A Randomized Exchange Algorithm for Computing Optimal Approximate Designs of Experiments

A Randomized Exchange Algorithm for Computing Optimal Approximate Designs of Experiments

Radoslav Harman[1,2], Lenka Filová[1], and Peter Richtárik[3,4,5]

[1]Comenius University in Bratislava, Slovakia
[2]Johannes Kepler University Linz, Austria
[3]King Abdullah University of Science and Technology, Kingdom of Saudi Arabia
[4]University of Edinburgh, United Kingdom
[5]Moscow Institute of Physics and Technology, Russia

January 18, 2018

Abstract

We propose a class of subspace ascent methods for computing optimal approximate designs that covers both existing as well as new and more efficient algorithms. Within this class of methods, we construct a simple, randomized exchange algorithm (REX). Numerical comparisons suggest that the performance of REX is comparable or superior to the performance of state-of-the-art methods across a broad range of problem structures and sizes. We focus on the most commonly used criterion of D-optimality that also has applications beyond experimental design, such as the construction of the minimum volume ellipsoid containing a given set of data-points. For D-optimality, we prove that the proposed algorithm converges to the optimum. We also provide formulas for the optimal exchange of weights in the case of the criterion of A-optimality. These formulas enable one to use REX for computing A-optimal and I-optimal designs.

Keywords: optimal approximate design of experiments, D-optimality, A-optimality, I-optimality, convex optimization, minimum volume ellipsoid

1 Introduction

The topic of this paper is the computation of optimal approximate designs for regression models with uncorrelated errors (e.g., [11], [28], [31], [3]). As a special case, we also briefly discuss the minimum volume enclosing ellipsoid problem (e.g., [44]).

Suppose that we intend to perform an experiment consisting of a set of trials. Assume that the observed response of each trial depends on a design point x chosen from a finite set $\mathfrak{X}$. For simplicity but without the loss of generality, we will assume that $\mathfrak{X} = \{1, \ldots, n\}$. Note that in the experimental design problems, n can be a large number, often many thousands.

For $x \in \mathfrak{X}$, the real-valued observation $Y(x)$ is assumed to satisfy the linear[1] regression model $Y(x) = \mathbf{f}'(x)\beta + \varepsilon(x)$, where $\mathbf{f}(x) \in \mathbb{R}^m$ is the known regressor associated with x, the vector $\beta \in \mathbb{R}^m$ contains the unknown parameters of the model, and $\varepsilon(x) \sim N(0, \sigma^2)$, with $\sigma^2 > 0$, is an unknown variance.[2] For different trials, the errors are assumed to be uncorrelated. Let the model be non-singular in the sense that $\{\mathbf{f}(x) : x \in \mathfrak{X}\}$ spans $\mathbb{R}^m$. We also avoid redundant regressors by assuming that $\mathbf{f}(x) \neq \mathbf{0}_m$ for all $x \in \mathfrak{X}$. Note that in the applications, the number m of the parameters tends to be relatively small, mostly less than 10.

We formalize an (approximate experimental) design on $\mathfrak{X}$ as an n-dimensional vector $\mathbf{w}$ with non-negative real components summing to one.[3] From the point of view of an experimenter, the component w_x of $\mathbf{w}$ represents the proportion of the trials to be performed in the design point $x \in \mathfrak{X}$. The support of a design $\mathbf{w}$ is $\mathrm{supp}(\mathbf{w}) = \{x \in \mathfrak{X} : w_x > 0\}$.[4] The set of all designs on $\mathfrak{X}$ denoted by Ξ is the probability simplex in $\mathbb{R}^n$, which is compact and convex.

The information matrix associated with a design $\mathbf{w}$ is defined by

$$\mathbf{M}(\mathbf{w}) := \sum_{x \in \mathfrak{X}} w_x \mathbf{f}(x)\mathbf{f}'(x).$$

Under our model's assumptions, the information matrix is proportional to the Fisher information matrix corresponding to β. Therefore, the general aim is to choose $\mathbf{w}$ such that $\mathbf{M}(\mathbf{w})$ is "as large as possible", which we make precise next.

Let $\mathcal{S}^m_+$ be the set of $m \times m$ symmetric non-negative definite matrices, and let $\Phi : \mathcal{S}^m_+ \to \mathbb{R} \cup \{-\infty\}$ be a criterion of optimality, that is, a function measuring the "size" of information matrices. A design $\mathbf{w}^*$ is said to be a Φ-optimal design if it maximizes $\Phi(\mathbf{M}(\mathbf{w}))$ in the class Ξ of all designs:

$$\mathbf{w}^* \in \mathrm{argmax}\{\Phi(\mathbf{M}(\mathbf{w})) : \mathbf{w} \in \Xi\}. \tag{1}$$

Matrix $\mathbf{M}(\mathbf{w}^*)$ is referred to as the Φ-optimal information matrix.

Due to their natural statistical interpretations, the two most common optimality criteria are D-optimality and A-optimality. In this paper, we use them in the forms (e.g., [31]) $\Phi_D(\mathbf{M}) = (\det(\mathbf{M}))^{1/m}$ and $\Phi_A(\mathbf{M}) = (\mathrm{tr}(\mathbf{M}^{-1}))^{-1}$, respectively, for any positive definite matrix $\mathbf{M}$. For a singular non-negative definite matrix $\mathbf{M}$, the values of the criteria are defined to be 0. Functions Φ_D and Φ_A are positively homogeneous, continuous, and concave on the set of all non-negative definite matrices. Hence, for both D- and A-optimality, the optimal design always exists, and the optimal information matrix is non-singular.

[1]It is also straightforward to apply the algorithms proposed in this paper to the computation of locally-optimal designs for *non-linear* regression models. In this sense, we gain clarity and do not lose generality if we formulate our method for the linear regression (see [3], chapter 17).

[2]We assume homoskedasticity and the normality of errors for the sake of simplicity. Our algorithms can be applied to all models with uncorrelated $\epsilon(x)$, $x \in \mathfrak{X}$, with finite variances, after a simple transformation of the problem ([3], chapter 23).

[3]Often, an approximate experimental design is formalized as a probability measure on $\mathfrak{X}$, but for a finite design space, the representations by a probability measure and by a vector of probabilities are equivalent.

[4]Note that there are theoretical reasons corroborated by extensive numerical evidence that the "optimal" designs w tend to be sparse in the sense of having support much smaller than n. More precisely, there always exists an optimal design supported on a set of a size that does not exceed $1 + m(m+1)/2$.

Let Ξ_R denote the set of regular designs, i.e.,

$$\Xi_R = \{\mathbf{w} \in \Xi : \mathbf{M}(\mathbf{w}) \text{ is non-singular}\}.$$

For this paragraph, let $\mathbf{w} \in \Xi_R$ be fixed. The variance function of $\mathbf{w}$ is the n-dimensional vector $\mathbf{d}(\mathbf{w})$ with components

$$d_x(\mathbf{w}) = \mathbf{f}'(x)\mathbf{M}^{-1}(\mathbf{w})\mathbf{f}(x),\ x \in \mathfrak{X}.$$

Therefore, $\mathbf{d}(\cdot)$ is a function that maps regular designs to $\mathbb{R}^n$. From the statistical point of view, $d_x(\mathbf{w})$ is proportional to the variance of the best linear unbiased estimator (BLUE) of $\mathbf{f}'(x)\beta$, provided that the approximate design $\mathbf{w}$ can be actually carried out (hence the term). We also note that

$$d_x(\mathbf{w}) = \lim_{\alpha \to 0_+} \frac{\Psi_D[(1-\alpha)\mathbf{M}(\mathbf{w}) + \alpha\mathbf{M}(\mathbf{e}_x)] - \Psi_D(\mathbf{M}(\mathbf{w}))}{\alpha} + m, \tag{2}$$

where $\Psi_D(\mathbf{M}) = \log\det(\mathbf{M})$ is a version of the D-optimality criterion and $\mathbf{e}_x$ is the singular design in x (formally the x-th standardized unit vector, i.e., the vertex of Ξ). This explains why $\mathbf{d}$ is often used in iterative algorithms for computing the D-optimal designs to select "the most promising direction" of change of the current design.

For A-optimality, because the directional derivative of $\Psi_A(\mathbf{M}) = -\mathrm{tr}(\mathbf{M}^{-1})$ in $\mathbf{M}(\mathbf{w})$ in the direction of $\mathbf{M}(\mathbf{e}_x)$ is $\mathbf{f}'(x)\mathbf{M}^{-2}(\mathbf{w})\mathbf{f}(x) - \mathrm{tr}(\mathbf{M}^{-1}(\mathbf{w}))$, we can define an n-dimensional vector $\mathbf{a}(\mathbf{w})$ with components

$$a_x(\mathbf{w}) = \mathbf{f}'(x)\mathbf{M}^{-2}(\mathbf{w})\mathbf{f}(x),\ x \in \mathfrak{X}$$

as a quantity analogous to the variance function in the case of D-optimality.

In the following, let $\mathbf{g}(\mathbf{w})$ be either $\mathbf{d}(\mathbf{w})$ or $\mathbf{a}(\mathbf{w})$, depending on the criterion under consideration. Besides indicating a degree of importance of design points, another reason for utilizing the vector $\mathbf{g}(\mathbf{w})$ is that $\max_x g_x(\mathbf{w})$ can be used to compute a natural stopping rule for algorithms based on the notion of statistical efficiency, as follows.

The D- and A-efficiencies of a design $\mathbf{w}$ relative to $\bar{\mathbf{w}} \in \Xi_R$ are (see [31])

$$\mathrm{eff}_D(\mathbf{w} \mid \bar{\mathbf{w}}) := \frac{\Phi_D(\mathbf{M}(\mathbf{w}))}{\Phi_D(\mathbf{M}(\bar{\mathbf{w}}))}, \qquad \mathrm{eff}_A(\mathbf{w} \mid \bar{\mathbf{w}}) := \frac{\Phi_A(\mathbf{M}(\mathbf{w}))}{\Phi_A(\mathbf{M}(\bar{\mathbf{w}}))}.$$

Because Φ_D and Φ_A are positively homogeneous, the notion of efficiency as defined above can be interpreted in terms of the relative numbers of trials needed to achieve a given criterion value. For instance, $\mathrm{eff}_D(\mathbf{w}|\bar{\mathbf{w}}) = 0.99$ means that, asymptotically, to achieve the same amount of information measured by the criterion of D-optimality, we only need 99% of the trials using the design $\bar{\mathbf{w}}$ compared to number of trials needed if we use the design $\mathbf{w}$. Let $\mathbf{w} \in \Xi_R$. If $\mathbf{w}^D$ is a D-optimal design and $\mathbf{w}^A$ is an A-optimal design, then (see [31], Ch.6)

$$\mathrm{eff}_D(\mathbf{w} \mid \mathbf{w}^D) \geq \frac{m}{\max_{x\in\mathfrak{X}} d_x(\mathbf{w})}, \qquad \mathrm{eff}_A(\mathbf{w} \mid \mathbf{w}^A) \geq \frac{\mathrm{tr}(\mathbf{M}^{-1}(\mathbf{w}))}{\max_{x\in\mathfrak{X}} a_x(\mathbf{w})}. \tag{3}$$

If $\mathbf{w}$ is the current design and its efficiency compared to the optimal design is almost equal to 1, there is no practical reason to continue the computation.

1.1 Methods for computing optimal designs of experiments

In this paper, we are concerned with the numerical computation of (approximate) Φ-optimal designs as given in (1), with special focus on D- and A-optimality. Within the field of optimal experimental design, the first contributions to solving this problem were made by V. V. Fedorov and H. Wynn (e.g., [11] and [50]), who developed the so-called vertex direction methods (VDMs) that are closely related to the Frank-Wolfe algorithm. In each iteration, VDMs move the current design $\mathbf{w}$ in the direction of $\mathbf{e}_x$ for some design point x while decreasing all remaining components of $\mathbf{w}$ by the same proportion. Here, x can be chosen to maximize $g_x(\mathbf{w})$. Although these methods converge, some of them monotonically, they tend to be inconveniently slow. More efficient variants (e.g., [4]) also allow the decrease of a single component of the current design. A related alternative approach called the vertex exchange method for D-optimality (VEM) was proposed by Bohning [6].

The VEM algorithm (see Algorithm 1 for a pseudo-code) is one of the simplest special cases of the class of algorithms presented in this paper. In each iteration, with a current design $\mathbf{w}$, VEM selects a pair of points $k, l \in \mathfrak{X}$ defined by

$$k \in \operatorname{argmin}\{d_u(\mathbf{w}) : u \in \operatorname{supp}(\mathbf{w})\},\ l \in \operatorname{argmax}\{d_v(\mathbf{w}) : v \in \mathfrak{X}\}.$$

We call such a pair (k, l) a Bohning's pair of design points. Then, the VEM computes $\alpha^* \in [-w_l, w_k]$ such that Φ_D in $\mathbf{M}(\mathbf{w} + \alpha\mathbf{e}_l - \alpha\mathbf{e}_k)$ is maximized. Such an α^*, which we call the Bohning's step, can be analytically computed. After each exchange, the variance function $\mathbf{d}$ is recomputed.

```
input  : Regressors $\mathbf{f}(1), \dots, \mathbf{f}(n)$ of the model, required efficiency $eff$, maximum time $t.max$
output: Approximate design $\mathbf{w}$
1 Generate a regular $m$-point design $\mathbf{w}$;
2 while $eff.act(\mathbf{w}) < eff$ and $time < t.max$ do
3     Let $k$ belong to $\operatorname{argmin}\{d_u(\mathbf{w}) : u \in \operatorname{supp}(\mathbf{w})\}$
4     Let $l$ belong to $\operatorname{argmax}\{d_v(\mathbf{w}) : v \in \mathfrak{X}\}$
5     Let $\alpha^*$ belong to $\operatorname{argmax}\{\Phi_D(\mathbf{M}(\mathbf{w} + \alpha\mathbf{e}_l - \alpha\mathbf{e}_k)) : \alpha \in [-w_l, w_k]\}$
6     $w_k \leftarrow w_k - \alpha^*$; $w_l \leftarrow w_l + \alpha^*$
7 end
```

Algorithm 1: A variant of the Bohning's vertex exchange method (VEM). The value of $eff.act(\mathbf{w})$ is calculated as a lower bound on the efficiency of the current design $\mathbf{w}$ based on a standard formula (see (3)). The variable *time* measures the time elapsed from the start of the computation.

A substantially different strategy for solving the problem (1), involving simultaneously updating all components of $\mathbf{w}$, is implemented in the multiplicative algorithm (MUL). The MUL and its variants can be attributed to [40]; extensions can be found in [10] and [17].

The three methods of VDM, VEM and MUL were more recently combined by Yu ([48]) in the "cocktail" algorithm for D-optimality. This approach often increases the speed while preserving convergence.

Another efficient recent method is presented in [46]. It is the simplicial decomposition algorithm where the master non-linear problem is solved by a second-order Newton method and can be used for various twice-differentiable concave criteria, including both D- and A-optimality. See also [45] for an earlier application of simplicial decomposition in the optimal design of experiments and further references.

The problem of searching for an optimal design can be also solved by specific mathematical programming methods. Namely, in [49], it is shown that the problem of A-optimality can be cast as semidefinite programming, while D-optimality can be reduced to maxdet programming. More recently, the work [38] enables one to solve both A- and D-optimal design problems (for the approximate design case without additional constraints[5]) by the second-order cone programming approach.

The optimal design problem is closely related to the minimum volume enclosing ellipsoid (MVEE) problem. Considerable attention has been paid to developing fast algorithms for the MVEE problem in mathematical programming and optimization literature. For the latest developments, see [43], [2], and [44].

Consider a set $\{\mathbf{f}_1, \dots, \mathbf{f}_n\}$ spanning $\mathbb{R}^m$. Then, the task of finding MVEE $\mathcal{E}(\mathbf{M}) := \{\mathbf{f} \in \mathbb{R}^m : \mathbf{f}'\mathbf{M}\mathbf{f} \leq 1\}$, $\mathbf{M} \in \mathcal{S}^m_+$, containing $\mathbf{f}_1, \dots, \mathbf{f}_n$ can be cast as

$$\begin{array}{rl} \min_{\mathbf{M}\in\mathcal{S}^m_{++}} & -\log\det\mathbf{M} \\ \text{subject to} & \mathbf{f}_i'\mathbf{M}\mathbf{f}_i \leq 1, \ i = 1, \dots, n \end{array}$$

with the dual problem

$$\begin{array}{rl} \max_{\mathbf{w}\geq 0} & \log\det\sum_{i=1}^n w_i \mathbf{f}_i\mathbf{f}_i' \\ \text{subject to} & \sum_{i=1}^n w_i = 1. \end{array}$$

Thus, the problem of finding the D-optimal design is equivalent to the MVEE problem (see, e.g., [44]), making the task of developing a fast and efficient algorithm for computing D-optimal designs relevant for a wider optimization community.

With D-optimality being the most popular criterion, the literature focusing on the computational issues of A-optimality is scarcer, although in some areas, the use of A-optimality is very natural. In particular, the computation of the important I-optimal designs[6] (e.g., [8], [13]) can be converted into A-optimality (cf. [3], Section 10.6). That is, I-optimal designs on a finite design space can also be computed using the algorithm developed in this paper. In addition to the mathematical programming methods mentioned above, a generalization of the vertex direction method that focuses on A-optimality, together with an analytical formula for the optimal VDM step-length, is presented in [1].

1.2 Summary of contributions

In Section 2, we introduce a general class of methods for finding optimal approximate designs that we call the subspace ascent method (SAM) and describe some known algorithms as special cases.

[5]For an improvement that also enables solving exact design problems and approximate design problems with non-standard constraints using the second-order cone programming, see [39].

[6]These are occasionally called IV- or V-optimal designs.

In Section 3, we present our key method: REX (randomized exchange method). REX is a simple batch-randomized exchange algorithm that is a member of the SAM family and whose performance is superior to the state-of-the-art algorithms in nearly all problems we tested. The proposed algorithm can be viewed as an efficient extension or combination of both the VEM algorithm and the KL exchange algorithm that is used to compute exact designs (see [3]). In the same section, we compare our method to known subspace ascent/descent methods that were recently developed in the optimization and machine learning communities by highlighting the similarities and differences. The numerical comparison of the state-of-the art algorithms with this newly proposed method is the subject of Section 4.

In the Appendix, we provide a proof of convergence of the proposed algorithm in the case of D-optimality. Additionally, in the existing literature, many of the mentioned methods have only been tailored to D-optimality, and although the transition to A-optimality is possible, it is not trivial. Therefore, in the Appendix, we also present formulas for the optimum exchange of weights for A-optimality, which, to the best of our knowledge, were previously only derived for the D-optimality criterion.

2 Subspace ascent method

In Section 2.1, we propose a generic algorithmic paradigm for solving (1). It helps illustrate several existing approaches and the newly proposed method from a broader perspective. We refer to this paradigm as the "Subspace Ascent Method" (SAM), formalized as Algorithm 2. Subsequently, in Section 3.2, we briefly comment on the context of the "subspace ascent" idea within existing optimization, linear algebra and machine learning literature.

2.1 SAM

SAM (Algorithm 2) is an iterative procedure for finding an approximate Φ-optimal design. We start with an arbitrary regular design $\mathbf{w}^0 \in \Xi$. In iteration k, a subset S_k of the design points is chosen via a certain rule (e.g., a small random subset of all design points). In this iteration, we only allow for weights w_x for $x \in S_k$ to change. All other weights are frozen at their last values. That is, we are deliberately reducing our search for $\mathbf{w}^{k+1}$ to a subset of the set of all designs Ξ at the intersection of Ξ and a particular affine subspace of $\mathbb{R}^n$:

$$\Xi_k := \Xi \cap \mathcal{L}_k, \qquad \mathcal{L}_k := \{\mathbf{w} \ : \ w_x = w_x^k \text{ for all } x \notin S_k\},$$

Note that $\mathbf{w}^k \in \Xi_k$ for all k and, therefore, $\max_{\mathbf{w}\in\Xi_k} \Phi(\mathbf{M}(\mathbf{w})) \geq \Phi(\mathbf{M}(\mathbf{w}^k))$. In particular, there exists $\mathbf{w}^{k+1} \in \Xi_k$ such that the ascent condition $\Phi(\mathbf{M}(\mathbf{w}^{k+1})) \geq \Phi(\mathbf{M}(\mathbf{w}^k))$ is satisfied.

If $|S_k| = 1$, then Ξ_k is a singleton and the method cannot progress. As soon as $|S_k| > 1$, this problem is removed, and Ξ_k has a chance to contain points better than $\mathbf{w}^k$.

SAM is a generic method in the sense that it does not specify how the set S_k is chosen or how the approximate solution $\mathbf{w}^{k+1}$ is obtained. Formally, any optimum design algorithm for a discrete design space (including VDMs and the MUL) is a special case of SAM if we allow the choice $S_k = \mathfrak{X}$. However, the intended philosophy of the SAM approach is to make the sets S_k much smaller than $\mathfrak{X}$. This makes the partial optimization problem small-dimensional and potentially simple to solve, at least approximately.

```
  input  : Initial regular design $\mathbf{w}^0 \in \Xi$
  output: Approximate design $\mathbf{w}^k$
1 Set $k \leftarrow 0$
2 while $\mathbf{w}^k$ does not satisfy a stopping condition do
3     Select a subset $S_k$ of the set of design points $\mathfrak{X} = \{1, 2, \dots, n\}$
4     Define the active subspace of $\Xi$ as $\Xi_k := \{\mathbf{w} \in \Xi : w_x = w_x^k$ for all $x \notin S_k\}$
5     Compute $\mathbf{w}^{k+1}$ as an approximate solution of $\max_{\mathbf{w} \in \Xi_k} \Phi(\mathbf{M}(\mathbf{w}))$ satisfying the ascent
        condition $\Phi(\mathbf{M}(\mathbf{w}^{k+1})) \geq \Phi(\mathbf{M}(\mathbf{w}^k))$
6     Set $k \leftarrow k + 1$
7 end
```

Algorithm 2: Subspace Ascent Method (SAM)

The algorithm VEM chooses S_k to be a two point set, which allows the partial optimization problem to be analytically solved for D-optimality (and, as we show in the Appendix, for A-optimality). However, this method requires overly frequent updates of the set S_k, which is a computationally non-trivial operation for large n. Therefore, the speed of VEM significantly declines with the increase of the design space.

Similarly, SAM also includes the algorithm YBT ([46]), which chooses S_k to be the support of the current design enriched by one additional support point x. Then, a partial optimization problem is solved by a specific constrained Newton method. However, this approach is generally inefficient for larger values of m because then the sets S_k tend to be large and the efficiency of the Newton method deteriorates rapidly as the dimension of the problem increases (as demonstrated in Section 4).

3 Randomized exchange algorithm

In this section, we propose a particular case of SAM for which we coin the name *randomized exchange algorithm (REX)*. Then, we briefly comment on the connection of this method to the existing literature on mathematical optimization and machine learning.

3.1 REX

Let $\mathbf{w}$ be a regular design and $\mathbf{g}(\mathbf{w})$ be an n-dimensional vector with components $g_x(\mathbf{w})$, as defined in the introduction. Recall that the value $g_x(\mathbf{w})$ is a $\mathbf{w}$-based estimate of the plausibility that x is the support point of an optimal design.

The general idea behind the proposed batch-randomized algorithm is to repeatedly select a batch of several pairs of design points (the number of pairs is not fixed and may vary from iteration to iteration) and randomly perform optimal exchanges of weights between the pairs of design points. The selection of the batch depends on the vector $\mathbf{g}(\mathbf{w})$ evaluated in the best available design $\mathbf{w}$. Its size can be fine-tuned by a tuning parameter $\gamma \geq 1/m$. We will call the resulting algorithm the randomized exchange algorithm (REX). We now proceed to a description of the REX algorithm (See Algorithm 3 for a concise pseudo-code).

1. **LBE step.** Given the current design $\mathbf{w}$, compute $\mathbf{g}(\mathbf{w})$ and subsequently perform the "leading Bohning exchange" (LBE) as follows:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha^*_{k,l}(\mathbf{w})(\mathbf{e}_l - \mathbf{e}_k), \tag{4}$$

   where

$$\begin{aligned} k &\in \operatorname{argmin}\{g_u(\mathbf{w}) : u \in \operatorname{supp}(\mathbf{w})\}, \\ l &\in \operatorname{argmax}\{g_v(\mathbf{w}) : v \in \mathfrak{X}\}, \\ \alpha^*_{k,l}(\mathbf{w}) &\in \operatorname{argmax}\{\Phi(\mathbf{M}(\mathbf{w} + \alpha(\mathbf{e}_l - \mathbf{e}_k))) : \alpha \in [-w_l, w_k]\} \end{aligned}$$

   An optimal step $\alpha^*_{k,l}(\mathbf{w})$ is called "nullifying" if it is equal to either $-w_l$ or w_k.

2. **Subspace selection.** Subspace S of $\mathfrak{X}$ in which the method will operate arises as the union of two sets. One is formed by a greedy process (S_{greedy}) informed by the elements of the vector $\mathbf{g}(\mathbf{w})$, and the other is identical to the support of the current design $\mathbf{w}$ (S_{support}).

   (a) **Greedy.** Set $L = \min(\gamma m, n)$, and choose points

$$S_{\text{greedy}} = \{l^*_1, \dots, l^*_L\} \subseteq \mathfrak{X},$$

   where l^*_i corresponds to the ith largest component of the vector $\mathbf{g}(\mathbf{w})$.

   (b) **Support.** Set

$$S_{\text{support}} = \operatorname{supp}(\mathbf{w}).$$

   Let K be the size of $\operatorname{supp}(\mathbf{w})$: $K = |\operatorname{supp}(\mathbf{w})|$.

   (c) **Active subspace.** The active set of design points is defined as

$$S = S_{\text{greedy}} \cup S_{\text{support}}.$$

   The weights of no other design points are modified in this iteration. Note that if $m^2 \geq n$ or if $K = n$, then $S = \mathfrak{X} = \{1, 2, \dots, n\}$. However, a standard situation is $|S| < n$, and our method operates in a proper subspace of the full design space $\mathfrak{X}$.

3. **Subspace step.** We now perform a specific update of the design points in S. That is, we allow w_v for $v \in S$ to be updated. Elements w_v for $v \notin S$ are kept unchanged.

   (a) **Formation of pairs.** Let $(k_1, \dots, k_K)$ be a uniform random permutation of S_{support}, and let $(l_1, \dots, l_L)$ be a uniform random permutation of S_{greedy}. Form the sequence

$$(k_1, l_1), (k_2, l_1), \dots, (k_K, l_1), \dots, (k_1, l_L), (k_2, l_L), \dots, (k_K, l_L) \tag{5}$$

   of $K \times L$ pairs of active design points.

   (b) **Update.** If the LBE step was nullifying, sequentially perform all nullifying Φ-optimal exchanges between the $K \times L$ pairs in (5) with the corresponding updates of $\mathbf{w}$ and $\mathbf{M}(\mathbf{w})$. If the LBE was *not* nullifying in the current iteration, sequentially perform all Φ-optimal exchanges between the $K \times L$ pairs in (5) with the corresponding updates of $\mathbf{w}$ and $\mathbf{M}(\mathbf{w})$.

```
input  : Regressors $\mathbf{f}(1), \dots, \mathbf{f}(n) \in \mathbb{R}^m$, parameter $\gamma \geq 1/m$, threshold efficiency $\mathit{eff}$,
         maximum time $t.max$
output: Approximate design $\mathbf{w}$
1  Generate a random regular design $\mathbf{w}$
2  while $\mathit{eff}.act(\mathbf{w}) < \mathit{eff}$ and $time < t.max$ do
3      Perform the LBE in $\mathbf{w}$ as given by (4).
4      Let $\mathbf{k}$ be the vector corresponding to a random permutation of the $K$ elements of
        supp($\mathbf{w}$)
5      Let $\mathbf{l}$ be the vector corresponding to a random permutation of the indices of the
        $L = \min(\gamma m, n)$ greatest elements of $\mathbf{g}(\mathbf{w})$
6      for $l$ in $1 : L$ do
7          for $k$ in $1 : K$ do
8              Find $\alpha^*$ in argmax$\{\Phi(\mathbf{M}(\mathbf{w} + \alpha(\mathbf{e}_{\mathbf{l}_l} - \mathbf{e}_{\mathbf{k}_k}))) : \alpha \in [-w_{\mathbf{l}_l}, w_{\mathbf{k}_k}]\}$
9              if the LBE was not nullifying or $\alpha^* = -w_{\mathbf{l}_l}$ or $\alpha^* = w_{\mathbf{k}_k}$ then
10                 $w_{\mathbf{k}_k} \leftarrow w_{\mathbf{k}_k} - \alpha^*$ $w_{\mathbf{l}_l} \leftarrow w_{\mathbf{l}_l} + \alpha^*$
11             end
12         end
13     end
14 end
```

Algorithm 3: Randomized exchange algorithm (REX). The value of $\mathit{eff}.act(\mathbf{w})$ is the lower bound on the current design efficiency compared to the optimal design. For D- or A-optimality, it is possible to use formulas (3). The variable *time* determines the time from the beginning of the computation. The optimal step-length α^* can be computed by explicit formulas given in the appendix (in the case of D- or A-optimality) or by a procedure for the one-dimensional convex optimization on a finite interval.

4. **Stopping rule.** If a stopping rule is satisfied, stop. Otherwise, iterate by returning to step 1.

We remark that for the criteria of D- and A-optimality, we have rapid explicit formulas providing the optimal exchanges in steps 1 and 3b of REX; see Appendix A. However, in principle, REX can also be applied to any other concave (even non-differentiable) optimality criterion using numerical procedures for the one-dimensional convex optimization on an interval, provided that we find a suitable analogy to the function $\mathbf{g}(\cdot)$.

REX makes explicit use of the fast optimal weight exchange between two design points, which makes it much less dependent on m than YBT. Moreover, REX requires less frequent re-computations of the function $\mathbf{g}(\mathbf{w})$, which leads to significant computational savings compared to VEM for large n. Note that REX is very simple. In particular, it is simpler than the hybrid cocktail algorithm of Yu [48]. Moreover, unlike the method of [48], the result does not depend on the ordering of the design points. Finally, as we demonstrate, in most cases, it is also numerically more efficient than both of the state-of-the art methods proposed in [48] and [46].

3.2 Connection to existing literature on subspace descent methods

Subspace ascent/descent methods of various types have recently been extensively studied in the optimization [26, 36, 12], linear algebra [22, 21, 14], machine learning [41, 32, 20] and image processing [7] literature. Of particular importance are the *randomized* and *greedy* variants.

Randomized subspace descent methods operate by taking steps in *random* subspaces of the domain of interest according to some distribution over subspaces (all of which typically have a relatively small and fixed dimension) that is fixed throughout the iterative process. Once a subspace has been identified, a gradient, Newton or a proximal point step is usually taken in the subspace. These methods have been found to scale to extremely large dimensions for certain applications [36, 37] (e.g., billions of unknowns), and have become the state of the art for a variety of big data analysis tasks. Randomized selection rules are, in practice, often better than cyclic rules that pass through the subspaces in a fixed predefined order. Intuitively, this happens because randomization helps avoid/break unfavorable orderings. It was recently shown by Sun and Ye [42] that there is a large theoretical gap between randomized and cyclic rules (proportional to the square of the dimension of the problem), which favors randomized rules.

Greedy subspace descent methods operate by taking steps in subspaces selected *greedily* according to some potential/importance function of interest. Again, once a subspace has been identified, typically a gradient, Newton or a proximal point step is taken in the subspace. The ideal greedy method employs the *maximum improvement* rule, which requires that at any given iteration, one should choose the subspace that leads to the maximum improvement in the objective. Since this is almost always impractical, one needs to resort to approximating the maximum improvement rule by an efficiently implementable proxy rule. Whether one should use a greedy or a randomized method depends on the problem structure, as both have applications in which they dominate.

Let us now highlight some of the key similarities and differences between REX and existing subspace descent methods:

1. **Support plays a role.** The subspace selection rule in REX has a *greedy* component through the inclusion of the set S_{greedy}. However, unlike most existing subspace descent methods, it also has an "active set / support" component through the inclusion of S_{support}.

2. **Greedy subspaces.** Greedy subspace rules beyond subspaces of dimension one have not been explored in the optimization and machine learning literature. To the best of our knowledge, the only exception is the work of Csiba and Richtárik [9]. However, both their greedy subspace rule and the problem that they tackle is different from ours.

3. **Subspace step.** While existing subspace descent methods typically rely on traditional deterministic (as opposed to stochastic) optimization steps such as gradient or Newton steps, the REX performs a *randomized* pairwise exchange step in the subspace. We perform pairwise exchange steps as for the most important optimal design criteria. These can be calculated exactly in closed-form, which facilitates fast computations. This is similar to the sequential minimal optimization technique of Platt [29] that is influential in the machine learning literature. Methods with randomized steps performed in the subspace, such as REX, are rare in the optimization literature. One of the earliest examples of such an approach is the CoCoA framework of Jaggi et al. [19], aimed at distributed primal-dual optimization in machine learning. Both their subspace subroutine and the problem that they tackle are very different from ours.

4. **Random reshuffling.** Randomization in REX is present in the form of a *random permutation* of pairs from $S_{\text{support}} \times S_{\text{greedy}}$. Methods relying on random permutations can be seen as a hybrid between stochastic and cyclic methods. These approaches are not theoretically understood due the intrinsic difficulty in capturing their behaviors using complexity analysis. The work of Gürbüzbalaban et al. [15] provides one of the first insights into this problem. However, their techniques do not apply to our problem or algorithm.

5. **Nonseparable constraints.** Virtually all modern stochastic and greedy subspace descent methods in optimization apply to unconstrained problems[7], or to problems with separable constraints structure, as this property is crucial in the analysis of these methods. One of the early works that explicitly tackles non-separable constraints is that of Necoara et al. [25]. However, their work applies to linear constraints only.

6. **Smoothness.** Vast majority of papers on stochastic and greedy subspace descent methods assume that the objective function has a Lipschitz continuous gradient (this property is often called L-smoothness in the machine learning literature). However, this is not true in our case. Hence, fundamentally different convergence analysis techniques are needed in our case. Only a handful of subspace descent methods do not rely on this assumption. One of them is the stochastic Chambolle-Pock algorithm [7], which is a stochastic extension of a famous state-of-the-art method in the area of computational imaging: the Chambolle-Pock method.

4 Numerical comparisons

In this section, we compare the performance of an `R` ([33]) implementation of REX and two state-of-the-art algorithms for computing the D-optimal design of experiments. As a first candidate, we used the cocktail algorithm (CO), more precisely its `R` implementation, which was made available by the author of CO (see [48]).[8] As a second algorithm, we selected the Newton-type method [46] (YBT). For a fair comparison, the `SAS` code kindly provided by the authors of YBT was converted into `R` with as high fidelity as possible. The `R` codes of REX are available at

http://www.iam.fmph.uniba.sk/ospm/Harman/design/.

Note that it is possible to directly use the codes without any commercial software and without technical knowledge of the algorithms.

In the following two subsections, we compare the competing algorithms for two very different, generic D-optimum design problems with widely varying sizes. This approach provides comprehensive information about the relative strengths of the methods in a broad range of situations. However, we remark that the numerical comparisons must be taken with a grain of salt because they depend on a multitude of factors. Different implementations and different hardware capabilities can lead to somewhat different relative results.

[7]Problems with a convex constraint admitting a fast (i.e., closed form) projection are considered equally simple as unconstrained problems.

[8]There are two variants of CO. One has a pre-specified neighborhood structure, and the other has nearest neighbors computed on the fly. For each model, we selected the variant that performed better.

In all computations with REX, we chose the tuning constant $\gamma = 4$, uniformly random m-point initial designs. We used a computer with a 64-bit Windows 10 operating system running an Intel Core i7-5500U CPU processor at 2.40 GHz with 8 GB of RAM.

4.1 Quadratic models

Consider the full quadratic regression model on the d-dimensional cube $[-1,1]^d$, $d \in \mathbb{N}$. In other words, for $(t_1, \ldots, t_d)' \in [-1,1]^d$ the observations are modeled as

$$Y(t_1, \ldots, t_d) = \beta_1 + \sum_{i=1}^{d} \beta_{i+1} t_i + \sum_{j=1}^{d} \sum_{k=j}^{d} \tilde{\beta}_{i,j} t_i t_j + \varepsilon(t_1, \ldots, t_d), \tag{6}$$

where $\tilde{\beta}_{1,1}, \tilde{\beta}_{1,2}, \ldots, \tilde{\beta}_{d,d}$ correspond to the parameters $\beta_{d+1}, \beta_{d+2}, \ldots, \beta_m$, $m = (d+1)(d+2)/2$. We discretized the cube $[-1,1]^d$ to obtain a lattice $\mathcal{D}$ of n points, which are numbered by indices from $\mathfrak{X} = \{1, \ldots, n\}$. Thus, the set of regressors $\{\mathbf{f}(1), \ldots, \mathbf{f}(n)\}$ corresponds to $\{(1, t_1, t_2, \ldots, t_{d-1}t_d, t_d^2)' : (t_1, \ldots, t_d)' \in \mathcal{D}\}$. These are representatives of structured models, such as those used in the response surface methodology (see, e.g., [24]). Each of the above mentioned algorithms was run for various combinations of values n and d (see Figure 1).

The numerical results confirm the claim of [46] that YBT is superior to CO for a large size n of the design space and a small number m of model parameters (see panels (b) and (f) of Figure 1). However, for a smaller design space, particularly if the number of parameters is large, the algorithm CO tends to perform better than YBT (cf. panels (a), (c) and (e) of Figure 1). A similar observation can also be drawn for other models, such as the one presented in the next subsection.

Importantly, the performance of REX is comparable or superior to both state-of-the-art algorithms for all combinations of n and m that we explored.

4.2 Random models

As a second example, we chose a very different problem in which the regressors $\mathbf{f}(1), \ldots, \mathbf{f}(n)$ are drawn independently and randomly from the m-dimensional normal distribution $N_m(\mathbf{0}, \mathbf{I})$. These models represent unstructured and unordered optimal design situations in which the possible covariates of the model are drawn from a random population. The results are demonstrated in Figure 2.

The random models again show the superior performance of the proposed algorithm. The differences in favor of REX are even more pronounced compared to the quadratic models. This is possibly caused by the fact that on the unstructured set of regressors, procedures such as CO that depend on the ordering of design points are disadvantaged. Note that the algorithm REX is independent of the ordering of the design points.

4.3 Further remarks on the numerical comparisons

***A*-optimality.** In addition to D-optimality, we performed a comparison of CO, YBT and REX for several problems under the criterion of A-optimality. Note that the algorithm CO was originally developed for D-optimality only, but the formulas given in the Appendix (Subsection A.2) allowed

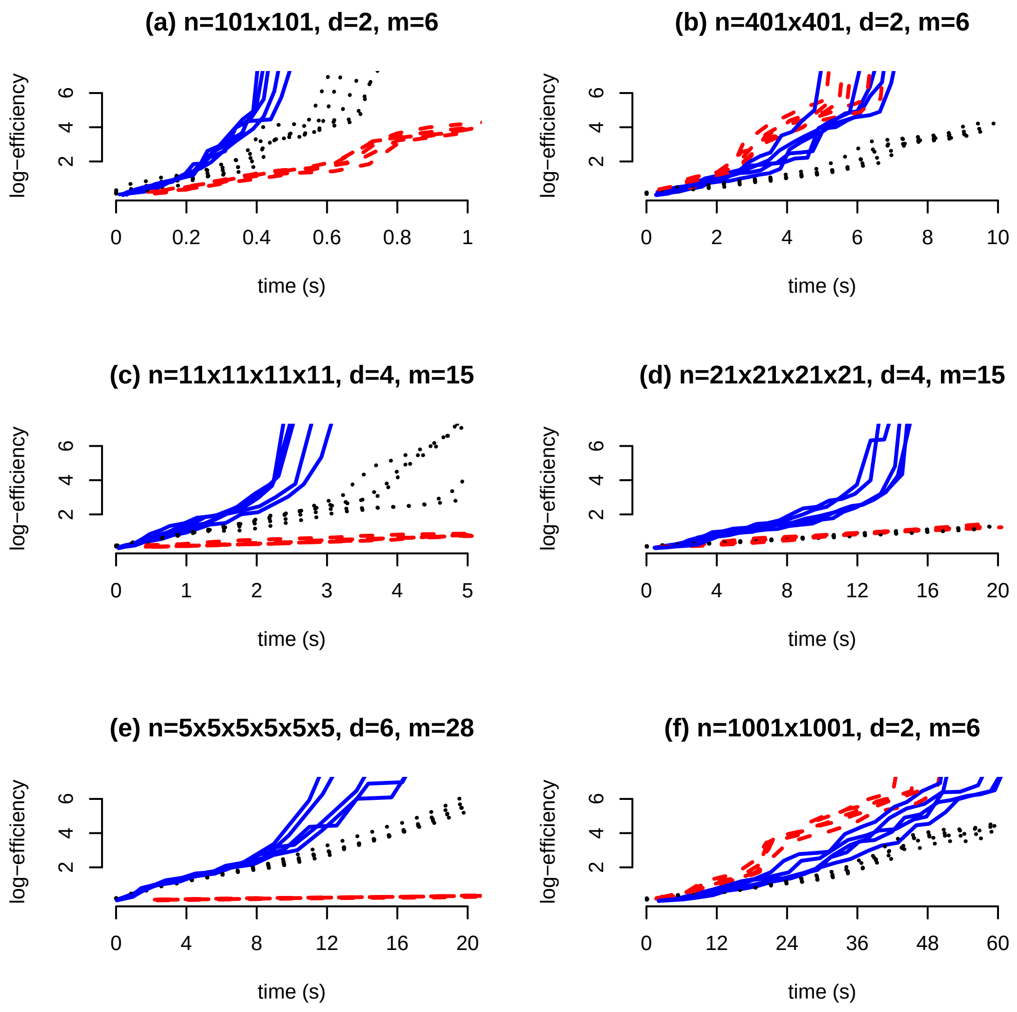

Figure 1: Typical performance of the algorithms CO (black dotted line), YBT (red dashed line) and REX (blue solid line) for D-optimality in the case of quadratic models (6) on an n-point discretization of $[-1,1]^d$. The vertical axis denotes the log-efficiency $-\log_{10}(1-\mathrm{eff})$, where eff is the D-efficiency of design. Thus, log-efficiencies 2, 4 and 6 correspond to D-efficiencies 0.99, 0.9999 and 0.999999, respectively. The horizontal axis corresponds to the computation time in seconds. Each algorithm was run 5 times to illustrate the degree of variability.

us to modify it for the computation of A-optimal designs. For A-optimality, the relative efficiencies of the three algorithms were similar to the case of D-optimality, although the advantage of REX is generally less pronounced.

Other algorithms for D-optimality. For some models with a small number of design points and a very large number of parameters, the classical methods, particularly the multiplicative algorithm, outperform all three "state-of-the-art" algorithms (REX, YBT, CA), at least in an initial part

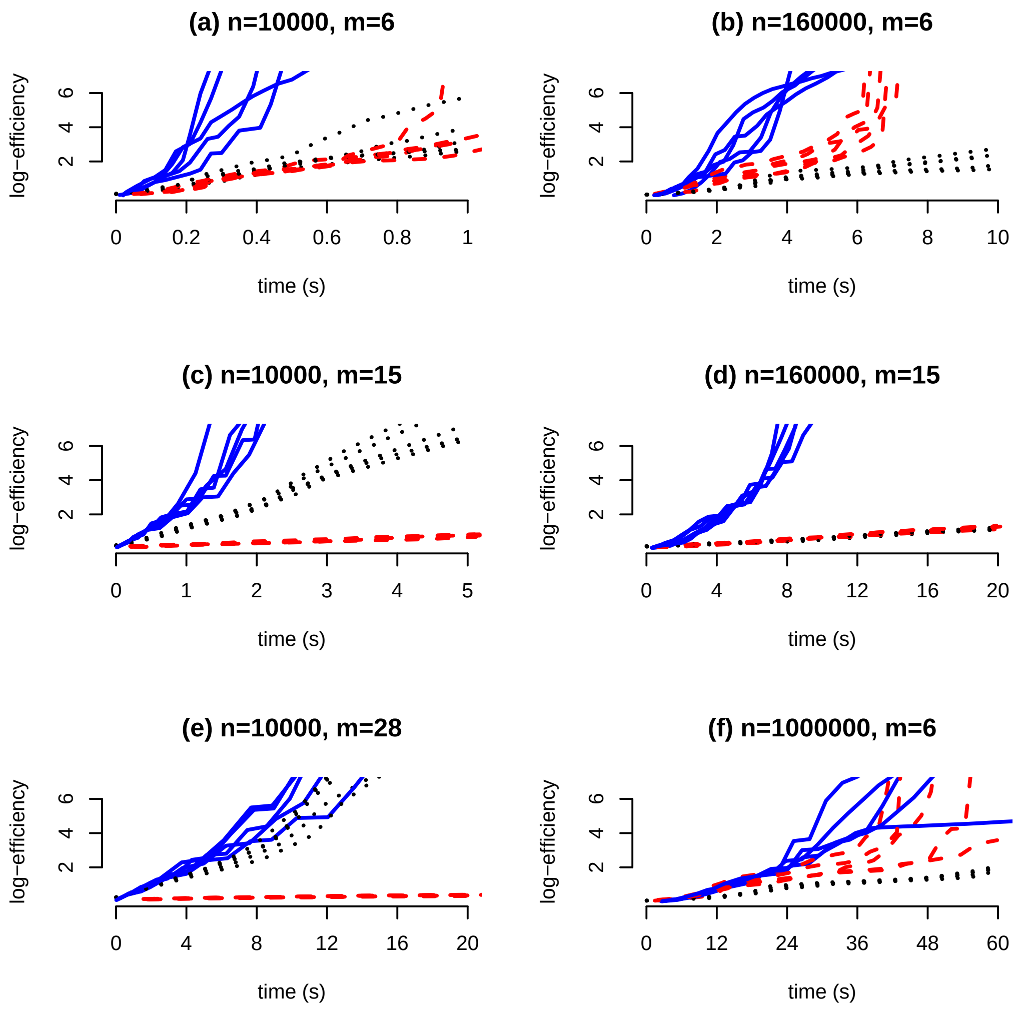

Figure 2: Typical performance of the algorithm CO (black dotted line), YBT (red dashed line) and REX (blue solid line) for D-optimality in the case of models with n Gaussian random regressors of dimension m. The vertical axis denotes the log-efficiency $-\log_{10}(1 - \text{eff})$, where eff is the D-efficiency. The horizontal axis corresponds to the computation time. Each algorithm was run 5 times.

of the computation. However, in a vast majority of test problems, the classical methods cannot compete with the three modern algorithms. A comprehensive comparison of all available methods is beyond the scope of this paper.

Disadvantages of REX. We believe it to be important that the reader is informed of the disadvantages of our method we are aware of. First, the method occasionally generates streaks of only slowly increasing criterion values that are, however, usually followed by a rapid improvement.

Second, we were not able to prove the convergence to the optimum[9] except for the case of D-optimality. Nevertheless, we also tested REX for other criteria, and the algorithm converged in all test problems.

5 Conclusions

As we have numerically demonstrated, the randomized exchange algorithm (REX) largely outperforms recent state-of-the-art methods for computing D-optimal approximate designs of experiments, mostly in the cases where the model consists of randomly generated or otherwise unstructured regressors. Overall, in comparison to the competing methods, the performance of REX deteriorates much less when both the size of the design space and the number of parameters increase. It is also worth noting that REX is concise, has relatively low memory requirements, and does not utilize any advanced mathematical programming solvers. In addition, for D-optimality, we theoretically proved the convergence of REX to the correct optimum. To adapt the proposed algorithm to the A-optimality criterion, we derived appropriate vertex-exchange formulas.

REX offers many possibilities for further development. Besides the exploration of its convergence properties for a general concave criterion, there is a space for improvement in an optimized, perhaps adaptive selection of the parameter γ that regulates the batch size. One way to further increase the speed of the algorithm is to incorporate specific methods of initial design construction and the deletion rules of non-supporting points (see [16] for D-optimality and [30] for A-optimality).

Moreover, it is possible to employ REX for the computations of optimal designs on continuous (infinite) design spaces. One approach is to choose a large number of random points inside the continuous design space, use the speed of REX for unstructured design spaces to construct the optimal approximate design on the finite sub-sample, and fine-tune the design by using standard routines of constrained continuous optimization. However, a numerical and theoretical analysis of this methodology is beyond the scope of this paper.

Acknowledgments The work of the first two authors was supported by Grant Number 1/0521/16 from the Slovak Scientific Grant Agency (VEGA). The last author acknowledges support through the KAUST baseline research funding scheme.

References

[1] Ahipasaoglu SD (2015): "A first-order algorithm for the A-optimal experimental design problem: a mathematical programming approach." Statistics and Computing, Vol 25.6, pp. 1113-1127.

[2] Ahipasaoglu SD (2015): "Fast algorithms for the minimum volume estimator." Journal of Global Optimization, Vol 62.2, pp. 351-370.

[9] Note that the convergence of the criterion values per-se follows trivially from the monotonicity of REX. However, the convergence to the value *optimal* for the problem at hand seems to be difficult to prove for a general concave criterion.

[3] Atkinson AC, Donev AN, Tobias RD (2007): Optimum Experimental Designs, With SAS. Oxford University Press

[4] Atwood, CL (1973): "Sequences converging to D-optimal designs of experiments", The Annals of Statistics 1, pp. 342-352

[5] Atwood CL (1976): "Convergent design sequences, for sufficiently regular optimality criteria." The Annals of Statistics 4.6, pp. 1124-1138.

[6] Böhning D (1986): "A vertex-exchange-method in D-optimal design theory." Metrika 33.1: pp. 337-347.

[7] Chambolle A, Ehrhardt MJ, Richtárik P, Schönlieb CB (2017): "Stochastic primal-dual hybrid gradient algorithm with arbitrary sampling and imaging applications." arXiv:1706.04957, pp. 1-25.

[8] Cook RD, Nachtsheim CJ (1982): "Model robust, linear-optimal designs." Technometrics 24.1, pp. 49-54.

[9] Csiba D, Richtárik P (2017): "Global convergence of arbitrary-block gradient methods for generalized Polyak-Lojasiewicz functions." arXiv preprint arXiv:1709.03014.

[10] Dette H, Pepelyshev A, Zhigljavsky A (2008): "Improving updating rules in multiplicative algorithms for computing D-optimal designs." Computational Statistics & Data Analysis 53.2, pp. 312-320.

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

[12] Fercoq O, Richtárik P (2015): "Accelerated, Parallel and PROXimal coordinate descent." SIAM Journal on Optimization, Vol 25.4, pp. 1997-2023.

[13] Goos P, Jones B, Syafitri U (2016): "I-Optimal Design of Mixture Experiments", Journal of the American Statistical Association, Vol. 111, pp. 899-911.

[14] Gower RM, Richtárik P (2015): "Randomized iterative methods for linear systems." SIAM Journal on Matrix Analysis and Applications, Vol 36.4, pp. 1660-1690.

[15] Gürbüzbalaban M, Ozdaglar A, Parrilo P (2015): "Why random reshuffling beats stochastic gradient descent", arXiv preprint arXiv:1510.08560.

[16] Harman R, Pronzato L (2007): "Improvements on removing nonoptimal support points in D-optimum design algorithms", Statistics & Probability Letters, Vol 77.1, pp. 90-94.

[17] Harman R (2014): "Multiplicative methods for computing D-optimal stratified designs of experiments", Journal of Statistical Planning and Inference, Vol 146, pp. 82-94.

[18] Harman R, Filová L (2016): "Package 'OptimalDesign'", https://cran.r-project.org/web/packages/OptimalDesign/index.html

[19] Jaggi M, Smith V, Takáč M, Terhorst J, Krishnan S, Hofmann T, Jordan MI (2014): "Communication-efficient distributed dual coordinate ascent", Advances in Neural Information Processing Systems, pp. 3068-3076.

[20] Konečný J, Liu J, Richtárik P, Takáč M (2016): "Mini-batch semi-stochastic gradient descent in the proximal setting." IEEE Journal of Selected Topics in Signal Processing, Vol 10.2, pp. 242-255.

[21] Lee YT, Sidford A (2010): "Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems." IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS).

[22] Leventhal D, Lewis AS (2010): "Randomized methods for linear constraints: convergence rates and conditioning." Mathematics of Operations Research, Vol 35, pp. 641-654.

[23] Mandal S, Torsney B (2006): "Construction of optimal designs using a clustering approach", Journal of Statistical Planning and Inference, Vol 136.3, pp. 1120-1134.

[24] Myers RH, Montgomery DC, Anderson-Cook CM (2016): "Response surface methodology: process and product optimization using designed experiments", John Wiley & Sons.

[25] Necoara I, Nesterov Yu, Glineur F (2011): "A random coordinate descent method on large optimization problems with linear constraints", Manuscript.

[26] Nesterov Yu (2012): "Efficiency of coordinate descent methods on huge-scale optimization problems." SIAM Journal on Optimization, Vol 22, pp. 341-362.

[27] Nutini, J, Schmidt M, Laradji IH, Friedlander M, Koepke H (2015): "Coordinate descent converges faster with the Gauss-Southwell rule than random selection", Proceedings of the 32nd International Conference on Machine Learning.

[28] Pázman A (1986): "Foundations of Optimum Experimental Design". Reidel.

[29] Platt J (1998): "Sequential minimal optimization: A fast algorithm for training support vector machines". Manuscript.

[30] Pronzato L (2013): "A delimitation of the support of optimal designs for Kiefers ϕ_p-class of criteria." Statistics & Probability Letters, Vol 83.12, pp. 2721-2728.

[31] Pukelsheim F (2006): "Optimal Design of Experiments (Classics in Applied Mathematics)", Society for Industrial and Applied Mathematics.

[32] Qu Z, Richtárik P, Zhang T (2015): "Quartz: Randomized dual coordinate ascent with arbitrary sampling." In Advances in Neural Information Processing Systems, Vol 28, pp. 865-873.

[33] R Core Team (2016): "R: A language and environment for statistical computing." R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

[34] Richtárik P, Takáč M (2016): "On optimal probabilities in stochastic coordinate descent methods." Optimization Letters, Vol 10.6, pp. 1233-1243.

[35] Richtárik P, Takáč M (2014): "Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function ." Mathematical Programming Vol 156.1, pp. 1-38.

[36] Richtárik P, Takáč M (2016): “Parallel coordinate descent methods for big data optimization.” Mathematical Programming, Vol 144.2, pp. 433-484.

[37] Richtárik P, Takáč M (2016): “Distributed coordinate descent method for learning with big data.” Journal of Machine Learning Research, Vol 17.75, pp. 1-25.

[38] Sagnol G (2011): ”Computing optimal designs of multiresponse experiments reduces to second-order cone programming”, Journal of Statistical Planning and Inference, Vol 121, pp. 1684-1708.

[39] Sagnol G, Harman R (2015): “Computing exact D-optimal designs by mixed integer second order cone programming”, The Annals of Statistics, Vol 43.5, pp. 2198-2224.

[40] Silvey SD, Titterington DH, Torsney B (1978): “An algorithm for optimal designs on a design space.” Communications in Statistics-Theory and Methods, Volume 7.14, pp. 1379-1389.

[41] Shalev-Shwartz S, Zhang T (2013): “Stochastic dual coordinate ascent methods for regularized loss minimization.” Journal of Machine Learning Research, Vol 14, pp. 567-599.

[42] Sun R, Ye Y (2016): “Worst-case complexity of cyclic coordinate descent: $O(n^2)$ gap with randomized version.” arXiv:1604.07130, pp. 1-39.

[43] Todd MJ, Yildirim EA (2007): “On Khachiyan’s algorithm for the computation of minimum-volume enclosing ellipsoids”, Discrete Applied Mathematics, Vol 155.13, pp. 1731-1744.

[44] Todd MJ (2016): “Minimum-Volume Ellipsoids: Theory and Algorithms”. Society for Industrial and Applied Mathematics.

[45] Ucinski D, Patan M (2007): “D-optimal design of a monitoring network for parameter estimation of distributed systems”, Journal of Global Optimization, Vol 39,, pp. 291-322

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

[47] Yu Y (2010): “Monotonic convergence of a general algorithm for computing optimal designs”, The Annals of Statistics, Vol 38.3, pp. 1593-1606.

[48] Yu Y (2011): “D-optimal designs via a cocktail algorithm.” Statistics and Computing, Vol 21.4, pp. 475-481.

[49] Vandenberghe L, Boyd S, Wu S (1998): “Determinant maximization with linear matrix inequality constraints”, SIAM Journal on Matrix Analysis, Vol 19, pp. 499-533.

[50] Wynn HP (1970): “The sequential generation of D-optimum experimental designs.” The Annals of Mathematical Statistics, Vol 41.5, pp. 1655-1664.

A Optimal Step-length and Convergence

Let $\mathbf{w} \in \Xi$, $u, v \in \mathfrak{X}$, and let $\Phi : \mathcal{S}^m_+ \to [0, \infty)$ be a criterion of design optimality. The *optimal step-length* of weight exchange in the design $\mathbf{w}$ between design points u and v is any

$$\alpha^*_{u,v}(\mathbf{w}) \in \operatorname{argmax}\{\Phi(\mathbf{M}[\mathbf{w} + \alpha(\mathbf{e}_v - \mathbf{e}_u)]) : \alpha \in [-w_v, w_u]\}. \tag{7}$$

In accord with the definition (7), we provide an optimal step-length formula for any design $\mathbf{w} \in \Xi_R$. We also prove the convergence of the REX algorithm for D-optimality.

A.1 D-optimality: Optimal Step-length and a Proof of Convergence of REX

Let $\mathbf{w} \in \Xi_R$ and $u, v \in \mathfrak{X}$. If $w_u = w_v = 0$, the optimal step-length is trivially $\alpha^*_{u,v}(\mathbf{w}) = 0$. Therefore, we can assume that at least one of the weights w_u and w_v is strictly positive. We use the notation

$$d_{u,v}(\mathbf{w}) := \mathbf{f}'(u)\mathbf{M}^{-1}(\mathbf{w})\mathbf{f}(v).$$

For D-optimality, it has been shown (see [6], [48]) that an optimal choice of the step-length in (7) is as follows.

If $\mathbf{f}(u)$ and $\mathbf{f}(v)$ are linearly independent, then[10]

$$\alpha^*_{u,v}(\mathbf{w}) = \min\left\{w_u, \max\left\{-w_v, \frac{d_v(\mathbf{w}) - d_u(\mathbf{w})}{2[d_u(\mathbf{w})d_v(\mathbf{w}) - d^2_{u,v}(\mathbf{w})]}\right\}\right\}. \tag{8}$$

If $\mathbf{f}(u)$ and $\mathbf{f}(v)$ are linearly dependent, we can[11] set

$$\alpha^*_{u,v}(\mathbf{w}) = \begin{cases} w_u & \text{if } d_u(\mathbf{w}) < d_v(\mathbf{w}), \\ 0 & \text{if } d_u(\mathbf{w}) = d_v(\mathbf{w}), \\ -w_v & \text{if } d_u(\mathbf{w}) > d_v(\mathbf{w}). \end{cases} \tag{9}$$

A D-optimal vertex-exchange step for $\mathbf{w}$ between design points u, v is then

$$T^D_{u,v}(\mathbf{w}) := \mathbf{w} + \alpha^*_{u,v}(\mathbf{w})(\mathbf{e}_v - \mathbf{e}_u).$$

The D-optimal vertex exchange step is called nullifying if $\alpha^*_{u,v}(\mathbf{w}) = w_u$ or if $\alpha^*_{u,v}(\mathbf{w}) = -w_v$, that is, if the u-th or the v-th component of $T^D_{u,v}(\mathbf{w})$ is equal to 0. The pair of indices

$$\begin{aligned} u^*(\mathbf{w}) &\in \operatorname{argmin}\{d_u(\mathbf{w}) : u \in \operatorname{supp}(\mathbf{w})\}, \\ v^*(\mathbf{w}) &\in \operatorname{argmax}\{d_v(\mathbf{w}) : v \in \mathfrak{X}\} \end{aligned}$$

will be called the D-optimal Bohning's vertex pair and the steplength $\alpha^*_{u,v}(\mathbf{w})$ for $u = u^*(\mathbf{w})$, $v = v^*(\mathbf{w})$, will be called the D-optimal Bohning's step. In the following, we will use the well-known facts (i) $d_{u^*(\mathbf{w})}(\mathbf{w}) \le m$, (ii) $d_{v^*(\mathbf{w})}(\mathbf{w}) \ge m/\mathrm{eff}_D(\mathbf{w})$, and (iii) $\mathbf{w}$ is D-optimal if and only if $d_{v^*(\mathbf{w})}(\mathbf{w}) = m$ (see, e.g., [3], 9.2). In (ii), $\mathrm{eff}_D(\mathbf{w})$ denotes the D-efficiency of $\mathbf{w}$ relative to a D-optimal design.

[10] Note that if $\mathbf{f}(u)$ and $\mathbf{f}(v)$ are linearly independent, the Cauchy-Schwarz inequality implies $d_u(\mathbf{w})d_v(\mathbf{w}) > d^2_{u,v}(\mathbf{w})$.

[11] For $d_u(\mathbf{w}) = d_v(\mathbf{w})$, the choice of the optimal $\alpha^*_{u,v}(\mathbf{w})$ is not unique.

Lemma 1. *Let $u, v \in \mathfrak{X}$ and $\mathbf{w} \in \Xi_R$. Then*

$$\det(\mathbf{M}(\mathbf{w})) \leq \det(\mathbf{M}[T^D_{u,v}(\mathbf{w})]), \text{ and} \tag{10}$$

$$\{T^D_{u,v}(\mathbf{w}) \neq \mathbf{w}\} \Rightarrow \{\det(\mathbf{M}(\mathbf{w})) < \det(\mathbf{M}[T^D_{u,v}(\mathbf{w})])\}. \tag{11}$$

Moreover, if $\mathbf{w}$ is not D-optimal and $u = u^*(\mathbf{w})$, $v = v^*(\mathbf{w})$ is the D-optimal Bohning's vertex pair, then $\det(\mathbf{M}(\mathbf{w})) < \det(\mathbf{M}[T^D_{u,v}(\mathbf{w})])$.

Proof. Inequality (10) follows from the choice of $\alpha^*_{u,v}(\mathbf{w})$. We will prove (11). If $T^D_{u,v}(\mathbf{w}) \neq \mathbf{w}$, then $\alpha^*_{u,v}(\mathbf{w}) \neq 0$. Assume that $\alpha^*_{u,v}(\mathbf{w}) > 0$. Then the rules for $\alpha^*_{u,v}(\mathbf{w})$ imply $d_u(\mathbf{w}) < d_v(\mathbf{w})$ and $w_u > 0$. Now, to show that $\det(\mathbf{M}(\mathbf{w})) < \det(\mathbf{M}[T^D_{u,v}(\mathbf{w})])$, it is enough to verify that the directional derivative of $\log\det(\cdot)$ in $\mathbf{M}(\mathbf{w})$ in the direction of $\mathbf{M}[T^D_{u,v}(\mathbf{w})]$ is strictly positive. Noting that $\mathbf{M}^{-1}$ is the gradient of $\log\det(\cdot)$ in a non-singular $\mathbf{M}$, we obtain that the directional derivative is

$$\begin{aligned}
\partial \log\det(\mathbf{M}(\mathbf{w}), \mathbf{M}[T^D_{u,v}(\mathbf{w})]) &= \operatorname{tr}(\mathbf{M}^{-1}(\mathbf{w})(\mathbf{M}[T^D_{u,v}(\mathbf{w})] - \mathbf{M}[\mathbf{w}])) \\
&= \operatorname{tr}(\mathbf{M}^{-1}(\mathbf{w})[\alpha^*_{u,v}(\mathbf{w})(\mathbf{f}(v)\mathbf{f}'(v) - \mathbf{f}(u)\mathbf{f}'(u))]) \\
&= \alpha^*_{u,v}(\mathbf{w})(d_v(\mathbf{w}) - d_u(\mathbf{w})) > 0.
\end{aligned}$$

For the case $\alpha^*_{u,v}(\mathbf{w}) < 0$, the strict inequality in (11) can be proved analogously.

Finally, we prove the last statement of the theorem. If $\mathbf{w} \in \Xi_R$ is not optimal, then $d_{v^*(\mathbf{w})}(\mathbf{w}) > m$, i.e., (i) implies that $d_{u^*(\mathbf{w})}(\mathbf{w}) < d_{v^*(\mathbf{w})}(\mathbf{w})$. However, $u^*(\mathbf{w}) \in \operatorname{supp}(\mathbf{w})$, therefore $w_{u^*(\mathbf{w})} > 0$, and the formulas for the optimal step-length imply $\alpha^*_{u,v}(\mathbf{w}) > 0$, i.e., $T^D_{u,v}(\mathbf{w}) \neq \mathbf{w}$. We can close the proof by using (11). □

Recall that for $\mathbf{w} \in \Xi_R$ a Bohning's pair $u = u^*(\mathbf{w}), v = v^*(\mathbf{w}) \in \mathfrak{X}$ is called nullifying if $\alpha^*_{u,v}(\mathbf{w})$ is equal to $-w_v$ or w_u.

Lemma 2. *For any $0 < \epsilon < 1$, there exists $K_\epsilon < \infty$ such that for all $\mathbf{w} \in \Xi_R$ satisfying $\operatorname{eff}_D(\mathbf{w}) \in [\epsilon, 1)$ and a non-nullifying D-optimal Bohning's pair $u = u^*(\mathbf{w})$, $v = v^*(\mathbf{w})$, we have*

$$\operatorname{eff}_D\left(T^D_{u,v}(\mathbf{w})\right) \geq \operatorname{eff}_D(\mathbf{w}) \left[1 + \frac{(\operatorname{eff}^{-1}_D(\mathbf{w}) - 1)^2}{K_\epsilon}\right]^{1/m}.$$

Proof. Let $\tilde{u}, \tilde{v} \in \mathfrak{X}$ and $D_{\tilde{u},\tilde{v}}(\mathbf{w}) := d_{\tilde{u}}(\mathbf{w})d_{\tilde{v}}(\mathbf{w}) - d^2_{\tilde{u},\tilde{v}}(\mathbf{w})$. Clearly, $4m^{-2}D_{\tilde{u},\tilde{v}}(\mathbf{w})$ is a continuous function of $\mathbf{w}$ on the compact $\Xi_\epsilon := \{\mathbf{w} \in \Xi : \operatorname{eff}_D(\mathbf{w}) \in [\epsilon, 1]\} \subset \Xi_R$. Therefore, it is bounded on Ξ_ϵ from above by some constant $K_{\tilde{u},\tilde{v},\epsilon} < \infty$. Set $K_\epsilon := \max_{\tilde{u},\tilde{v} \in \mathfrak{X}} K_{\tilde{u},\tilde{v},\epsilon}$, which is clearly finite.

Assume first that $\mathbf{f}(u)$ and $\mathbf{f}(v)$ are linearly dependent. Then, the Bohning's step can only be non-nullifying if $d_u(\mathbf{w}) = d_v(\mathbf{w})$. However, this cannot happen because $d_u(\mathbf{w}) \leq m$ from fact (i) and since $\operatorname{eff}_D(\mathbf{w}) < 1$, $d_v(\mathbf{w}) > m$ from fact (iii).

Therefore, $\mathbf{f}(u)$ and $\mathbf{f}(v)$ are linearly independent. Then, since the D-optimal Bohning's step given by (8) is assumed to be non-nullifying, we have $\alpha^*_{u,v}(\mathbf{w}) = [d_v(\mathbf{w}) - d_u(\mathbf{w})]/[2D_{u,v}(\mathbf{w})]$. From the matrix determinant lemma and the Sherman-Morrison formula, it is straightforward to show that for any positive definite $m \times m$ matrix $\mathbf{M}$, any $\alpha \in \mathbb{R}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$

$$\begin{aligned}
&\det(\mathbf{M} + \alpha(\mathbf{y}\mathbf{y}' - \mathbf{x}\mathbf{x}')) = \\
&\det(\mathbf{M}) \left[(1 + \alpha\mathbf{y}'\mathbf{M}^{-1}\mathbf{y})(1 - \alpha\mathbf{x}'\mathbf{M}^{-1}\mathbf{x}) + \alpha^2(\mathbf{y}'\mathbf{M}^{-1}\mathbf{x})^2\right].
\end{aligned} \tag{12}$$

Setting $\alpha = \alpha^*_{u,v}(\mathbf{w})$, $\mathbf{y} = \mathbf{f}(v)$, $\mathbf{x} = \mathbf{f}(u)$ and $\mathbf{M} = \mathbf{M}(\mathbf{w})$ in (12) gives

$$\det(\mathbf{M}(T^D_{u,v}(\mathbf{w}))) = \det(\mathbf{M}(\mathbf{w})) \left(1 + \frac{[d_v(\mathbf{w}) - d_u(\mathbf{w})]^2}{4D_{u,v}(\mathbf{w})}\right).$$

Recalling that $4D_{u,v}(\mathbf{w}) < m^2 K_\epsilon$, $d_u(\mathbf{w}) \leq m$ and $d_v(\mathbf{w}) \geq m/\mathrm{eff}_D(\mathbf{w})$, we obtain the inequality from the lemma. □

Theorem 1. *The algorithm REX with the step-length rule defined by* (8) *and* (9) *converges to a D-optimal design in the sense that the sequence of D-criterion values of designs produced by the algorithm converges to the D-optimal value Φ^*.*[12][13]

Proof. Let the initial design of REX be $\mathbf{w}^0 \in \Xi_R$, let $\epsilon := \mathrm{eff}_D(\mathbf{w}^0)$, and let $(\mathbf{w}^i)_{i=1}^\infty$ be the sequence of designs that REX generates by the leading D-optimal Bohning's exchanges at iterations $1, 2, \ldots$. Because all the transformations used by REX are non-decreasing with respect to Φ_D, the sequence $(\Phi_D(\mathbf{M}(\mathbf{w}^i)))_{i=1}^\infty$ has a limit, say Φ^+. Assume that Φ^+ is not equal to the D-optimal value Φ^* of the problem and let $\mathrm{eff}_+ := \Phi^+/\Phi^* < 1$. Now, inspired by the proof strategy of Bohning ([6]), we will split the proof into two cases.[14] In each case, we show a contradiction, which then implies the claim of the theorem.

Case 1: Assume that there is only a finite number of non-nullifying leading D-optimal Bohning's exchanges. Each nullifying exchange can only decrease the size of the support of the current design, or it can keep the size of the support constant. Since there can only be a finite number of support-reducing nullifying exchanges, there is some index i such that all the exchanges performed after the i-th iteration are nullifying and keep a constant size of the support. Hence, after the i-th iteration, the algorithm does not alter the *set* of all non-zero weights. At the same time, REX is designed such that after the i-th iteration, it will perform only nullifying exchanges (also in the non-leading exchanges). Since $\mathfrak{X}$ is finite, this means that the generated designs must return to one of the previous designs after a finite number of steps. This is impossible because according to Lemma 1, each D-optimal Bohning's step (even if it is nullifying) strictly increases the criterion value of a sub-D-optimal design.

Case 2: Assume that there is an infinite number of non-nullifying leading D-optimal Bohning's steps, at iterations $n_1, n_2, \ldots$. Then, Lemmas 1 and 2 imply that

$$\mathrm{eff}_D(\mathbf{w}^{n_k}) \geq \epsilon \left[1 + \frac{(\mathrm{eff}_+^{-1} - 1)^2}{K_\epsilon}\right]^{k/m} \tag{13}$$

for all $k = 1, 2, \ldots$ and some positive real numbers ϵ, K_ϵ. This is impossible if we assume that $\mathrm{eff}_+ < 1$ because the RHS of (13) converges to infinity with $k \to \infty$ but the efficiency of any design compared to the D-optimal design is bounded by 1 from above. □

A.2 A-optimality: Optimal Step-length

Let $u, v \in \mathfrak{X}$ be a fixed pair of design points and let $\mathbf{w} \in \Xi_R$ be a fixed design. Let at least one of the weights w_u and w_v be strictly positive.

[12]The convergence is the sure convergence of random variables.

[13]The D-optimal value Φ^* is defined as $\Phi_D(\mathbf{M}^*)$, where $\mathbf{M}^*$ is the D-optimal information matrix.

[14]We will deal with Case 1 differently than Bohning.

Denote $\mathbf{f}_u := \mathbf{f}(u)$, $\mathbf{f}_v := \mathbf{f}(v)$, $\mathbf{M} := \mathbf{M}(\mathbf{w})$ and $\mathbf{V} := \mathbf{M}^{-1}(\mathbf{w})$. For α in the open interval $I = (-w_v, w_u)$, the matrix $\mathbf{M} + \alpha\mathbf{f}_v\mathbf{f}_v'$ as well as the information matrix

$$\mathbf{M}_\alpha := \mathbf{M} + \alpha\mathbf{f}_v\mathbf{f}_v' - \alpha\mathbf{f}_u\mathbf{f}_u'$$

are positive definite, and we can use the Woodbury formula to obtain[15]

$$\mathbf{M}_\alpha^{-1} = \mathbf{V} + \frac{\mathbf{V}[\alpha\mathbf{f}_u\mathbf{f}_u' - \alpha\mathbf{f}_v\mathbf{f}_v' - \alpha^2\mathbf{f}_u'\mathbf{V}\mathbf{f}_v(\mathbf{f}_u\mathbf{f}_v' + \mathbf{f}_v\mathbf{f}_u') + \alpha^2(\mathbf{f}_v'\mathbf{V}\mathbf{f}_v\mathbf{f}_u\mathbf{f}_u' + \mathbf{f}_u'\mathbf{V}\mathbf{f}_u\mathbf{f}_v\mathbf{f}_v')]\mathbf{V}}{1 + \alpha\mathbf{f}_v'\mathbf{V}\mathbf{f}_v - \alpha\mathbf{f}_u'\mathbf{V}\mathbf{f}_u - \alpha^2\mathbf{f}_u'\mathbf{V}\mathbf{f}_u\mathbf{f}_v'\mathbf{V}\mathbf{f}_v + \alpha^2(\mathbf{f}_u'\mathbf{V}\mathbf{f}_v)^2},$$

$$-\mathrm{tr}\mathbf{M}_\alpha^{-1} = -\mathrm{tr}\mathbf{V} - \alpha\frac{\mathbf{f}_u'\mathbf{V}^2\mathbf{f}_u - \mathbf{f}_v'\mathbf{V}^2\mathbf{f}_v - 2\alpha\mathbf{f}_u'\mathbf{V}\mathbf{f}_v\mathbf{f}_u'\mathbf{V}^2\mathbf{f}_v + \alpha\mathbf{f}_v'\mathbf{V}\mathbf{f}_v\mathbf{f}_u'\mathbf{V}^2\mathbf{f}_u + \alpha\mathbf{f}_u'\mathbf{V}\mathbf{f}_u\mathbf{f}_v'\mathbf{V}^2\mathbf{f}_v}{1 + \alpha\mathbf{f}_v'\mathbf{V}\mathbf{f}_v - \alpha\mathbf{f}_u'\mathbf{V}\mathbf{f}_u - \alpha^2\mathbf{f}_u'\mathbf{V}\mathbf{f}_u\mathbf{f}_v'\mathbf{V}\mathbf{f}_v + \alpha^2(\mathbf{f}_u'\mathbf{V}\mathbf{f}_v)^2}.$$

Denote

$$d_v = \mathbf{f}_v'\mathbf{V}\mathbf{f}_v,\ d_u = \mathbf{f}_u'\mathbf{V}\mathbf{f}_u,\ d_{uv} = \mathbf{f}_u'\mathbf{V}\mathbf{f}_v.$$
$$a_v := \mathbf{f}_v'\mathbf{V}^2\mathbf{f}_v,\ a_u := \mathbf{f}_u'\mathbf{V}^2\mathbf{f}_u,\ a_{uv} := \mathbf{f}_u'\mathbf{V}^2\mathbf{f}_v,$$

and

$$A = a_v - a_u,\ B = 2d_{uv}a_{uv} - d_ua_v - d_va_u,\ C = d_v - d_u,\ D = d_ud_v - d_{uv}^2. \tag{14}$$

Then

$$-\mathrm{tr}(\mathbf{M}_\alpha^{-1}) = -\mathrm{tr}(\mathbf{V}) + \frac{\alpha A + \alpha^2 B}{1 + \alpha C - \alpha^2 D} =: h(\alpha),$$

The function h is smooth and concave on I. Additionally, if $\mathbf{f}_u \neq \pm\mathbf{f}_v$, then it is also *strictly* concave, which follows from the smoothness and strict concavity of $-\mathrm{tr}(\mathbf{M}^{-1})$ on the set of all positive definite matrices $\mathbf{M}$ (see [28], Prop. IV.3).

Note that the derivative of $h(\alpha)$ in $\alpha \in I$ is

$$\frac{\mathrm{d}h(\alpha)}{\mathrm{d}\alpha} = \frac{A + 2\alpha B + \alpha^2(AD + BC)}{(1 + \alpha C - \alpha^2 D)^2}. \tag{15}$$

We will find $\alpha^* \in \bar{I} := [-w_v, w_u]$ maximizing the continuous extension $\bar{h} : \bar{I} \to \mathbb{R} \cup \{-\infty\}$ of h.

Lemma 3. *$B^2 - A(AD + BC) \geq 0$.*

Proof. It is straightforward to verify that $B^2 - A(AD + BC)$ is equal to

$$\left[(d_u + d_v)^2 - 4d_{uv}^2\right]\left[a_ua_v - a_{uv}^2\right] + \left[d_{uv}(a_u + a_v) - a_{uv}(d_u + d_v)\right]^2. \tag{16}$$

Using the Cauchy-Schwarz inequality and the AM-GM inequality, we obtain $4d_{uv}^2 \leq 4d_ud_v \leq (d_u + d_v)^2$ and $a_{uv}^2 \leq a_ua_v$. Therefore, (16) is non-negative. $\square$

Proposition 1. *If $AD = -BC$ and $B \neq 0$ let $\alpha_s^* = -\frac{A}{2B}$, and if $AD \neq -BC$ let*

$$\alpha_s^* = -\frac{B + \sqrt{B^2 - A(AD + BC)}}{AD + BC}.$$

If $\alpha_s^* \in I$ then α_s^* maximizes $\bar{h}$ on $\bar{I}$. If $AD = -BC$ and $B = 0$ or if $\alpha_s^* \notin I$, let

$$\alpha_n^* = \begin{cases} w_u & \text{if } A > 0, \\ 0 & \text{if } A = 0, \\ -w_v & \text{if } A < 0. \end{cases} \tag{17}$$

Then, α_n^* maximizes $\bar{h}$ on $\bar{I}$.

[15]Note that (12) implies $1 + \alpha\mathbf{f}_v'\mathbf{V}\mathbf{f}_v - \alpha\mathbf{f}_u'\mathbf{V}\mathbf{f}_u - \alpha^2\mathbf{f}_u'\mathbf{V}\mathbf{f}_u\mathbf{f}_v'\mathbf{V}\mathbf{f}_v + \alpha^2(\mathbf{f}_u'\mathbf{V}\mathbf{f}_v)^2$ is strictly positive for $\alpha \in I$.

Proof. The case $\mathbf{f}_u = \pm\mathbf{f}_v$ means that the numerator of (15) is zero, i.e., any $\alpha_s^* \in \bar{I}$ is a maximizer. Therefore, we will consider the non-trivial case $\mathbf{f}_u \neq \pm\mathbf{f}_v$, which implies that h is strictly concave on I. First, note that (15) is equal to A for $\alpha = 0$.

Let $AD = -BC$, $B \neq 0$ and $\alpha_s^* = -A/(2B) \in I$. Then, based on (15), α_s^* is a stationary point of h in I. Therefore, it must maximize $\bar{h}$ on $\bar{I}$.

Let $AD \neq -BC$ and $\alpha_s^* \in I$. Taking Lemma 3 into account, we see that the quadratic function $g(\alpha)$ in the numerator of (15) has real roots on $\mathbb{R}$ given by

$$\alpha_{1,2} = -\frac{B \pm \sqrt{B^2 - A(AD + BC)}}{AD + BC},$$

where $\alpha_s^* = \alpha_1$. Since h is strictly concave on I, it has at most one stationary point on I, which is the unique maximizer on I.

If $AD = -BC$, $B = 0$ or if $AD = -BC$, $B \neq 0$, $\alpha_s^* \notin I$, then (15) has the same sign as A for any $\alpha \in I$, which means that (17) indeed provides a maximizer of $\bar{h}$ on $\bar{I} = [-w_v, w_u]$.

Finally, let $AD \neq -BC$ and $\alpha_s^* \notin I$. It is enough to prove that for $A \neq 0$ none of α_1, α_2 belongs to I. If so, then the sign of the derivative (15) will be equal to the sign of A on the entire interval I, which implies (17). Since we assume that $\alpha_1 = \alpha_s^* \notin I$, it is enough to prove that $\alpha_2 \notin I$.

Let $A > 0$. Consider mutually exclusive cases a) $AD < -BC$ and b) $AD > -BC$. If a) then $\alpha_2 \leq 0$, it means that α_2 cannot be a stationary point of h on I (since the derivative of h in 0 is positive and h is concave on I), hence $\alpha_2 \notin I$. If b), then $\alpha_2 > 0$, but $\alpha_2 > \alpha_1 \geq 0$. As $\alpha_1 \notin I$, we have $\alpha_2 \notin I$. If $A < 0$, we can use an analogous argument. □

Thus, we can determine a maximizer α^* of $\bar{h}$ on $\bar{I}$ as follows:

1. Set $G := AD + BC$
2. If $G = 0$ and $B \neq 0$ set $r := -A/(2B)$. If $-w_v < r < w_u$, return $\alpha^* = r$.
3. If $G \neq 0$ set $r := -(B + \sqrt{B^2 - AG})/G$. If $-w_v < r < w_u$, return $\alpha^* = r$.
4. If $A > 0$ return $\alpha^* = w_u$, else if $A < 0$ return $\alpha^* = -w_v$, else return $\alpha^* = 0$.