A Branch-and-Bound Algorithm for the Exact Optimal Experimental Design Problem

Selin Damla Ahipaşaoğlu
Mathematical Sciences, University of Southampton

August 11, 2021

Abstract

We discuss a generalisation of the approximate optimal experimental design problem, in which the weight of each regression point needs to stay in a closed interval. We work with Kiefer's optimality criteria which include the well-known D- and A-optimality as special cases. We propose a first-order algorithm for the generalised problem that redistributes the weights of two regression points in each iteration. We develop a branch-and-bound algorithm for exact optimal experimental design problems under Kiefer's criteria where the subproblems in the search tree are equivalent to the generalized approximate design problem, and therefore, can be solved efficiently by the first-order method. We observe that our branch-and-bound algorithm is favourable to a popular exchange heuristic for certain problem instances.

1 The Exact Optimal Design Problem

We consider the following linear model

$$y = x^T(t)\theta + \epsilon(t), \tag{1}$$

where components of $x^T(t) = (x_1(t), x_1(t), \dots, x_n(t))$ are n linearly independent continuous functions on some compact space and $\theta \in \mathbb{R}^n$ is a vector of unknown parameters to be estimated. We assume that observations are conducted independently and the error terms $\epsilon(t)$ follow a multivariate distribution with mean 0 and standard deviation σ. Without loss of generality, we work with the following model function

$$y = x^T\theta + \epsilon, \tag{2}$$

in which the dependency of the vector $x(t)$ on the actual experimental conditions t is suppressed. We suppose that m regression points that span $\mathbb{R}^n$ are already selected and denote this set as $\mathcal{X} := \{x_1, \dots, x_m\} \subset \mathbb{R}^n$.

Definition 1.1 *An* experimental design of size N *is given by a finite number of regression points* $x_1, \ldots, x_m$ *in* $\mathbb{R}^n$ *and nonnegative integers (representing the number of repetitions at each respective point)* $n_1, \ldots, n_m$ *such that* $\sum_{i=1}^m n_i = N$. *The* support *of an experimental design is the set of regression points on which at least one experiment is conducted, i.e.,* x_i *is in the support if* $n_i > 0$.

The *Exact Optimal Experimental Design* problem aims to find integers n_i so that the *information matrix*, defined as $M := \frac{N}{\sigma^2} \sum_{i=1}^m \frac{n_i}{N} x_i x_i^T$, is maximized with respect to an information criterion. The most common approach is to use Kiefer's Φ_p criteria (see [19]), which uses the p^{th} *matrix mean* of the *information matrix*, defined over the cone of positive definite matrices as follows:

$$\Phi_p(C) = \begin{cases} \lambda_{\max}(C) & \text{for} \quad p = \infty, \\ \left(\frac{1}{n}\text{Trace} C^p\right)^{1/p} & \text{for} \quad p \neq 0, \pm\infty, \\ (\det C)^{1/n} & \text{for} \quad p = 0, \\ \lambda_{\min}(C) & \text{for} \quad p = -\infty. \end{cases}$$

Well studied examples of Kiefer's criteria are: (1) $\Phi_0(M) = \det(M)$ (D-optimality); (2) $\Phi_{-1}(M) = (\text{Trace}(M^{-1}))^{-1}$ (A-optimality); (3) $\Phi_{-\infty}(M) = \lambda_{\min}(M)$ (E-optimality). Note that the information matrix is the inverse of the dispersion matrix which is a generalized measure of the variance (or the error) of the least squares estimators of the model parameter θ. Therefore, maximizing an information criterion is equivalent to minimizing a function of the estimation error of the linear model built on the data of the experiment. Information matrices play a central role in experimental design and estimation in general. The use of matrix means is advantageous since they satisfy the necessary conditions to be an information function. They map information matrices to nonnegative real numbers, they are positively homogenous, superadditive, nonconstant, and upper semicontinuous. The p^{th} matrix mean of an information matrix and its derivative can be calculated efficiently for a given design. In addition, for most common values of p, they can be updated efficiently if a new experiment is added or the weights of a few of the experiments in a design are changed after an initial design is calculated.

Let us define $u_i := \frac{n_i}{N}$ and $M(u) := \sum_{i=1}^m u_i x_i x_i^T$, then the following *integer nonlinear program* can be used to calculate an exact optimal experimental design with Kiefer's Φ_p criteria:

$$\begin{array}{rrcl} \max & g_p(u) & := & \ln \Phi_p(M(u)) \\ s.t. & \sum_{i=1}^m u_i & = & 1, \\ & u_i & \geq & 0, \quad \forall i, \\ & u_i N & \in & \mathbb{Z}, \quad \forall i. \end{array} \tag{3}$$

Problem (3) has mostly been studied for the D-criterion, while the interest in solving the exact design problem for other criteria has been quite limited. In particular, the exact D-optimal experimental design problem is shown to be NP-hard in [37]. Recently, [24] has shown that A-optimal experimental design is also NP-hard. To the authors' knowledge, there are two main approaches to solve this problem exactly:

1. **Branch-and-Bound Algorithms:** The first B&B procedure was devised in [38] for the D-criterion and [18] has proposed a similar procedure for a generalization of the D-criterion.

2. **Commercial Solvers:** It is recently demonstrated in [27] that Problem (3) can be represented as a Mixed Integer Second Order Conic Program for $p = 0$ and $p = -1$, and therefore, can be solved by commercial MISOCP solvers.

On contrast to the exact methods, the methods listed below aim to find near-optimal feasible solutions to Problem (3). Someo f the inexact methods rely on solving a relaxation of the problem, namely the *Approximate Optimal Experimental Design* problem, where integrality constraints are omitted. As its integer restriction, the relaxed design problem is also extensively studied for the D-criterion, while a smaller number of studies focus on the A-criterion (See [1, 5, 7, 12, 17, 39, 41] for D-optimality and [2, 17] for A-optimality.)

1. **Rounding fractional solutions:** Once an optimal design for the relaxed problem is found, the number of experiments at each design point can be rounded to the closest integer. Unfortunately, this will not necessarily lead to an exact design with N experiments, and the solutions need to be modified accordingly, which may be done in a number of heuristic ways. One can refer to Page 157 in [12], Chapter 12 in [25] or [32] for a valuable discussion on how to come up with an exact experimental design for a finite sample size once the optimal design for an infinite sample size is found. A more sophisticated and efficient procedure is provided in [26].

2. **Heuristics:** Exchange heuristics in which an exact design is improved by exchanging the weights of two design points is commonly used in the literature. There are several variants of such methods (e.g., [4, 11, 12, 22, 23]) depending on how the points of exchange are determined, while the KL-exchange algorithm of [4] is particularly popular. These methods can be used to improve the solutions obtained by other inexact methods or used on their own. Recently, several metaheuristics have also been used for both the approximate and exact experimental design problems. (See [15] and the references therein for a recent survey of such methods.)

3. **Quadratic Approximations:** Integer programming methods based on the quadratic approximation of the design criterion in the neighbourhood of the optimal approximate information matrix is proposed in [16] for $p = 0$ and improved and extended to general integer p including $p = 0$ and $p = -1$ in [14] . These methods are well suited for large problem instances of the problem and advantegous over the rounding algorithms discussed above as demonstrated in [14]. They are especially useful if there are linear constraints on the design weights.

4. **Approximation Algorithms:** For $p = 0$, a series of approximation algorithms based on various convex relaxations of the problem have recently been proposed in [8], [6] [36], [3], [29], and finally [21] proved

that a simple local search algorithm (known as the Fedorov Exchange method) finds an ϵ-approximate solution for $N \geq n + \frac{n}{\epsilon}$. These approximation algorithms also work for $p = -1$. In addition, [24] provided an ϵ-approximation for $N \geq \frac{(1+\epsilon)(n-1)}{\epsilon}$. This paper also shows that finding a constant approximation when $N = n$ is NP-hard.

In common settings where experimental design is heavily used, such as clinical trials in medicine, the cost of each experiment is high, therefore, the number of actual experiments that will be conducted is moderate. This makes the assumption $N \to +\infty$ used in approximate design undesirable. Interestingly, the experimental design community, even those who collaborate closely with medical research groups, seem to use the inexact designs mentioned above with the belief that exact methods are out of reach for all but the smallest problem instances. We will demonstrate in the computational studies below that although inexact methods are necessary for large scale problems, they can and should be replaced by an exact method for smaller sized problems.

The main goal of this paper is to describe a Branch-and-Bound procedure for the Exact Optimal Experimental Design Problem given in (3) for any p and demonstrate how to implement it efficiently to decrease the number of branches explored in the search tree and also to decrease the solution time of each subproblem by an efficient first-order algorithm that makes use of the information calculated at the parent notes. We will demonstrate that this customized branch and bound procedure beats the exchange heuristics for certain instances of the problem.

Our work is related to several papers some of which are introduced in the discussion above. Here we would like to highlight the contributions in this paper compared to these existing work.

1. First, we develop a branch-and-bound algorithm that has a different branching strategy than those in [18] and [35]. These papers build a search tree by conditioning on the value of a given variable, i.e., each subtree considers solutions where a set of variables are fixed already. In particular, subproblems that are used in the later study are instances of approximate optimal design problems in which only a subset of design points are considered, the weights of the rest of the design points are either 0 or $1/N$ as these points have already been used to branch-on. In contrast, we build a general non-binary search tree. Each subproblem is constructed by adding a lower and upper bound on the weight of a chosen design point. Therefore it is equal to an approximate optimal design problem with bounds on the design weights. The structure of the search tree and the subproblems in our framework are instead similar to those used in [38]. Nevertheless, we solve the subproblems more efficiently as we discuss in detail in the next section.

2. Our methods apply to exact experimental design problems with Kiefer's Φ_p criteria in general; while existing literature consider a few integer values of p such as 0 and -1. In particular, our framework can be applied

to any value of p even though obvious sacrifices are needed in computational effort for $p \notin \{0, -1\}$. We argue that these neglected values are useful in situations where robustness of the design with respect to different criteria is desired. For example, an optimal design with p for some value of $p \in (-1, 0)$ may provide a good a trade-off between D-optimal or A-optimal designs.

3. The approximation algorithms with explicit guarantees as well as methods based on quadratic approximations mentioned above work well for large values of N, i.e., when the number of experiments is large compared to the number of parameters in the model. The branch-and-algorithm will still be very expensive in these situations; nevertheless in common applications, such as clinical research, the number of experiments that can be afforded is more modest and within the reach of the branch-and-bound algorithm given in this paper. These 'medium-size' instances are still too large for the commercial integer programming solvers and benefit from our customized treatment.

4. Although various extensions and generalizations of the D-optimal approximate design problem have been studied in the literature, according to our current knowledge, the *box-constrained approximate design problem with Kiefer's Φ_p criteria* proposed and solved in Section 3 is not studied in detail before. In particular, the box-constrained problem is a special case of experimental design problems with linear constraints studied in [9, 10, 27] and theory and solution techniques developed in these references apply here. Similarly, although there are related results in [13, 33, 34, 35], the equivalence theorem for this specific version of the problem (see Theorem 3.1 below) is novel and will be useful in future studies, potentially for different applications.

In Section 2, we will present the main steps of the B&B algorithm and introduce the subproblems to be solved in each branch of the tree. The subproblems correspond to a generalization of the Approximate Optimal Experimental Design in which the weights of the design points belong to a given interval. We will call this problem as the *box-constrained approximate design problem (with Kiefer's Φ_p criteria)*. In Section 3, we study the generalized approximate design problem by introducing its dual and the necessary and sufficient optimality conditions that must be satisfied at its optimal solution. An efficient algorithm to solve this problem will be provided in Section 4, where we will also demonstrate how this algorithm can be tailored for specific criteria, especially for $p = 0$ and $p = -1$. We will discuss how to implement the B&B algorithm efficiently in Section 5 and provide computational results in Section 6. We close by concluding remarks and further research questions in Section 7.

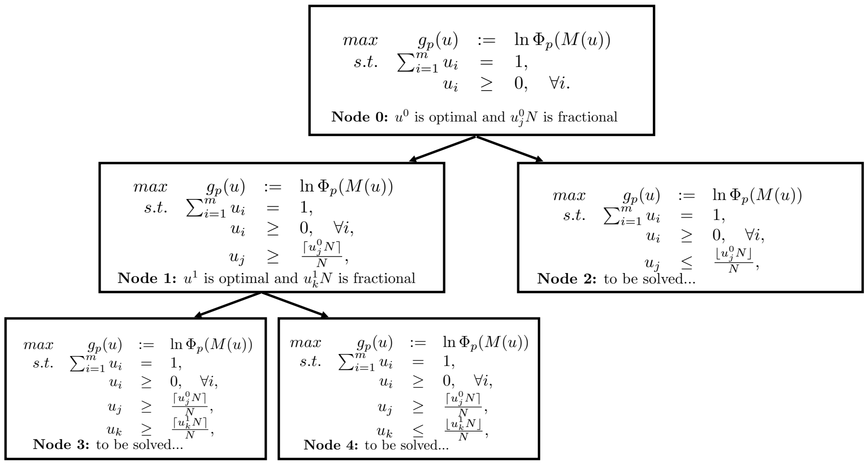
$max \quad g_p(u) := \ln \Phi_p(M(u))$
$s.t. \quad \sum_{i=1}^m u_i = 1,$
$u_i \geq 0, \quad \forall i.$
Node 0: u^0 is optimal and $u_j^0 N$ is fractional
$u_j \geq \frac{\lceil u_j^0 N \rceil}{N},$
Node 1: u^1 is optimal and $u_k^1 N$ is fractional
$u_j \leq \frac{\lfloor u_j^0 N \rfloor}{N},$
Node 2: to be solved...
$u_k \geq \frac{\lceil u_k^1 N \rceil}{N},$
Node 3: to be solved...
$u_k \leq \frac{\lfloor u_k^1 N \rfloor}{N},$
Node 4: to be solved...

Figure 1: B&B search tree

2 A Branch and Bound Algorithm for Φ_p criteria

As usual in B&B algorithms for maximization problems, we start with calculating a lower bound on the optimal value. A feasible solution, and therefore a lower bound, can be obtained by any of the inexact methods mentioned above. It is very important to start with a good lower bound since this dictates the number of branches explored in the tree. We will present the algorithm of our choice in Section 5. Once we have a lower bound, we start searching for a better solution using a search tree (an example is given in Figure 1). At node 0, we solve the relaxation of Problem (3) for the given p to obtain its optimal solution, say u^0. If $u^0 N$ is not integer, we choose an index j such that $u_j^0 N$ is fractional and create two subproblems (Nodes 1 and 2 in Figure 1) by adding either $u_j \geq \frac{\lfloor u_j N \rfloor}{N}$ (Node 1) or $u_j \leq \frac{\lceil u_j N \rceil}{N}$ (Node 2) as an additional constraint to the approximate design problem in Node 0. We refer Node 0 as the parent node of Nodes 1 and 2. Next we solve one of the subproblems, say Node 1, to obtain its optimal solution, say u^1. If the objective function at u^1 is less than or equal to the lower bound, we stop investigating this branch as there cannot be a better solution in the subtree below Node 1. We refer to this as *fathoming*. Otherwise, there are two options. First, $u^1 N$ may be integral. In this case, we update the lower bound since the objective function at u^1 is greater than the current lower bound, and fathom the branch. Second, when $u^1 N$ is fractional, we choose an index k such that $u_k^1 N$ is fractional and branch the problem further using index k. This creates two new subproblems (Nodes 3 and 4 in Figure 1) to be explored at a later stage. We continue our search on the tree in this manner until all branches are fathomed, i.e., all subproblems are investigated.

At each node of the search tree, we solve the following *box-constrained approximate design problem with Keifer's Φ_p criteria*,

$$\begin{array}{lrcl} \textit{max} & g_p(u) & := & \ln \Phi_p(M(u)) \\ \textit{s.t.} & \sum_{i=1}^m u_i & = & 1, \\ & \alpha_i \quad \leq u_i & \leq & \beta_i, \quad \forall i, \end{array} \tag{4}$$

where $\alpha \leq \beta \in [0,1]^m$. For $p = 0$ or $p = -1$, this problem can be solved efficiently when $\alpha_i = 0$ and $\beta_i = 1$, for all i, by one of the methods in [1], [2] , [17], [41]. For other values of α and β, the first order-methods presented in these papers would fail since they all exploit the fact that the solution lies in the unit simplex and therefore cannot deal with linear constraints even though they are very simple. Although conic formulations can be adapted to include linear constraints, we will solve the subproblems by a first-order method which is tailored to our specific needs, since solving the conic formulations is significantly more expensive than our approach as discussed in Section 6.2 below. We will study the subproblem in detail in the next section before we present the first-order method in Section 4.

3 The Box-Constrained Approximate Design Problem with Kiefer's Φ_p Criteria

Let q be the conjugate of p, i.e., $p, q \in (-\infty, 1]$ and $pq = p + q$, and consider also the following problem:

$$\begin{array}{lrcl} \min & f_q(H) & = & -\ln \Phi_q(H) \\ \textit{s.t.} & x_i^T H x_i + \lambda_i - \bar{\lambda}_i & = & n + \lambda^T \alpha - \bar{\lambda}^T \beta, \quad \forall i, \\ & \lambda_i, \bar{\lambda}_i & \geq & 0, \quad \forall i, \\ & H & \succ & 0, \end{array} \tag{5}$$

where $\alpha \in \mathbb{R}^m$ and $\beta \in \mathbb{R}^m$ are given lower and upper bounds on the weights of design points and $\lambda \in \mathbb{R}^m$ and $\bar{\lambda} \in \mathbb{R}^m$ are auxilary variables whose meaning will be apparent in the proof of the following theorem. This theorem will prove that problems (4) and (5) are dual to each other, and therefore solving one to optimality provides a solution for the other. This is a generalization of the known primal-dual relationship between the approximate D-optimal experimental design and the minimum volume enclosing ellipsoid (MVEE) problem. (See [28] for an early discussion of duality of these problems with D-criterion and [2] for Kiefer's criteria in general.) This relationship has been exploited many times in the optimization literature as some of the most efficient methods to solve the MVEE problem rely on solving its dual via first-order algorithms such as in [1, 20, 31].

Theorem 3.1 *We have $f(H) \geq g(u)$ for any u and $(H, \lambda, \bar{\lambda})$ feasible for (4) and (5), respectively. Furthermore, the following conditions together with feasibility w.r.t. problem constraints are both necessary and sufficient for optimality of u and $(H, \lambda, \bar{\lambda})$ for (4) and (5), respectively.*

1. *$H = \frac{n}{\text{Trace}(M(u))^p}(M(u))^{p-1}$;*

2a. *if $i \in A$, then $\omega_i(u) \leq n^*$, or equivalently, $\lambda_i = n^* - \omega_i(u)$ and $\bar{\lambda}_i = 0$,*

2b. *if $i \in B$, then $\omega_i(u) \geq n^*$, or equivalently, $\lambda_i = 0$ and $\bar{\lambda}_i = \omega_i(u) - n^*$,*

2c. *otherwise, $\omega_i(u) = n^*$, or equivalently, $\lambda_i = 0$ and $\bar{\lambda}_i = 0$,*

where $\omega_i(u) = x_i^T H x_i$, $A = \{i | u_i = \alpha_i\}$, $B = \{i | u_i = \beta_i\}$, and $n^* := \frac{n + \sum_{i \in B} \beta_i \omega_i - \sum_{i \in A} \alpha_i \omega_i}{1 - \sum_{i \in A} \alpha_i + \sum_{i \in B} \beta_i}$.

Proof: For any feasible u and $(H, \lambda, \bar{\lambda})$, we have

$$\begin{aligned}
\sum_{i=1}^{m} u_i(x_i^T H x_i + \lambda_i - \bar{\lambda}_i) &= n + \lambda^T \alpha - \bar{\lambda}^T \beta \\
\sum_{i=1}^{m} u_i(x_i^T H x_i) &= n - \sum_{i=1}^{m} \bar{\lambda}_i(\beta_i - u_i) - \sum_{i=1}^{m} (\lambda_i(u_i - \alpha_i) \\
\text{Tr}(HM(u)) &\leq n. \qquad (6)
\end{aligned}$$

This, together with an application of the Hölder's inequality (on the eigenvalues of the matrices at hand, see [25] for a detailed proof), proves weak duality:

$$\begin{aligned}
f_q(H) - g_p(u) &= -\ln \Phi_q(H) - \ln \Phi_p(M(u)) \\
&= -\ln(\Phi_q(H)\Phi_p(M(u))) \\
&\geq -\ln\left(\frac{1}{n}\text{Trace}(HM(u))\right) \qquad (7) \\
&\geq -\ln 1 = 0. \qquad (8)
\end{aligned}$$

Inequalities (7) and (8) hold as equality when the conditions (1, 2a-c) given in the theorem are satisfied. This proves sufficiency, necessity follows from the KKT conditions for (5). $\square$

Note that for $p = 0$, $\alpha_i = 0$ and $\beta_i = 1$, for all i, (4) and (5) reduce to the approximate D-optimal design problem and the Minimum Volume Enclosing Ellipsoid problem, respectively. The duality of these problems together with necessary and sufficient conditions for optimality have already been established for D-optimality as well as under the more general Kiefer's optimality criteria. (See [1], [2], and [25] for details.) Similar to these results, the optimal solution H^* for the dual problem (5) also correspond to an ellipsoid centered at the origin. This ellipsoid covers all design points whose weights are lower than their upper bounds, while the design points whose weights are set to their upper bounds may lie outside of the ellipsoid. The design points whose weights lie between the lower and upper bounds (referred to as the *critical points* and

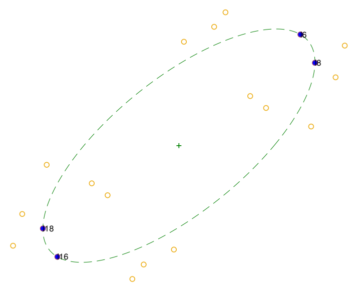

Figure 2: Optimal solution of a problem with $p = q = 0$ in $\mathbb{R}^2$.

marked with blue in Figure 2) lie on the surface of an ellipsoid whose shape matrix is proportional to the optimal dual solution H^*. Note that the ellipsoidal distance of the critical points is equal to n^* that is a function of the optimal primal solution u^*. Theorem 3.1 will play an important role in developing a first-order algorithm below, as it provides the necessary criteria to stop the algorithm.

4 A First-Order Algorithm for the Box-Constrained Approximate Design Problem

We now propose a simple iterative procedure that solves Problem (4). The outline of the algorithm is presented in Algorithm 4.1. Note that the input of the algorithm is not necessarily a feasible point, and therefore, the first step is to find a feasible point. Since the feasible region is the intersection of two convex sets, the unit simplex and the hyperrectangle defined by the box constraints, a feasible point can be found by the alternative projections method where solutions are projected onto these two sets alternatively until a feasible solution is obtained. After an initial feasible solution is found, Algorithm 4.1 preserves primal feasibility in each iteration by updating the current iterate, say u, as follows:

$$u_+ = u + \theta e_j - \theta e_k, \tag{9}$$

where j, k represent two distinct indices whose design weights are to be respectively increased and decreased by an amount θ and e_j and e_k are the j^{th} and k^{th} unit vectors, respectively. Note that this leads a 2-rank update of the information matrix, i.e.,

$$M(u_+) = M(u) + \theta x_j x_j^T - \theta x_k x_k^T. \tag{10}$$

Therefore, we will be able to update the objective function and its gradient in an efficient way. The following lemma will be useful in choosing the indices

j and k so that the iterative algorithm improves the objective function in each iteration.

Lemma 4.1 *Let u be a feasible solution and also $j \in \arg\max\{\omega_i : u_i < \beta_i\}$ and $k \in \arg\min\{\omega_i : u_i > \alpha_i\}$. If $\omega_j \le \omega_k$, then u is optimal. Otherwise, $d = e_j - e_k$ is a feasible ascent direction, i.e., $g_p(u_+) > g_p(u)$.*

Proof: Assume that $\omega_j \le \omega_k$. Case 1: $\{i : \alpha_i < u_i < \beta_i\} \neq \emptyset$, then we have

$$\begin{aligned}\omega_k &= min\{\omega_i : \alpha_i < u_i \le \beta_i\} \le min\{\omega_i : \alpha_i < u_i < \beta_i\}\\ &\le max\{\omega_i : \alpha_i < u_i < \beta_i\} \le max\{\omega_i : \alpha_i \le u_i < \beta_i\} = \omega_j.\end{aligned}$$

This, together with $\omega_j \le \omega_k$, implies that $\omega_j = \omega_k$ and therefore all inequalities must be equalities and conditions (1, 2a-c) in Theorem 3.1 must hold. Case 2: $\{i : \alpha_i < u_i < \beta_i\} = \emptyset$. In this case, since $\omega_j \le \omega_k$, we can always find an n^*, which would satisfy the conditions (1, 2a-b) in Theorem 3.1 and Condition (2c) is void. Therefore, whenever $\omega_j \le \omega_k$, we can conclude that u is optimal. Otherwise, since $\nabla g_p(u)^T d = \omega_j - \omega_k > 0$, $u_j < \beta_j$ and $u_k > \alpha_k$, d is a feasible ascent direction. Hence, we can always find a positive θ such that $g_p(u_+) > g_p(u)$. □

In each iteration, we solve the following single variable maximization problem to determine the opimal step size θ^*:

$$\begin{array}{rl}\max_\theta & g_p(u + \theta e_j - \theta e_k)\\ s.t. & 0 \le \theta \le \beta_j - u_j,\\ & 0 \le \theta \le u_k - \alpha_k.\end{array} \tag{11}$$

For some values of p, this problem is easier to deal with as we will discuss shortly below. For general p, we can apply a simple line search or Armijo's rule to determine the step size. In either case, the algorithm is guaranteed to converge to the optimal solution. After the step size is chosen, we update u as in (9) and the gradient $\omega(u)$, and check if the following optimality criterion is satisfied: $\frac{\omega_j}{\omega_k} - 1 \le tol$, where tol is a given small positive number. Note that when $tol = 0$, this condition guarantees $\omega_j \le \omega_k$, and therefore, optimality of the current iterate, but we stop once a predetermined level of accuracy is achieved. When $p = 0$ and there are no constraints on the design weights, i.e., $\alpha_i = 0$ and $\beta_i = 1$, for all i, then Algorithm 4.1 is similar to the vertex-exchange algorithm of [7].

Algorithm 4.1

```
Require: $X, u^0, \alpha, \beta, tol$
  Calculate $u$, the projection of $u^0$ onto the feasible set $\{\alpha \le u \le \beta, \sum u_i = 1\}$.
  Calculate $M(u)$ and $\omega(u)$.
  Find $j := \arg\max\{\omega_i : u_i < \beta_i\}$ and $k := \arg\min\{\omega_i : u_i > \alpha_i\}$.
  while $\frac{\omega_j}{\omega_k} - 1 > tol$ do
    Replace $u$ by $u_+$ in (9), where $\theta^*$ is the maximizer for Problem (11).
    Update $M(u)$ and $\omega(u)$.
  end while
  return $u$ and $g(u)$.
```

4.1 D-optimality

Let us assume that $p = 0$, in this case we have $\omega_i = x_i^T M(u)^{-1} x_i$. Let also $\omega_{jk} = x_j^T M(u)^{-1} x_k$. The change in the objective value can be expressed as a function of θ by an application of the matrix determinant lemma and the Woodbury Sherman Morrison Identity for matrix inverses:

$$\begin{aligned}
\det(M(u_+)) &= \det\left(M(u) + \theta x_j x_j^T - \theta x_k x_k^T\right) \\
&= \det\left(M(u) + \theta x_j x_j^T\right)\left(1 - \theta x_k (M(u) + \theta x_j x_j^T)^{-1} x_k\right) \\
&= \det(M(u)) + (1 + \theta x_j^T M(u)^{-1} x_j) \dots \\
&\quad \dots \left(1 - \theta x_k (M^{-1}(u) - \frac{\theta}{1 + \theta x_j^T M(u)^{-1} x_j} M(u)^{-1} x_j x_j^T M(u)^{-1}) x_k\right) \\
&= \det(M(u)) + (1 + \theta\omega_j)\Big(1 - \theta x_k \Big(M^{-1}(u) - \frac{\theta}{1 + \theta\omega_j} M(u)^{-1} x_j x_j^T M(u)^{-1}\Big) x_k\Big) \\
&= \det(M(u)) + (1 + \theta\omega_j)\Big(1 - \theta\omega_k + \frac{\theta^2}{1 + \theta\omega_j}\omega_{jk}^2\Big) \\
&= \det(M(u)) + (1 + \theta\omega_j)(1 - \theta\omega_k) + \theta^2\omega_{jk}^2,
\end{aligned}$$

which gives

$$g(u_+) = g(u) + \ln\left(\theta^2(\omega_{jk}^2 - \omega_j\omega_k) + \theta(\omega_j - \omega_k) + 1\right).$$

Hence, we can find the optimal solution θ^* for Problem (11) exactly as follows:

$$\theta^* = \begin{cases} \min\{\beta_j - u_j, u_k - \alpha_k\}, & \text{if } \omega_{jk}^2 = \omega_j\omega_k, \\ \min\{\beta_j - u_j, u_k - \alpha_k, \bar{\theta}\}, & \text{if } \omega_{jk}^2 < \omega_j\omega_k, \end{cases}$$

where $\bar{\theta} = \dfrac{\omega_j - \omega_k}{2(\omega_j\omega_k - \omega_{jk}^2)}$.

It is easy to see that θ^* is the optimal solution. There are two cases: First, we may have $\omega_{jk}^2 = \omega_j\omega_k$, then the objective function of (11) is an increasing linear function, and therefore $\theta^* = \min\{\beta_j - u_j, u_k - \alpha_k\}$ is the optimal solution. Else, a simple application of Cauchy-Schwarz gives $\omega_{jk}^2 \le \omega_j\omega_k$, and hence the objective function is a negative quadratic with root $\bar{\theta}$. Since the problem constraints limit θ to $\min\{\beta_j - u_j, u_k - \alpha_k\}$, the maximizer is θ^*.

4.2 A-optimality

Although, it is more complicated than the D-criterion, we can also derive an expression for the objective function when $p = -1$. First let $E = M(u) + \theta x_j x_j^T$, then Woodbury Sherman Morrison Identity for matrix inverse gives

$$\begin{aligned} E^{-1} &= M(u)^{-1} - \frac{\theta M(u)^{-1} x_j x_j^T M(u)^{-1}}{1 + \theta x_j^T M(u)^{-1} x_j} \\ &= M(u)^{-1} - \frac{\theta}{1 + \theta \omega_j} M(u)^{-1} x_j x_j^T M(u)^{-1}, \end{aligned}$$

and $\text{Trace}(E^{-1}) = \text{Trace}(M(u)^{-1}) - \frac{\theta \zeta_j}{1+\theta\omega_j}$ where $\zeta_i = x_i^T M(u)^{-2} x_i$. Let also $\zeta_{jk} = x_j^T M(u)^{-2} x_k$. We can now calculate the objective function at the next iterate as a function of θ as follows:

$$\begin{aligned} \text{Trace}(M(u_+)^{-1}) &= \text{Trace}(A^{-1}) + \frac{\theta}{1 - \theta x_k^T E^{-1} x_k} \text{Trace}(E^{-1} x_k x_k^T E^{-1}) \\ &= \text{Trace}(A^{-1}) - \frac{\theta A + \theta^2 B}{1 + \theta C - \theta^2 D}, \end{aligned}$$

where $A := \zeta_j - \zeta_k$, $B := 2\omega_{jk}\zeta_{jk} - \omega_j \zeta_k - \omega_k \zeta_j$, $C := \omega_j - \omega_k$ and $D := \omega_j \omega_k - \omega_{jk}^2$. Now we have closed form expression for $g(u_+)$ and can calculate the optimal value of θ for which $g(u_+)$ is maximized as follows (see Proposition 1 in [17]):

$$\theta^* = \begin{cases} \min\{\max\{-\frac{B+\sqrt{B^2 - A\Delta}}{\Delta}, u_j\}, u_k\}, & \text{if } \Delta \neq 0, \\ \min\{\max\{-\frac{A}{2B}, u_j\}, u_k\}, & \text{if } \Delta = 0 \text{ and } B \neq 0, \\ u_k, & \text{if } \Delta = B = 0 \text{ and } A > 0, \end{cases}$$

where $\Delta := AD + BC$.

5 Implementation of the B&B algorithm

We believe that the success of the algorithm depends on a number of choices made during its implementation. We list below the most important ones according to our practical experience. Items 2 and 3 below are commonly used for branch and bound procedures and known to be effective.

1. Initial lower bound on the objective function: Any of the inexact methods listed in Section 1 can be used to find a feasible design, and hence a lower bound, for the exact design problem (3). We use a simple rounding up procedure as follows: Solve the relaxed problem to obtain the optimal approximate design say $\tilde{u}$. If $N\tilde{u}$ is not integer, round each component $N\tilde{u}_j$ to the closest integer. If the total number of experiments is less than N, i.e., $N \sum_j \tilde{u}_j < N$, add one experiment to the design point in the support of $\tilde{u}$ with the least number of experiments; if the total number of experiments is more than N, i.e., $N \sum_j \tilde{u}_j > N$, take away one experiment from the design point with most number of experiments. Repeat the process until the total number of experiments is exactly N.

2. Choice of indices to branch: We chose the index of the variable to branch on according to the *most fractional variable rule*, i.e., we branch on the variable with fractional part closest to 0.5. (Note that for variable u_j the fractional part is defined as $\min\{u_jN - \lfloor u_jN \rfloor, \lceil u_jN \rceil - u_jN\}$.)

3. Order of the subproblems to be solved: We maintain a priority queue on the subproblems to be solved, where the subproblems are sorted in descending objective function values of their parent nodes. By doing so, we hope to solve 'more promising' subproblems first and are able to 'cut the tail of the queue' whenever a new lower bound is obtained.

4. Fast calculation of step-sizes and gradients (while solving the subproblems): In our opinion, the two key advantages of the first-order algorithm for the subproblem are the availability of a fast update for the gradient of the new iterate and the analytical formulae available for the optimal step size for the most important p values (i.e., for D-optimality and A-optimality). In particular, the calculation of the gradient and the step size in each iteration of the algorithm requires the current values of $\omega_{jk}(u)$ and $\zeta_{jk}(u)$. The most efficient efficient way of calculating these quantities is to update the Cholesky factors of $M(u)^{-1}$ and $M(u)^{-2}$ from their previous values, respectively. Since each iterate is a rank-2 update of the previous one, we employ two rank-1 updates in a row to update these factors efficiently.

5. Information stored for each parent node and inherited by its descendants: The solution at the parent node is stored and used as an initial point for the subproblems branching from the parent. These initial points are infeasible but they are very close to the feasible region. Since the feasible region of each subproblem is the intersection of the unit simplex with a hyperrectangle, we use the alternating projections method to project the infeasible solution onto the feasible region of the subproblem.

6 Computational Results

6.1 Experimental Setup

All experiments described in the rest of this section were carried out on a Intel(R) Xeon(R) W-2012 2.90GHz processor with 16 GB RAM using MATLAB version R2019b. We have used the following two methods to generate our datasets.

First-degree linear regression with randomly generated regressors: First, we consider a simple first-degree linear regression model as follows:

$$E(y) = \sum_{j=1}^{n} \theta_j x_j,$$

and generate a set of random regression points following the experiments in [1, 30]. We refer to these as the SFrandom data sets. These sets are generated by mixing points from several independent Gaussian distributions, in order to mimic the data points from one or more clusters as might be encountered in practice. We use these to study the performance of the first-order algorithm (Algorithm 4.1) discussed in Section 4. These data sets are usually more challenging (for the approximate design problem) than the model based data sets used in statistics literature. Therefore, they are especially useful for benchmarking algorithms for the approximate D- and A- optimal designs as well as their generalization discussed in this paper.

Second-order response surface in f factors: We assume the model as in:

$$E(y) = \theta_0 + \sum_{j=1}^{f} \theta_j x_j + \sum_{j=1}^{f-1} \sum_{k=j+1}^{f} \theta_{jk} x_j x_k + \sum_{j=1}^{f} \theta_{jj} x_j^2,$$

and generate regressors from a 3^f factorial design, i.e., we let $x_j \in \{-1, 0, 1\}$ for each j.

6.2 Performance of the First-Order Algorithm

In this section, we show that solving Problem (4) with the first-order method given in Algorithm 4.1 is faster than solving the corresponding Second-Order Conic Program following the formulation given in [27] and Semidefinite Programming formulation of the problem using CVX calling the SDPT3 solver using default parameters. The CPU times required by the three methods are given in Table 1 when regressors generated from the second-order response surface model in 2-7 factors for $p = 0$ (labelled as D-optimal) and $p = -1$ (labeled as A-optimal). The lower and upper bounds were generated randomly in $(1/4m, 0.5)$ and $(0.5, 1)$, respectively, and only feasible instances were used in the experiments. We set $\epsilon = 10^{-6}$ for the first-order algorithm to obtain solutions with at least the same level of accuracy with the SDP solver. Note that the number of regression points to be considered in such a factorial design (denoted by m as usual) increases exponentially as the number of factors f increase, therefore we are able to solve only upto $f = 7$ with the SDP solver. On the other hand, the first-order algorithm can tackle larger number of factors. Similarly, average CPU times for the two algorithms for SFrandom sets are given in Table 2, where each row corresponds to the average of 10 random instances of the given size. Both SOCP and SDP formulations are only available for some of the criteria and are slower than the first-order method developed here, nevertheless they have the advantage of being able to include any type of linear constraints in the model formulation. If the constraints are only lower and upper bounds on the design weights such as the subproblems in the branch and bound procedure discussed in this manuscript, then the first-order method is applicable and has an obvious advantage.

			D-optimal			A-optimal		
f	n	m	First-Order	SOCP	SDP	First-Order	SOCP	SDP
2	6	9	0.006	0.3	1.4	0.005	0.2	1.3
3	10	27	0.003	0.5	1.7	0.01	0.3	1.4
4	15	81	0.018	0.7	2.1	0.03	0.8	2.9
5	21	253	0.2	1.3	4.5	0.2	3.4	16.8
6	28	729	0.5	3.1	13.6	0.9	63.4	256.8
7	36	2187	1.9	16.2	90.1	3.3	98.2	≥ 3600

Table 1: CPU times (in seconds)to solve Problem (4) with Algorithm 4.1 versus SDPT3 for Response Surface Models with f factors and 3 factorial design.

		D-optimal			A-optimal		
n	m	First-Order	SOCP	SDP	First-Order	SOCP	SDP
5	100	0.005	0.5	1.9	0.02	0.4	1.9
5	150	0.02	0.4	1.6	0.01	0.4	1.8
5	200	0.02	0.6	1.8	0.01	0.3	1.7
10	100	0.006	0.5	1.4	0.002	0.6	2.0
10	150	0.01	0.7	1.8	0.007	0.6	2.8
10	200	0.01	0.7	1.8	0.02	0.9	3.4
15	100	0.005	0.9	2.1	0.02	0.9	3.9
15	150	0.01	0.9	2.3	0.01	1.2	5.6
15	200	0.02	1.2	2.9	0.03	1.9	8.2
20	100	0.01	2.9	3.5	0.01	3.1	11.5
20	150	0.03	3.5	4.7	0.03	4.3	15.9
20	200	0.0172	4.8	5.1	0.05	7.6	23.8

Table 2: CPU times (in seconds) to solve Problem (4) with Algorithm 4.1 versus SDPT3 for SFrandom data sets.

6.3 Performance of the B&B Algorithm

In this section, we compare the performance of the B&B algorithm with the KL-exchange algorithm of [4]. There are multiple versions of the exchange algorithm, we provide the details of the version we have coded in Algorithm 6.1. We denote $\max^L$ and $\min^K$ as the set functions that return the L largest and K smallest values in a list of numbers, respectively. We set $L = m/4$, $K = n$, $num\_trials = 5000$ and $num\_exchanges = 200$. This means that the heuristic starts at 5000 different solutions and for each start it performs at most 200 exchanges as long as exchanging the weights of two points improves the objective function. We have chosen the "greedy" implementation that choses the two points which maximize the objective function value among the candidate pairs which are the L design points that have the maximum variance and K design points that have the lowest variance and non-zero weights. The choices of the parameters and which exchanges are chosen affect the run time of the heuristic significantly as expected. However, the conclusions we draw here are mostly unaffected by these changes.

Algorithm 6.1

```
Require: $X, N, K, L, num\_trials, num\_exchanges$
  $g_{KL} = -\infty$
  for trials=1:num_trails do
    Generate $u$ randomly. (Uniformly sample $N$ integers with replacement from the set $\{1, \ldots, m\}$ and assign $1/N$ weight to each of these points.)
    $exchanges = 0$, $opt2ex = 0$
    while $exchanges < num\_exchanges$ and $opt2ex = 0$ do
      Calculate $\omega(u)$ and $g(u)$.
      Find $S_L := \arg\max^L\{\omega_i\}$ and $S_K := \arg\min^K\{\omega_i : u_i > 0\}$.
      Find $\{l, k\} := \arg\max_{k \in S_K, l \in S_L}\{|(1 - w_k))(1 + w_l + w_{kl})|\}$.
      Calculate $u_+$ as in (9), where $\theta = 1/N$.
      Update $\omega(u_+)$ and $g(u_+)$.
      if $g(u_+) < g(u)$ then
        Set $opt2ex = 1$
      else
        $u = u_+$, $g(u) = g(u_+)$, and $exchanges++$
      end if
    end while
    if $g(u) > g_{KL}$ then
      $u_{KL} = u$, $g_{KL} = g(u)$
    end if
  end for
  return $u_{KL}, g_{KL}$
```

We have generated 100 SFrandom datasets for several dimensions (combinations of (n, m, N)) and report the min, mean, and max of CPU times required to solve these instances in Table 3. By construction, the B&B method always finds the optimal solution, while the KL-exchange algorithm doesn't always find the optimal solution. The penultimate column of the table stores the number instances among 100 that were not solved to optimality by the heuristic; while the last column shows the minimum D-efficiency of the designs obtained by the heuristic. We have repeated the same experiment for the response surface models with $f = 2$ and $f = 3$ for various values of N. In all such experiments the KL heuristic has found the optimal solution, but in slightly more time than the branch and bound method. The computational results show that the KL heuristic is very reliable, nevertheless when the size permits, the branch and bound algorithm is capable of finding the exact solution in less amount of time.

			B&B			KL-exchange				
n	m	N	min	mean	max	min	mean	max	min D-eff	#solved
3	25	8	0.0007	0.005	0.05	0.6	0.7	0.9	0.96	93
3	50	10	0.0001	0.005	0.01	0.8	0.9	2.5	0.97	87
3	75	12	0.0002	0.004	0.01	1.1	1.2	1.4	0.96	93
3	100	15	0.0002	0.005	0.02	1.4	1.7	2.1	0.98	91
5	25	8	0.007	0.01	0.03	0.7	0.8	0.9	0.99	98
5	50	10	0.0009	0.02	0.05	1.07	1.2	1.4	0.98	96
5	75	12	0.008	0.02	0.1	1.3	1.4	1.5	0.97	94
5	100	15	0.006	0.03	0.2	1.5	1.7	2.7	0.96	91
10	25	15	0.02	0.08	0.3	1.1	1.2	1.4	0.97	89
10	50	20	0.04	0.4	2.6	1.5	5.9	4.4	0.97	85

Table 3: CPU times (in seconds) to solve Problem (3) with the Branch-and-Bound algorithm versus the KL-exchange algorithm for SFrandom data sets.

7 Conclusions

We have provided a branch-and-bound algorithm for the exact Optimal Experimental Design problem which can be implemented for any of the Kiefer's optimality criteria. The algorithm is fast when the problem size is moderate and should be used instead of the inexact methods that (with the exception of [26]) don't have optimality guarantees.

References

[1] S. D. Ahipaşaoğlu and P. Sun and M. J. Todd, Linear convergence of a Modified Frank-Wolfe algorithm for computing minimum-volume enclosing ellipsoids, Optimization Methods and Software, 23, 5–19 (2008)

[2] S. D. Ahipaşaoğlu, A First-Order Algorithm for the A-Optimal Experimental Design Problem: A Mathematical Programming Approach, Statistics and Computing, 25, 1113–1127 (2015)

[3] Z. Allen-Zhu and Y. Li and A. Singh and Y. Wang, Near-optimal design of experiments via regret minimization. Proceedings of Machine Learning Research, 126–135 (2017)

[4] A. C. Atkinson and A. N. Donev and R. D. Tobias, Optimum Experimental Designs, with SAS, Oxford University Press, Oxford (1992)

[5] C. L. Atwood, Sequences converging to D-optimal designs of experiments, The Annals of Statistics, 1(2), 342–352 (1973)

[6] H. Avron and C. Boutsidis, Faster subset selection for matrices and applications. SIAM Journal on Matrix Analysis and Applications, 34(4), 1464–1499 (2013)

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

[8] M. Bouhtou and S. Gaubert and Guillaume Sagnol, Submodularity and randomized rounding techniques for optimal experimental design. Electronic Notes in Discrete Mathematics, 36, 67–686 (2010)

[9] R. D. Cook and L. A. Thibodeau, Marginally restricted D-optimal designs. Journal of American Statistical Association, 75(370), 366–371 (1980)

[10] R. D. Cook and V. V. Fedorov, Constrained Optimization of Experimental Design. Statistics 26, 129–178 (1995)

[11] R. D. Cook and C. J. Nachtsheim, A comparison of algorithms for constructing exact D-optimal designs. Technometrics, 22, 315–324 (1995)

[12] V. V. Fedorov, Theory of Optimal Experiments. Academic Press, New York (1972)

[13] V. V. Fedorov and S. L. Leonov, Optimal design for nonlinear response models. CRC Press (2013)

[14] L. Filova R. Harman, Ascent with quadratic assistance for the construction of exact experimental designs. Computational Statistics, 35, 775–801 (2020)

[15] R. García-Ródenas and J. C. García-García, J. López-Fidalgo, J. A. Martín-Baos, W. K. Wong, A comparison of general-purpose optimization algorithms for finding optimal approximate experimental designs, Computational Statistics & Data Analysis, 144, Article 106844 (2020)

[16] R. Harman and L. Filova, Computing efficient exact designs of experiments using integer quadratic programming, Computational Statistics & Data Analysis, 71, 1159–1167 (2014)

[17] R. Harman and L. Filova and P. Richtarik, A Randomized Exchange Algorithm for Computing Optimal Approximate Designs of Experiments, Journal of the American Statistical Association, 115(529), 348–361 (2020)

[18] R. Harman and G Sagnol, Computing D-optimal experimental designs for estimating treatment contrasts under the presence of a nuisance time trend. In: Steland A., Rafajlowicz E., Szajowski K. (eds.) Stochastic Models, Statistics and Their Applications, Springer Proceedings in Mathematics & Statistics, vol. 122, pp. 83–91 (2015)

[19] J. Kiefer, General Equivalence Theory for Optimum Designs (Approximate Theory), Annals of Statistics, 2 (5), 849–879 (1974)

[20] P. Kumar and E. A. Yıldırım, Minimum volume enclosing ellipsoids and core sets, Journal of Optimization Theory and Applications, 126(1), 1–21 (2005)

[21] V. Madan and M. Singh and U. Tantipongpipat and W. Xie, Combinatorial Algorithms for Optimal Design, Proceedings of Machine Learning Research, 99, 2210–2258 (2019)

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

[23] T. J. Mitchell, An algorithm for the construction of D-optimal designs. Technometrics, 20, 203–210 (1974)

[24] A. Nikolov and M. Singh and U. Tantipongpipat, Proportional volume sampling and approximation algorithms for A-optimal design. SODA'19: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, 1369–1386. SIAM, 2019.

[25] F. Pukelsheim, Optimal Design of Experiments. John Wiley and Sons, New York (1993)

[26] F. Pukelsheim and S. Rieder, Efficient rounding of approximate designs. Biometrika, 79, 763–770 (1992)

[27] G. Sagnol, R. Harman R, Computing exact D-optimal designs by mixed integer second-order cone programming, The Annals of Statistics, 43, 2198–2224 (2015)

[28] S.D. Silvey and D.M. Titterington, A geometric approach to optimum design theory. Biometrika, 60, 21–32 (1973)

[29] M. Singh and W. Xie, Approximate positive correlated distributions and approximation algorithms for D-optimal design. SODA'18: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 2240–2255. SIAM, 2018.

[30] P. Sun and R. M. Freund, Computation of Minimum Volume Covering Ellipsoids, Operations Research, 52, 690–706 (2004)

[31] M. J. Todd and E. A. Yıldırım, On Khachiyan's algorithm for the computation of minimum volume enclosing ellipsoids, Discrete and Applied Mathematics, 155, 1731–1744 (2007)

[32] B. Torsney and R. R. Martin-Martin, Multiplicative algorithms for computing optimum designs, Journal of Statistical Planning and Inference, 139, 3947–3961 (2009)

[33] D. Ucinski, Sensor network scheduling for identi

cation of spatially distributed processes. International Journal of Applied Mathematics and Computer Science, 22(1), 25–40 (2012)

[34] D. Ucinski, An algorithm for construction of constrained D-optimum designs. In Stochastic Models, Statistics and Their Applications (pp. 461-468). Springer, Cham (2015)

[35] D. Ucinski and M. Patan. D-optimal design of a monitoring network for parameter estimation of distributed systems, Journal of Global Optimization, 39, 291–322 (2007)

[36] Y. Wang and A. W. Yu and A. Singh. On computationally tractable selection of experiments in regression models. Journal of Machine Learning Research, 18(143), 1–41 (2017)

[37] W. J. Welch, Algorithmic Complexity: Three NP-hard problems in computational statistics, Journal of Statistical Computation and Simulation, 15, 17–25 (1982)

[38] W. J. Welch, Branch-and-Bound search for experimental designs based on D optimality and other criteria, Technometrics, 24, 41–48 (1982)

[39] H. P. Wynn, Results in the theory and construction of D-optimum experimental designs, Journal of the Royal Statistical Society, Series B (Methodological), 34,133–147 (1972)

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

[41] Y. Yu, D-optimal designs via a cocktail algorithm, Statistics and Computing, vol. 21, pp. 475-481 (2011)