.3	0.3	0.2
6	0.2	0.5	0.1	0.2
7	0.25	0.1	0.1	0.55
8	0.3	0.1	0.4	0.2
9	0.3	0.4	0.1	0.2
10	0.45	0.1	0.1	0.35
Total	2.5	2.1	2.1	3.3
Availability	2.5	6	3	7

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.

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. In the two scenarios with six ingredients, the number of points in the candidate set, N, was 252.

For Scenario 6.1, in which a first-order model is assumed, 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

Table 5: I-optimal design for Scenario 4.2 (in which the availabilities of the four ingredients are 2.5 kg, 6 kg, 3 kg and 7 kg, the lower bounds on the proportions are 0.2, 0.1, 0.1 and 0.2, and a second-order Scheffé model is considered).

Number	x_1	x_2	x_3	x_4
1	0.2	0.1	0.1	0.6
2	0.2	0.1	0.3	0.4
3	0.2	0.1	0.5	0.2
4	0.2	0.3	0.1	0.4
5	0.2	0.3	0.3	0.2
6	0.2	0.5	0.1	0.2
7	0.25	0.1	0.1	0.55
8	0.3	0.1	0.35	0.25
9	0.3	0.35	0.1	0.25
10	0.45	0.15	0.15	0.25
Total	2.5	2.3	2.1	3.1
Availability	2.5	6	3	7

Table 6: D-optimal design for Scenario 6.1 (in which the availabilities of the six ingredients are 4 kg, 4 kg, 5 kg, 5 kg, 8 kg and 16 kg, the lower bounds on the proportions are 0.05, 0.1, 0.1, 0.1, 0.2 and 0.2, and a first-order Scheffé model is considered).

Number	x_1	x_2	x_3	x_4	x_5	x_6	Multiplicity
1	0.05	0.1	0.1	0.1	0.2	0.45	9
2	0.05	0.1	0.1	0.1	0.45	0.2	4
3	0.05	0.1	0.1	0.35	0.2	0.2	6
4	0.05	0.1	0.35	0.1	0.2	0.2	6
5	0.05	0.35	0.1	0.1	0.2	0.2	2
6	0.3	0.1	0.1	0.1	0.2	0.2	8
Total	3.75	4	5	5	8	9.25	35
Availability	4	4	5	5	8	16	42

design points of the two designs are shown in Tables 6 and 7, along with the number of replicates at each of them.

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% relative to the D-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 8, 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 9, 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,

Table 7: I-optimal design for Scenario 6.1 (in which the availabilities of the six ingredients are 4 kg, 4 kg, 5 kg, 5 kg, 8 kg and 16 kg, the lower bounds on the proportions are 0.05, 0.1, 0.1, 0.1, 0.2 and 0.2, and a first-order Scheffé model is considered).

Number	x_1	x_2	x_3	x_4	x_5	x_6	Multiplicity
1	0.05	0.1	0.1	0.1	0.2	0.45	6
2	0.05	0.1	0.1	0.1	0.45	0.2	6
3	0.05	0.1	0.1	0.35	0.2	0.2	7
4	0.05	0.1	0.35	0.1	0.2	0.2	5
5	0.05	0.35	0.1	0.1	0.2	0.2	3
6	0.05	0.15	0.3	0.1	0.2	0.2	1
7	0.3	0.1	0.1	0.1	0.2	0.2	5
Total	2.85	4	4.65	4.7	7.9	7.9	32
Availability	4	4	5	5	8	16	42

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% relative to the D-optimal design.

(

6.5. Computing times for the VNS algorithm

For each of the scenarios discussed above, in addition to the optimal experimental designs, we also recorded the computing times of the VNS algorithm at each iteration. For the ten scenarios, involving 66–253 candidate points, the average computing time per iteration of the VNS ranged from 11 seconds to 856 seconds for D-optimality on a standard CPU (Intel(R) Core(TM) i7 processor, 2.93 Ghz, 4 GB). The computing times for I-optimality were generally slightly longer. The shortest computing times were for D-optimal designs for Scenarios 3.1.1 and 3.1.2 and amounted to 11 seconds and 15 seconds per iteration, respectively. The I-optimal designs for these scenarios required 13–14 seconds per iteration. For the scenarios involving four ingredients, the average computing time per iteration of the VNS algorithm was at most 68 seconds. The longest computing time was for Scenario 6.2 involving 252 candidate points, six ingredients, a 21-dimensional information matrix and a total ingredient availability of 42.

The computing time of the VNS algorithm increases substantially with the value of h chosen to create the simplex-lattice design on which the candidate set is based. This is mainly due to neighborhoods N_2 and N_3, because these neighborhoods' sizes increase quadratically with the number of candidate points, N. Another factor that drives the computing time is the total ingredient availability. This is because a larger ingredient availability allows for a larger number of experimental runs, and the size of neighborhood N_3 also increases quadratically with the number of experimental runs in the designs under investigation, n_c. Finally, the computing time increases with the number of parameters in the regression model too. This is because the

Table 8: D-optimal design for Scenario 6.2 (in which the availabilities of the six ingredients are 4 kg, 4 kg, 5 kg, 5 kg, 8 kg and 16 kg, the lower bounds on the proportions are 0.05, 0.1, 0.1, 0.1, 0.2 and 0.2, and a second-order Scheffé model is considered).

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

Table 9: I-optimal design for Scenario 6.2 (in which the availabilities of the six ingredients are 4 kg, 4 kg, 5 kg, 5 kg, 8 kg and 16 kg, the lower bounds on the proportions are 0.05, 0.1, 0.1, 0.1, 0.2 and 0.2, and a second-order Scheffé model is considered).

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

dimensions of the information matrix and its inverse depend on that number of parameters, and computing determinants and inverses of larger matrices is computationally more demanding than computing determinants and inverses of smaller matrices.

7. A mixed integer nonlinear programming (MINLP) formulation

In this section, building on the work of Duarte et al. (2019), we introduce MINLP formulations for the problems of finding D- and I-optimal mixture designs in the presence of ingredient availability constraints. This serves four purposes. First, it is interesting to investigate what the maximum size of problem is that can be tackled using an exact optimization approach. Second, our MINLP formulation for I-optimality is the first one proposed for that optimality criterion. Third, we introduce several symmetry breaking constraints in the MINLP literature on optimal experimental design. Finally, the MINLP approach provides a benchmark for the solution quality and the computing time of our VNS algorithm. In our MINLP formulations, the number of design points is optimized (this is a fundamental difference between our formulations and those of Duarte et al. (2019)), along with the values of the mixture ingredients' proportions at the design points. So, our MINLP approach to compute optimal mixture designs in the presence of ingredient availability constraints does not require a candidate set.

7.1. Notation and preliminaries

Before presenting our MINLP formulations for D- and I-optimal design of mixture experiments with ingredient availability constraints, we need to introduce some new notation and redefine some of the symbols used earlier. First, we denote by N the maximum number of experimental runs. As explained in Section 3, the maximum number of runs in the optimal design is bounded by $\lfloor R/\lambda \rfloor$. So, $N = \lfloor R/\lambda \rfloor$. In our MINLP formulations, we use $q+1$ decision variables for each of the N potential runs. The first of these decision variables is a binary variable a_c, which takes the value 1 if the cth potential run is included in the optimal design (in which case the run is active) and the value 0 otherwise. The N variables $a_1, a_2, \ldots, a_N$ are the only binary decision variables in the formulations. The remaining q decision variables corresponding to the cth potential run are its proportions of the q mixture ingredients. We denote these ingredient proportions by $\mathbf{x}_c = (x_{1c}, x_{2c}, \ldots, x_{qc})'$, and the model expansion of $\mathbf{x}_c$ by the p-dimensional vector $\mathbf{f}(\mathbf{x}_c)$. The vector $\mathbf{a}$ in our MINLP formulations collects the values of all binary decision variables $a_1, a_2, \ldots, a_N$. We represent the lower and upper bounds for the ith ingredient's proportion at each run by B_{il} and B_{iu}, respectively, and the availability of ingredient i by R_i.

In our MINLP formulations, we use a normalized information matrix. As we explain in Section 7.4, this has the technical advantage that we can impose several useful bounds on the continuous decision variables. The normalization does not affect the optimal solution of the formulations. Following Duarte et al. (2019), our MINLP formulations calculate the determinant and the inverse of the $p \times p$ normalized information matrix, which we denote by $\mathbf{M}$ in this section, using its Cholesky factorization. The Cholesky factorization of $\mathbf{M}$ is given by $\mathbf{M} = \mathbf{U}'\mathbf{U}$, where

$\mathbf{U}$ is a $p \times p$ upper triangular matrix. Computing the determinant of $\mathbf{M}$ is equivalent to computing the squared product of the diagonal elements of the matrix $\mathbf{U}$, and computing the inverse of $\mathbf{M}$ can be done by multiplying the inverse of $\mathbf{U}$ with its transpose. In the formulations, we denote by m_{ij} the element in the ith row and jth column of the normalized information matrix $\mathbf{M}$ and by u_{ij} the element in the ith row and jth column of matrix $\mathbf{U}$. Since $\mathbf{M}$ is symmetric, $m_{ij} = m_{ji}$ for each i and j. Because $\mathbf{U}$ is an upper triangular matrix, $u_{ij} = 0$ for each $i > j$. Finally, in both our formulations, ϵ is an auxiliary constant larger than zero, which we set to 0.00001.

After solving the MINLPs to optimality, there are two main pieces of output. First, the nonzero a_c values indicate the active runs, i.e., the runs that actually appear in the optimal design, and the sum $n = \sum_{c=1}^{N} a_c$ is its total number of runs. Second, the $\mathbf{x}_c$ vectors of the active runs represent the optimal values of the ingredient proportions at the runs of the optimal design.

7.2. MINLP formulation for D-optimality

Our MINLP formulation for the problem of finding D-optimal mixture designs in the presence of ingredient availability constraints is as follows:

$$\max_{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N,\boldsymbol{a},\mathbf{M},\mathbf{U}} \sum_{i=1}^{p} 2\log(u_{ii}) \tag{8a}$$

subject to

Model constraints:

$$\mathbf{M} = \sum_{c=1}^{N} \frac{a_c}{N} \mathbf{f}(\mathbf{x}_c)\mathbf{f}'(\mathbf{x}_c), \tag{8b}$$

$$\mathbf{M} = \mathbf{U}'\mathbf{U}, \tag{8c}$$

Mixture-specific constraints:

$$\sum_{i=1}^{q} x_{ic} = 1, \quad c = 1,\ldots,N, \tag{8d}$$

$$\sum_{c=1}^{N} a_c x_{ic} \le R_i, \quad i = 1,\ldots,q, \tag{8e}$$

$$B_{il} \le x_{ic} \le B_{iu}, \quad i = 1,\ldots,q,\ c = 1,\ldots,N, \tag{8f}$$

Symmetry breaking constraints:

$$a_c = 1, \quad c = 1,\ldots,p, \tag{8g}$$

$$a_c \ge a_{c+1}, \quad c = 1,\ldots,N-1, \tag{8h}$$

$$x_{1c} \le x_{1,c+1}, \quad c = 1,\ldots,N-1, \tag{8i}$$

Technical constraints:

$$u_{ii} \ge \epsilon, \quad i = 1,\ldots,p, \tag{8j}$$

$$u_{ij} = 0, \quad i = 1,\ldots,p,\ j = 1,\ldots,i-1, \tag{8k}$$

$$m_{ii} \ge u_{ij}^2, \quad i = 1,\ldots,p,\ j = i+1,\ldots,p, \tag{8l}$$

$$m_{ij} = m_{ji}, \quad i = 1,\ldots,p,\ j = i+1,\ldots,p, \tag{8m}$$

$$a_c \in \{0,1\}, \quad c = 1,\ldots,N. \tag{8n}$$

The nonlinear objective function in Equation (8a) represents the logarithm of the determinant of the normalized information matrix $\mathbf{M}$, expressed in terms of the diagonal elements of Cholesky factor $\mathbf{U}$. More specifically, $\log(|\mathbf{M}|)$ is expressed as twice the sum of the logarithms of the variables u_{ii}. Our objective function uses the logarithm of the determinant of the information matrix because it is a convex objective function which is computationally more convenient to optimize than the determinant itself (Atkinson et al., 2007), and because optimizing the logarithm of a determinant is, of course, equivalent to optimizing the determinant itself.

The MINLP formulation for D-optimality involves four kinds of constraints: model constraints, mixture-specific constraints, symmetry breaking constraints and technical constraints.

Both model constraints are nonlinear. The constraint in (8b) links the values of the ingredient proportions of the active runs to the normalized information matrix $\mathbf{M}$, through the model expansions $\mathbf{f}(\mathbf{x}_c)$. The constraint in Equation (8c) represents the Cholesky factorization of $\mathbf{M}$.

There are three types of mixture-specific constraints. The first type of mixture-specific constraint, given in Equation (8d), is a linear constraint which ensures that the proportions of the ingredients sum to 1 at each potential run (see also Equation (5)). The second type of mixture-specific constraint, given in Equation (8e), ensures that the ingredient availability constraints are satisfied for each of the q ingredients. The third type of mixture-specific constraint, given in Equation (8f), imposes lower and upper bounds on the proportions of the q ingredients at each run.

It is beneficial to linearize the constraints in Equation (8e) to speed up the solution of the MINLP formulation. To this end, we need to introduce additional continuous decision variables w_{ic} representing the products $x_{ic}a_c$, and replace the constraints in Equation (8e) with the following five types of linear constraints:

$$w_{ic} \leq a_c, \quad i = 1, \ldots q, \; c = 1, \ldots, N, \tag{9a}$$

$$w_{ic} \leq x_{ic}, \quad i = 1, \ldots q, \; c = 1, \ldots, N, \tag{9b}$$

$$w_{ic} \geq x_{ic} - (1 - a_c), \quad i = 1, \ldots q, \; c = 1, \ldots, N, \tag{9c}$$

$$w_{ic} \geq 0, \quad i = 1, \ldots q, \; c = 1, \ldots, N, \tag{9d}$$

$$\sum_{c=1}^{N} w_{ic} \leq R_i, \quad i = 1, \ldots, q. \tag{9e}$$

The first four of these types of constraints, given in Equations (9a)–(9d), ensure that $w_{ic} = x_{ic}a_c$. The fifth type of constraint, given in Equation (9e), expresses the ingredient availability constraints in terms of the new decision variables w_{ic}. Since the linear constraints in Equations (9a)–(9e) allowed a substantially faster solution of the MINLP formulation than the original nonlinear constraints in Equation (8e), we report results only for the MINLP formulation using Equations (9a)–(9e) rather than Equation (8e).

Our formulation involves three types of symmetry breaking constraints. These constraints are intended to speed up the solution of the MINLP, by removing isomorphic solutions from the solution space (for a review on symmetry breaking techniques, we refer to Vo-Thanh et al. (2018)). In our MINLP, reordering the runs does not affect the solution quality. Therefore, if one solution can be obtained from another by reordering the runs, then these two solutions are equivalent. In the jargon, we say these solutions are isomorphic or symmetric. There is obviously no point in evaluating more than one solution from each given set of isomorphic solutions. This is why it makes sense to add constraints to a formulation that reduce the number of isomorphic solutions considered when solving a MINLP. The first symmetry breaking constraint in Equation (8g)

informs the optimization model that at least p runs are required in the experiment, because estimating a regression model involving p parameters requires at least p data points. Without loss of generality, we can make the first p potential runs active. The constraints in Equation (8h) only allow a potential design point $c+1$ to become active if design point c is active already. Finally, the constraints in Equation (8i) impose a lexicographic ordering on the experimental runs. Lexicographic ordering is a commonly used symmetry breaking technique (Vo-Thanh et al., 2018).

The technical constraints in Equations (8j) and (8k) ensure that the Cholesky factor $\mathbf{U}$ has positive diagonal elements and that it is an upper triangular matrix, respectively. The technical constraints in Equation (8l) are intended to guarantee numerical stability for the Cholesky factorization and were used by Duarte et al. (2019) as well. In other words, these constraints allow the solver to compute the Cholesky factorization effectively. Equation (8m) indicates that the normalized information matrix is a symmetric matrix. The final technical constraints are given in Equation (8n) and ensure that the variables a_c are binary.

7.3. MINLP formulation for I-optimality

Just like the formulation for A-optimality in Duarte et al. (2019), our MINLP formulation for finding I-optimal designs uses Cholesky factorization for determining the inverse of the normalized information matrix $\mathbf{M}$. In our formulation, we denote the inverse of $\mathbf{M}$ by $\overline{\mathbf{M}}$ and the inverse of the Cholesky factor $\mathbf{U}$ by $\overline{\mathbf{U}}$. So, we calculate the inverse of the normalized information matrix as $\overline{\mathbf{M}} = \overline{\mathbf{U}}\overline{\mathbf{U}}'$. In our formulation for I-optimality, we also utilize the moments matrix $\mathbf{B}$ introduced in Section 2.4, and we denote the elements of $\overline{\mathbf{M}}$ and $\overline{\mathbf{U}}$ by $\overline{m}_{ij}$ and $\overline{u}_{ij}$, respectively. Since $\overline{\mathbf{M}}$ is symmetric, we have that $\overline{m}_{ij} = \overline{m}_{ji}$ for each i and j. Also, because $\overline{\mathbf{U}}$ is an upper triangular matrix, $\overline{u}_{ij} = 0$ for each $i > j$. Finally, we denote the $p \times p$ identity matrix by $\mathbf{I}_p$. Our MINLP

formulation for I-optimality is as follows:

$$\max_{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N,\boldsymbol{\alpha},\mathbf{M},\overline{\mathbf{M}},\mathbf{U},\overline{\mathbf{U}}} \; -\operatorname{tr}\left[\overline{\mathbf{M}}\mathbf{B}\right] \tag{10a}$$

subject to

Model constraints:

$$\mathbf{M} = \sum_{c=1}^{N} \frac{z_c}{N} \mathbf{f}(\mathbf{x}_c)\mathbf{f}'(\mathbf{x}_c), \tag{10b}$$

$$\mathbf{M} = \mathbf{U}'\mathbf{U}, \tag{10c}$$

$$\mathbf{U}\overline{\mathbf{U}} = \mathbf{I}_p, \tag{10d}$$

$$\overline{\mathbf{M}} = \overline{\mathbf{U}}\overline{\mathbf{U}}', \tag{10e}$$

Mixture-specific constraints:

$$\sum_{i=1}^{q} x_{ic} = 1, \quad c = 1, \ldots, N, \tag{10f}$$

$$\sum_{c=1}^{N} a_c x_{ic} \leq R_i, \quad i = 1, \ldots, q, \tag{10g}$$

$$B_{il} \leq x_{ic} \leq B_{iu}, \quad i = 1, \ldots, q; c = 1, \ldots, N, \tag{10h}$$

Symmetry breaking constraints:

$$a_c = 1, \quad c = 1, \ldots, p, \tag{10i}$$

$$a_c \geq a_{c+1}, \quad c = 1, \ldots, N-1, \tag{10j}$$

$$x_{1c} \leq x_{1,c+1}, \quad c = 1, \ldots, N-1, \tag{10k}$$

Technical constraints:

$$u_{ii} \geq \epsilon, \quad i = 1, \ldots, p, \tag{10l}$$

$$u_{ij} = 0, \quad i = 1, \ldots, p, \; j = 1, \ldots, i-1, \tag{10m}$$

$$\overline{u}_{ii} \geq \epsilon, \quad i = 1, \ldots, p, \tag{10n}$$

$$\overline{u}_{ij} = 0, \quad i = 1, \ldots, p, \; j = 1, \ldots, i-1, \tag{10o}$$

$$m_{ii} \geq u_{ij}^2, \quad i = 1, \ldots, p, \; j = i+1, \ldots, p, \tag{10p}$$

$$\overline{m}_{ii} \geq \overline{u}_{ij}^2, \quad i = 1, \ldots, p, \; j = i+1, \ldots, p, \tag{10q}$$

$$m_{ij} = m_{ji}, \quad i = 1, \ldots, p, \; j = i+1, \ldots, p, \tag{10r}$$

$$\overline{m}_{ij} = \overline{m}_{ji}, \quad i = 1, \ldots, p, \; j = i+1, \ldots, p, \tag{10s}$$

$$a_c \in \{0, 1\}, \quad c = 1, \ldots, N. \tag{10t}$$

The objective function in Equation (10a) is the negative of the average prediction variance, which is calculated as the trace of the matrix $-\mathbf{M}^{-1}\mathbf{B} = -\overline{\mathbf{M}}\mathbf{B}$.

The problem formulation for I-optimality involves the same four kinds of constraints as the formulation for the D-optimal mixture design in Section 7.2: model constraints, mixture-specific constraints, symmetry breaking constraints and technical constraints.

There are four model constraints, all of which are nonlinear. The constraint in (10b) links the values of the ingredient proportions of the active design points to the normalized information matrix $\mathbf{M}$, through the model expansions $\mathbf{f}(\mathbf{x}_c)$. The constraint in Equation (10c) represents the Cholesky factorization of $\mathbf{M}$. The third type of constraint, given in Equation (10d), is intended to compute the inverse $\overline{\mathbf{U}}$ of the Cholesky factor $\mathbf{U}$. The final model constraint, appearing in Equation (10e), computes the inverse of the information matrix, $\overline{\mathbf{M}}$, using the inverse $\overline{\mathbf{U}}$ of the Cholesky factor $\mathbf{U}$.

The mixture-specific constraints in Equations (10f)–(10h) of the formulation for I-optimality are identical to those in Equations (8d)–(8f) for D-optimality in Section 7.2. However, also for the I-optimality criterion, we sped up the solution of our MINLP by utilizing the linear constraints in Equations (9a)–(9e). So, we replaced the nonlinear mixture-specific constraint in Equation (10g) by Equations (9a)–(9e). The symmetry breaking constraints in Equations (10i)–(10k) of the formulation for I-optimality are identical to those in Equations (8g)–(8i) for D-optimality.

The technical constraints in Equations (10l) and (10m) ensure that the Cholesky factor $\mathbf{U}$ has positive diagonal elements and lower diagonal elements equal to zero. The technical constraints in Equations (10n) and (10o) ensure that $\overline{\mathbf{U}}$ also has positive diagonal elements and lower diagonal elements equal to zero. The technical constraints in Equations (10p) and (10q) are intended to guarantee numerical stability for the Cholesky factorization. Equations (10r) and (10s) indicate that the normalized information matrix and its inverse are symmetric matrices. Finally, the constraints in (10t) ensure that the variables a_c are binary.

7.4. Implementation

We implemented our problem formulations for D- and I-optimality in GAMS v25.1.3 and used the solver BARON v18.5.8 (Sahinidis, 2017). We carried out all computations of D- and I-optimal designs on the NEOS server (Czyzyk et al., 1998).

As the performance of the solver heavily depended on the bounds we used for the continuous decision variables, we carefully defined bounds for these variables. The decision variables x_{ic} represent proportions. Therefore, they are bounded to be between 0 and 1 (and between B_{il} and B_{iu} if additional constraints on the proportions of the ingredients are present). This implies that the elements of the model expansion vectors $\mathbf{f}(\mathbf{x}_c)$ lie between 0 and 1 as well, and that the variables m_{ij}, representing the elements of the normalized information matrix $\mathbf{M}$, are also restricted to be between 0 and 1.

The constraints in Equations (8l) and (10p) impose upper and lower bounds on the variables u_{ij} of the Cholesky factor $\mathbf{U}$ of the normalized information matrix $\mathbf{M}$. More specifically, the values of these variables must take values between -1 and 1. For the variables $\overline{u}_{ij}$ and $\overline{m}_{ij}$, representing the elements of the inverses of $\mathbf{U}$ and $\mathbf{M}$, respectively, we were unable to identify

upper and lower bounds. Therefore, we arbitrarily set these variables to be between −1000 and 1000, to limit the probability of missing the optimal solution.

7.5. MINLP results

In this section, we compare the results of the MINLP approach with those of the VNS algorithm for the 3- and 4-ingredient examples from Section 6. We discuss both the solution quality and the computing time. As discussed in Section 6.5, the computing times for each iteration the VNS algorithm were expressed in minutes for the optimal design problems discussed here.

7.5.1. D-optimality

For Scenarios 3.1.1, 3.2.1, 3.3.1 and 4.1, BARON was able to solve our MINLP formulation for finding D-optimal designs to optimality within 1, 1, 173 and 448 seconds, respectively. For these scenarios, the BARON solver produced the same designs as the VNS algorithm. These designs are shown in Figures 3(a), 4 and 7, and in Table 2.

For none of the other scenarios, BARON was able to finish the search within our computing time limit of four hours. For Scenario 3.2.2, BARON did return the same solution as our VNS algorithm, but it was unable to confirm the optimality of that solution. The absolute gap between the solution returned and the best bound for this scenario was equal to 4.11 and thus still substantial after four hours of computing. For Scenarios 3.3.2 and 4.2, BARON returned slightly worse designs than our VNS algorithm after four hours of computing.

For Scenario 3.1.2 however, BARON produced a design that is 0.1% better in terms of D-optimality than the one produced by the VNS algorithm and shown in Figure 5(a). The absolute gap between this solution and the best bound was 2.52. The reason why BARON was able to find a better design than the VNS algorithm for Scenario 3.1.2 is that it does not use a candidate set. It is therefore free to select design points that do not belong to the candidate set we used for the VNS algorithm. The design produced by BARON is shown in Table 10. It differs only very slightly from the one in Figure 5(a). The design points that differ are those involving proportions that are not multiples of 0.05.

7.5.2. I-optimality

For Scenarios 3.1.1 and 3.3.1, BARON was able to solve our MINLP formulation for finding I-optimal designs to optimality within 8 and 80 minutes, respectively. For these scenarios, it produced the same designs as the VNS algorithm. These designs are shown in Figures 3(b), 3(c) and 7. For none the scenarios we considered, BARON was able to finish the optimization of our MINLP formulation for I-optimal design within four hours. For Scenario 3.2.1, BARON did however return the same I-optimal design as our VNS algorithm, with an absolute gap of 0.05 between the solution and the best bound. For Scenarios 3.3.2 and 4.2, BARON was unable to find a feasible solution within four hours of computing time.

Table 10: D-optimal design for Scenario 3.1.2 produced by BARON.

Number	x_1	x_2	x_3	Multiplicity
1	0	0.5	0.5	2
2	0	0	1	1
3	0	1	0	1
4	0.261	0.739	0	1
5	0.264	0	0.736	1
6	0.975	0.025	0	1
Total	1.5	2.764	2.736	7
Availability	1.5	3	3	7.5

Finally, for Scenarios 3.1.2, 3.2.2 and 4.1, BARON found better designs than the VNS algorithm. The I-optimal designs produced by BARON are shown in Tables 11, 12 and 13. BARON was unable to confirm the optimality of these solutions, as the absolute gaps between these solutions and the best lower bounds were 4.16, 2.88 and 0.83, respectively. The 3-ingredient design in Table 11 is 0.6% better than the one in Figure 5(b) in terms of I-optimality, while that in Table 12 is 0.05% better than the one in Figure 6(b). The reason BARON was able to find a better design for Scenarios 3.1.2 and 3.2.2 than the VNS algoritm is again that it does not use a candidate set. For these scenarios, the improved designs identified by BARON look only marginally different from those produced by the VNS algorithm. This can be seen by comparing Table 11 with Figure 5(b) for Scenario 3.1.2 and by comparing Table 12 with Figure 6(b) for Scenario 3.2.2. Again, the points that differ in the solutions produced by BARON are those involving proportions that are not multiples of 0.05.

The 4-ingredient design in Table 13 is 1.8% better than the one in Table 3 in terms of I-optimality. In this case, the improvement is not due to the fact that the MINLP approach does not use a candidate set, since the points of the design in Table 13 belong to the candidate set we considered in our VNS algorithm. Therefore, in this case, 30 executions of the VNS algorithm did not suffice to find the best possible design for the candidate set used. The most notable difference between the design produced by the VNS algorithm and the one produced by BARON is that the latter involves nine observations while the former involves ten (see Tables 3 and 13).

7.6. Discussion of the MINLP results

In all but one case, whenever BARON was able to produce a feasible solution within four hours of computing time, the VNS algorithm was able to match that solution or to get so close to the optimal solution that there is no practically relevant difference in D- or I-efficiency for an experimenter, within at most a few minutes. Whenever BARON produced a better solution than the VNS algorithm in those scenarios, this was due to its use of ingredient proportions that did not belong to the candidate sets used by the VNS (see Tables 10, 11 and 12). These proportions are generally hard, if not impossible, to implement in practice by experimenters. In

Table 11: I-optimal design for Scenario 3.1.2 produced by BARON.

Number	x_1	x_2	x_3	Multiplicity
1	0	0	1	1
2	0	1	0	1
3	0	0.5	0.5	2
4	0.34	0.646	0.014	1
5	0.341	0.013	0.646	1
6	0.819	0.091	0.09	1
Total	1.5	2.75	2.75	7
Availability	1.5	3	3	7.5

Table 12: I-optimal design for Scenario 3.2.2 produced by BARON.

Number	x_1	x_2	x_3	Multiplicity
1	0	0	1	2
2	0	0.499	0.501	2
3	0	1	0	1
4	0.054	0.464	0.482	1
5	0.464	0.054	0.482	1
6	0.485	0.483	0.032	1
7	0.499	0	0.501	2
8	0.499	0.501	0	1
9	0.5	0.5	0	1
10	1	0	0	1
Total	4	4	5	13
Availability	4	4	5	13

Table 13: I-optimal design for Scenario 4.1 produced by BARON.

Number	x_1	x_2	x_3	x_4	Multiplicity
1	0.2	0.1	0.1	0.6	3
2	0.2	0.1	0.5	0.2	2
3	0.2	0.5	0.1	0.2	2
4	0.5	0.15	0.15	0.2	1
5	0.6	0.1	0.1	0.2	1
Total	2.5	1.75	1.75	3	9
Availability	2.5	6	3	7	18.5

fact, it is not uncommon for experimenters to prefer experimental designs based on candidate sets with practically feasible test combinations. That the MINLP approach produced designs that were generally at best only marginally better than those generated by our VNS algorithm with a candidate set using $h = 20$ implies that this h value is sufficient for the purpose of the optimal design of mixture experiments with ingredient availability constraints in practice.

For one instance, BARON produced a better solution than the VNS algorithm without resorting to irrational proportions. This suggests that 30 executions of the VNS algorithm is not sufficient to guarantee a high-quality solution, even for relatively small instances. Also, the result suggests that, in some instances, there is added value in adding another neighborhood structure to the VNS algorithm, namely a neighborhood structure that replaces two design points with one point from the candidate set. As a matter of fact, the mere replacement of one occurrence of the third design point as well as the fourth design point from the design in Table 3 (which is the one produced by our VNS algorithm) by the point $(0.5, 0.15, 0.15, 0.2)$ yields the design in Table 13 (which is the one produced by BARON).

For two of our problem instances, the BARON solver was unable to produce feasible solutions for our MINLP formulations. Also, for most of the problems where a feasible solution was reported by BARON, its optimality could not be confirmed. We therefore also studied an alternative solver, namely SCIP v5.0 (Gleixner et al., 2018). SCIP produced the same D-optimal designs as BARON for Scenarios 3.1.1, 3.2.1, 3.3.1, 3.2.2 and 4.1, and the same I-optimal designs for Scenarios 3.1.1, 3.2.1 and 3.3.1. For Scenario 4.1, SCIP found an I-optimal design which differed from but was equivalent to the one produced by BARON. For the rest of the scenarios, however, SCIP was unable to produce a feasible solution within four hours of computing time. Therefore, for our MINLP formulations, BARON outperformed SCIP.

8. General discussion and conclusion

In this article, we discussed 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. When starting from a candidate set of design points, the resulting optimal design of experiments problem is a multidimensional knapsack problem. Due to the complex nature of the optimality criteria used in optimal experimental design, the knapsack problem is nonlinear and nonseparable. To solve instances of the new optimal design problem within a reasonable computing time, we proposed a variable neighborhood search 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 usefulness of the new algorithm by means of several examples. The resulting optimal designs are very different from the optimal designs in the literature, constructed 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. We also observed

substantial differences between D- and I-optimal designs. The most striking difference is that the number of runs in the D-optimal designs was higher on various occasions than the number of runs in the I-optimal designs. 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. 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 (2012b).

To evaluate the variable neighborhood search algorithm, we also utilized an exact approach. To this end, we developed MINLP formulations for both D- and I-optimal mixture designs in the presence of ingredient availability constraints and solved the formulations using BARON and SCIP. Solving the MINLP formulations was generally time consuming. In some cases involving a limited number of ingredients, the solvers we used were even unable to produce a feasible solution within four hours. In other cases, the solvers were unable to confirm the optimality of the solutions produced after four hours of computing time. Therefore, the computing time of the MINLP approach is prohibitively large for more challenging problem instances. For one of the smaller instances, the MINLP approach resulted in a substantially better solution than the VNS algorithm. As discussed in Section 7.6, that outcome suggests that we should either use more than 30 iterations of the VNS algorithm and/or add another neighborhood to replace two design points by one candidate point.

As explained in Section 2.5, we opted for a VNS algorithm based on a set of candidate design points. The size of the candidate set increases with the number of ingredients considered and with how close the experimenter wants the candidate design points to be to each other. The size of this set has a major impact on the computing time of the algorithm. Given that we strongly recommend using a fine grid of candidates on the experimental region, the computing time of our algorithm becomes prohibitively large for problems with more than eight ingredients. However, a study of the literature shows that most mixture experiments in the literature involve substantially fewer than eight ingredients (Cornell, 2002; Smith, 2005). So, mixture design problems involving more than eight ingredients, such as those discussed in Piepel et al. (2005) and Fleury et al. (2014), are the exception rather than the rule. Therefore, our variable neighborhood search algorithm will be capable of solving the vast majority of the mixture design problems with ingredient availability constraints that occur in practice. Constructing a set of candidate design points in the presence of multiple multicomponent constraints can be readily done using the XVERT1 algorithm (to determine extreme vertices) and using the CONAEV algorithm (to determine the centroid points up to a specified degree) (Nigam et al., 1983; Piepel, 1988). These algorithms are currently available in commercial design of experiments software, for instance in the JMP v13 software.

Besides the number of candidate design points, the computing time is also impacted by the availability of the ingredients and the model complexity. More complex models require a longer

computation time because the dimension of the information matrix is equal to the number of parameters in the model, and computing the determinant and the inverse of a larger matrix takes more time than computing the determinant and the inverse of a smaller matrix. Therefore, finding an optimal design for a second-order model is more time consuming than finding an optimal design for a first-order model. A larger availability of the ingredients results in an increase of the computing time, because the larger ingredient availability allows for larger designs to be constructed and these are harder to optimize than smaller designs. Finally, computing I-optimal designs is also substantially more time consuming than computing D-optimal designs. This is due to the fact that the update procedures for the D-optimality criterion value are faster than those for the I-optimality criterion value.

Finally, we would also like to point out that it is especially neighbourhood N_3 which absorbs much computing time. This is because that neighborhood has by far the largest size of all four neighborhoods, and our algorithm always ends with a complete exploration of this entire neighborhood. As we have observed on multiple occasions that exploring neighbourhood N_3 produces a substantially better experimental design, we believe the time spent on the exploration of that neighborhood is a worthwhile investment.

The multidimensional knapsack problem we describe arose in the context of the design of mixture experiments, where the experimental factors are proportions of ingredients. Copelli et al. (2018) describe a related design of experiments problem, in which the experimental tests are not all equally expensive and the total budget is fixed. Their problem could be reformulated as a unidimensional knapsack problem with a nonlinear objective function. The authors present an ad-hoc algorithm for their problem, and obtain a design in which inexpensive experimental tests are used more frequently than expensive experimental tests. It would be an interesting topic for future research to investigate to what extent the variable neighborhood search algorithm we present here improves the experimental design obtained by Copelli et al. (2018).

In this paper, we did not impose any upper bound on the number of experimental runs. Therefore, in our problem, the number of runs is implicitly bounded by the availabilities of the ingredients. It is, however, possible to take into account an explicit upper bound on the number of runs too. As a matter of fact, the current algorithm keeps track of the number of runs already (see, for example, Figures 1(b) and 2(b)). As soon as the maximum number of runs allowed is reached, we can instruct the algorithm to avoid neighborhoods N_0 and N_2. In that case, the number of runs will never exceed the upper bound on the number of runs.

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

$$|\mathbf{M}^*| = |\mathbf{M}|\,|\mathbf{D}|\,|\mathbf{D}^{-1} + \mathbf{U}\mathbf{M}^{-1}\mathbf{U}'| \tag{11}$$

and

$$\mathbf{M}^{*-1} = \mathbf{M}^{-1} - \mathbf{M}^{-1}\mathbf{U}'(\mathbf{D}^{-1} + \mathbf{U}\mathbf{M}^{-1}\mathbf{U}')^{-1}\mathbf{U}\mathbf{M}^{-1}, \tag{12}$$

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

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

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 (11) and (12).

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 $|\mathbf{D}| = -1$ for all design changes studied in neighborhood N_2.

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., 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., Tobias, R. D., 2007. Optimum Experimental Designs, with SAS. Oxford: Oxford University Press.

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

Becker, H., Buriol, L. S., 2019. An empirical analysis of exact algorithms for the unbounded knapsack problem. European Journal of Operational Research 277, 84–99.

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

Brent, R. P., 1973. Algorithms for Minimization without Derivatives. Englewood Cliffs, NJ: Prentice Hall.

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

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

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

Copelli, D., Falchi, A., Ghiselli, M., Lutero, E., Osello, R., Riolo, D., Schiaretti, F., Leardi, R., 2018. Sequential "asymmetric" D-optimal designs: a practical solution in case of limited resources and not equally expensive experiments. Chemometrics and Intelligent Laboratory Systems 178, 24–31.

Cornell, J. A., 2002. Experiments with Mixtures: Designs, Models, and the Analysis of Mixture Data, 3rd Edition. New York: Wiley.

Czyzyk, J., Mesnier, M. P., Moré, J. J., 1998. The neos server. IEEE Journal on Computational Science and Engineering 5, 68–75, available at https://neos-server.org/neos/.

D'Ambrosio, C., Martello, S., 2011. Heuristic algorithms for the general nonlinear separable knapsack problem. Computers and Operations Research 38, 505 – 513.

D'Ambrosio, C., Martello, S., Mencarelli, L., 2018. Relaxations and heuristics for the multiple non-linear separable knapsack problem. Computers and Operations Research 93, 79 – 89.

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

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

Dell'Amico, M., Delorme, M., Iori, M., Martello, S., 2019. Mathematical models and decomposition methods for the multiple knapsack problem. European Journal of Operational Research 274, 886–899.

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

Duarte, B. P. M., Granajo, J. F. O., Wong, W.-K., 2019. Optimal exact designs of experiments via mixed integer nonlinear programming. Statistics and ComputingTo appear.

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

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

Fleury, B., Godon, N., Ayral, A., Perret, D., Dussossoy, J.-L., Gin, S., 2014. Development of an experimental design to investigate the effects of R7T7 glass composition on the residual rate of alteration. Procedia Materials Science 7, 193 – 201.

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

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

Garroi, J.-J., Goos, P., 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.

Gleixner, A., Bastubbe, M., Eifler, L., Gally, T., Gamrath, G., Gottwald, R. L., Hendel, G., Hojny, C., Koch, T., Lübbecke, M. E., Maher, S. J., Miltenberger, M., Müller, B., Pfetsch, M. E., Puchert, C., Rehfeldt, D., Schlösser, F., Schubert, C., Serrano, F., Shinano, Y., Viernickel, J. M., Walter, M., Wegscheider, F., Witt, J. T., Witzig, J., 2018. The SCIP Optimization Suite 6.0. Tech. rep., Optimization Online.
URL `http://www.optimization-online.org/DB_HTML/2018/07/6692.html`

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

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

Goos, P., Jones, B., Syafitri, U., 2016. I-optimal design of mixture experiments. Journal of the American Statistical Association 111 (514), 899–911.

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

Goos, P., Vandebroek, M., 2004. Outperforming completely randomized designs. Journal of Quality Technology 36, 12–26.

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

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

Huang, Y., Gilmour, S. G., Mylona, K., Goos, P., 2019. Optimal design of experiments for nonlinear response surface models. Journal of the Royal Statistical Society: Series C (Applied Statistics) 68, To appear.

Jones, B., Goos, P., 2012a. An algorithm for finding D-efficient equivalent-estimation second-order split-plot designs. Journal of Quality Technology 44 (4), 363–374.

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

Kellerer, H., Pferschy, U., Pisinger, D., 2010. Knapsack Problems. Heidelberg: Springer.

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

Kytöjoki, J., Nuortio, T., Bräysy, O., 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.

Liberti, L., Lavor, C., Maculan, N., Marinelli, F., 2009. Double variable neighbourhood search with smoothing for the molecular distance geometry problem. Journal of Global Optimization 43 (2), 207–218.

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

Malaguti, E., Monaci, M., Paronuzzi, P., Pferschy, U., 2019. Integer optimization with penalized fractional values: The knapsack case. European Journal of Operational Research 273, 874–888.

Meyer, R. K., 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., Hansen, P., 1997. Variable neighborhood search. Computers and Operations Research 24, 1097–1100.

Mladenović, N., Petrović, J., Kovačević-Vujčić, V., 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., Marsten, R. E., 1976. An algorithm for nonlinear knapsack problems. Management Science 22, 1147–1158.

Nigam, A. K., Gupta, S. C., Gupta, S., 1983. A new algorithm for extreme vertices designs for linear mixture models. Technometrics 25, 367–371.

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

Piepel, G. F., 1988. Programs for generating extreme vertices and centroids of linearly constrained experimental regions. Journal of Quality Technology 20, 125–139.

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

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

Ruseckaite, A., Goos, P., Fok, D., 2017. Bayesian D-optimal choice designs for mixtures. Journal of the Royal Statistical Society: Series C (Applied Statistics) 66 (2), 363–386.

Sahinidis, N. V., 2017. BARON 17.8.9: Global Optimization of Mixed-Integer Nonlinear Programs, *User's Manual*.
URL `http://www.minlp.com/downloads/docs/baron%20manual.pdf`

Sartono, B., Goos, P., Schoen, E., 2015. Constructing general orthogonal fractional factorial split-plot designs. Technometrics 57, 488–502.

Schauer, J., 2016. Asymptotic behavior of the quadratic knapsack problem. European Journal of Operational Research 255, 357–363.

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, Kyushu University, General Education Department.

Vazquez, A. R., Goos, P., Schoen, E. D., 2019. Constructing two-level designs by concatenation of strength-3 orthogonal arrays. Technometrics 61, 219–232.

Vo-Thanh, N., Jans, R., Schoen, E. D., Goos, P., 2018. Symmetry breaking in mixed integer linear programming formulations for blocking two-level orthogonal experimental designs. Computers and Operations Research 97, 96 – 110.

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

Wang, H., Kochenberger, G., 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.