Statistics and Computing (2021) 31:70
https://doi.org/10.1007/s11222-021-10046-2

Optimal design of multifactor experiments via grid exploration

Radoslav Harman[1] · **Lenka Filová**[1] · **Samuel Rosa**[1]

Received: 10 April 2021 / Accepted: 22 August 2021 / Published online: 13 September 2021

Abstract
We propose an algorithm for computing efficient approximate experimental designs that can be applied in the case of very large grid-like design spaces. Such a design space typically corresponds to the set of all combinations of multiple genuinely discrete factors or densely discretized continuous factors. The proposed algorithm alternates between two key steps: (1) the construction of exploration sets composed of star-shaped components and separate, highly informative design points and (2) the application of a conventional method for computing optimal approximate designs on medium-sized design spaces. For a given design, the star-shaped components are constructed by selecting all points that differ in at most one coordinate from some support point of the design. Because of the reliance on these star sets, we call our algorithm the galaxy exploration method (GEX). We demonstrate that GEX significantly outperforms several state-of-the-art algorithms when applied to D-optimal design problems for linear, generalized linear and nonlinear regression models with continuous and mixed factors. Importantly, we provide a free R code that permits direct verification of the numerical results and allows researchers to easily compute optimal or nearly optimal experimental designs for their own statistical models.

Keywords Optimal design · Multifactor experiments · Regression models · Generalized linear models · Algorithms

Mathematics Subject Classification 62K05 · 90C59

1 Introduction

The usual aim of the so-called "optimal" design of experiments is to perform experimental trials in a way that enables efficient estimation of the unknown parameters of an underlying statistical model (see, e.g., Fedorov 1972; Pázman 1986; Pukelsheim 2006; Atkinson et al. 2007; Goos and Jones 2011; Pronzato and Pázman 2013). The literature provides optimal designs in analytical forms for many specific situations; for a given practical problem at hand, however, analytical results are often unavailable. In such a case, it is usually possible to compute an optimal or nearly optimal design numerically (e.g., Chapter 4 in Fedorov 1972, Chapter 5 in Pázman 1986, Chapter 12 in Atkinson et al. 2007, and Chapter 9 in Pronzato and Pázman 2013).

In this paper, we propose a simple algorithm for solving one of the most common optimal design problems: computing efficient approximate designs for experiments with uncorrelated observations and several independent factors. The proposed algorithm employs a specific strategy to adaptively explore the grid of factor-level combinations without the need to enumerate all elements of the grid. The key idea of this algorithm is to form exploration sets composed of star-like subsets and other strategically selected points; therefore, we refer to this algorithm as the "galaxy" exploration method (GEX).

If the set of all combinations of factor levels is finite and not too large, it is possible to use many available efficient and provably convergent algorithms to compute an optimal design (e.g., those of Fedorov 1972; Atwood 1973; Silvey et al. 1978; Böhning 1986; Vandenberghe et al. 1998; Uciński and Patan 2007; Yu 2011; Sagnol 2011; Yang et al. 2013; Harman et al. 2020). However, in the case of multiple factors, each with many levels, the number of factor-level combinations is often much larger than the applicability limit of these methods.

The main advantage of GEX is that it can be used to solve problems with an extensive number of combinations of factor levels, e.g., 10^{15} (5 factors, each with 1000 levels), and

✉ Radoslav Harman
harman@fmph.uniba.sk

[1] Department of Applied Mathematics and Statistics, Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava, Bratislava, Slovakia

obtain an optimal design in several seconds. Note that factors with large, yet finite, numbers of levels often correspond to practical requirements on the factor resolution. Even in the theoretical case of fully continuous factors, GEX can be applied; the factors can simply be densely discretized (to, say, 3 decimal places). We will show that this straightforward approach usually outperforms intricate state-of-the-art methods that can choose the factor levels anywhere in given continuous intervals (e.g., those of Pronzato and Zhigljavsky 2014; Duarte et al. 2018; Lukemire et al. 2019; Xu et al. 2019; García-Ródenas et al. 2020). As a byproduct, we demonstrate that it is rarely necessary to discretize each factor to an enormous number of levels to admit an approximate design that is almost perfect compared to the optimum achievable with fully continuous factors.

Let k be the number of experimental factors. Suppose that for any trial, the factors can be independently chosen from predefined finite sets $\mathfrak{X}_1, \ldots, \mathfrak{X}_k \subset \mathbb{R}$ of permissible levels, without mutual constraints. The design space $\mathfrak{X}$ is then the set of all combinations of the factor levels, formally, $\mathfrak{X} = \mathfrak{X}_1 \times \cdots \times \mathfrak{X}_k$. Geometrically, $\mathfrak{X}$ is a finite grid of "design points" $\mathbf{x} = (x_1, \ldots, x_k)^T$.

An approximate experimental design is any probability measure ξ on $\mathfrak{X}$. For $\mathbf{x} \in \mathfrak{X}$, the interpretation of $\xi(\mathbf{x})$ is the proportion of trials to be performed at $\mathbf{x}$. The value $\xi(\mathbf{x})$ is often referred to as the "weight" of $\mathbf{x}$. The set of all design points $\mathbf{x}$ with nonzero weights is called the support of ξ and is denoted by $\mathrm{supp}(\xi)$.

In practical experiments, an approximate design ξ must be converted into an "exact" design of size N determined by the available experimental resources.[1] The exact design assigns nonnegative integer numbers $n_1, \ldots, n_s$ of trials to properly selected points $\mathbf{x}_1, \ldots, \mathbf{x}_s$ such that $\sum_{i=1}^{s} n_i = N$. A standard strategy is to select the points $\mathbf{x}_1, \ldots, \mathbf{x}_s$ that form the support of an optimal or nearly optimal approximate design ξ and then compute the integer numbers of trials through appropriate "rounding" of the numbers

$$N\xi(\mathbf{x}_1), \ldots, N\xi(\mathbf{x}_s);$$

see, e.g., Pukelsheim and Rieder (1992). Alternatively, since the support size s is usually small, it may be feasible to compute the integers $n_1, \ldots, n_s$ by specialized procedures for optimal exact designs (e.g., Chapter 12 of Atkinson et al. 2007) restricted to $\mathrm{supp}(\xi)$.

Computational strategies employing optimal approximate designs lead to exact designs that are typically very efficient (albeit not perfectly optimal) within the class of all exact designs of size N on the full $\mathfrak{X}$, especially when N is large. Importantly, the approximate design approach leads to an optimization problem that is generally much simpler than the problem of computing a perfectly optimal exact design of a given size N. Moreover, the approximate optimal design, once computed, can be rounded to an exact design of any size. In addition, the optimal approximate design can be utilized to provide informative lower bounds on the quality of candidate exact designs.

In the rest of this paper, we will use the term "design" specifically to refer to an approximate experimental design.

The quality of a design ξ is usually expressed as a function of the normalized information matrix

$$\mathbf{M}(\xi) = \sum_{\mathbf{x} \in \mathfrak{X}} \xi(\mathbf{x}) \mathbf{f}(\mathbf{x}) \mathbf{f}^T(\mathbf{x}), \tag{1}$$

where $\mathbf{f} : \mathfrak{X} \to \mathbb{R}^m$, $m \geq 2$, is known. To avoid uninteresting pathological situations, we will assume that $\mathbf{f}(\mathbf{x}_1), \ldots, \mathbf{f}(\mathbf{x}_m)$ are linearly independent for some $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathfrak{X}$. An information matrix of the form given in (1) is typical of models with independent, real-valued observations with nonzero variance, where the stochastic distribution of each observation y depends on the design point $\mathbf{x}$ chosen for the corresponding trial as well as on the unknown m-dimensional vector θ of model parameters.

For linear regression, that is, if $\mathrm{E}(y(\mathbf{x})) = \mathbf{h}^T(\mathbf{x})\theta$ and $\mathrm{Var}(y(\mathbf{x})) = \sigma^2 / w(\mathbf{x})$, where $\mathbf{h} : \mathfrak{X} \to \mathbb{R}^m$ and $w : \mathfrak{X} \to (0, \infty)$ are known and $\sigma^2 \in (0, \infty)$ may be unknown, we have

$$\mathbf{f}(\mathbf{x}) = \sqrt{w(\mathbf{x})}\mathbf{h}(\mathbf{x}).$$

However, if $\mathrm{E}(y(\mathbf{x})) = \eta(\mathbf{x}, \theta)$, where $\eta : \mathfrak{X} \times \mathbb{R}^m \to \mathbb{R}$ is nonlinear in θ, the situation is generally much more difficult. In this paper, we adopt the usual simplification, called the approach of "local" optimality, which makes use of a "nominal" parameter θ_0 that is assumed to be close to the true value (see, e.g., Chernoff 1953 or Chapter 17 in Atkinson et al. 2007 and Chapter 5 in Pronzato and Pázman 2013). If, for each $\mathbf{x} \in \mathfrak{X}$, the observation $y(\mathbf{x})$ is normally distributed with a possibly unknown constant variance σ^2 and $\eta(\mathbf{x}, \cdot)$ is differentiable in θ_0, then the approach of local optimality leads to

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}, \theta_0) = \left. \frac{\partial \eta(\mathbf{x}, \theta)}{\partial \theta} \right|_{\theta = \theta_0}$$

and

$$\mathbf{M}(\xi) = \mathbf{M}(\xi, \theta_0) = \sum_{\mathbf{x} \in \mathfrak{X}} \xi(\mathbf{x}) \mathbf{f}(\mathbf{x}, \theta_0) \mathbf{f}^T(\mathbf{x}, \theta_0).$$

[1] This holds if each trial consumes a constant amount of resources, independent of any other trials. The situation is fundamentally more complex under general restrictions on permissible experimental designs; see, e.g., Harman et al. (2016), Sagnol and Harman (2015), and Filová and Harman (2020).

Table 1 Selected GLM classes and the corresponding functions $w(\mathbf{x}, \theta_0)$. The symbol f denotes the standard normal density, and F denotes the standard normal cumulative distribution function

GLM class (distribution)	Link $g(\eta)$	Function $w(\mathbf{x}, \theta_0)$
Logistic (Bernoulli)	$\log \frac{\eta}{1-\eta}$	$\frac{e^{\mathbf{h}^T(\mathbf{x})\theta_0}}{(1+e^{\mathbf{h}^T(\mathbf{x})\theta_0})^2}$
Probit (Bernoulli)	$F^{-1}(\eta)$	$\frac{f^2(\mathbf{h}^T(\mathbf{x})\theta_0)}{F(\mathbf{h}^T(\mathbf{x})\theta_0)(1-F(\mathbf{h}^T(\mathbf{x})\theta_0))}$
Poisson (Poisson)	$\log \eta$	$e^{\mathbf{h}^T(\mathbf{x})\theta_0}$

For a generalized linear model (GLM) for which the mean value of the observations is $\eta(\mathbf{x}, \theta) = g^{-1}(\mathbf{h}^T(\mathbf{x})\theta)$, where $g : \mathbb{R} \to \mathbb{R}$ is a strictly monotonic and differentiable link function and $\mathbf{h} : \mathfrak{X} \to \mathbb{R}^m$ is known, we have

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}, \theta_0) = \sqrt{w(\mathbf{x}, \theta_0)}\mathbf{h}(\mathbf{x})$$

for some $w : \mathfrak{X} \times \mathbb{R}^m \to (0, \infty)$. In Table 1, we provide the form of w for several common classes of GLMs (for more details on computing optimal designs in GLMs, see, e.g., Khuri et al. 2006; Atkinson and Woods 2015). Since in the approach of local optimality, the nominal parameter θ_0 is treated as a constant, we henceforth write $\mathbf{f}(\mathbf{x})$ and $\mathbf{M}(\xi)$ instead of $\mathbf{f}(\mathbf{x}, \theta_0)$ and $\mathbf{M}(\xi, \theta_0)$, respectively, for both linear and nonlinear models.

The size of an information matrix (and, implicitly, the quality of the corresponding design) is measured in terms of an optimality criterion Φ. For clarity, we will focus on the most common criterion of D-optimality, but the method proposed in this paper can be trivially adapted to a large class of criteria (cf. Sect. 4). The D-optimal design problem is to find a design ξ^* that maximizes

$$\Phi(\mathbf{M}(\xi)) = \{\det(\mathbf{M}(\xi))\}^{1/m}$$

over the set Ξ of all designs. The D-optimal design minimizes the volume of the confidence ellipsoid for the vector of unknown parameters.[2] The criterion Φ is concave, positive homogeneous and Loewner isotonic on the set of all $m \times m$ nonnegative definite matrices. Note also that the set Ξ is convex and compact, that is, the problem of D-optimal design is convex[3] and always has at least one optimal solution. The information matrix of all D-optimal designs is nonsingular and unique, although for some models, even some that appear in practice, the D-optimal design itself is not unique.

The D-efficiency of a design ξ relative to a design ζ with $\Phi(\mathbf{M}(\zeta)) > 0$ is defined as $\text{eff}(\xi|\zeta) = \Phi(\mathbf{M}(\xi))/\Phi(\mathbf{M}(\zeta))$. The D-efficiency of a design ξ (per se) means the D-efficiency of ξ relative to the D-optimal design ξ^*. The D-efficiency of ξ with nonsingular $\mathbf{M}(\xi)$ satisfies (see, e.g., Pukelsheim 2006, Section 5.15)

$$\text{eff}(\xi|\xi^*) \geq \frac{m}{\max_{\mathbf{x}\in\mathfrak{X}} d_\xi(\mathbf{x})}, \tag{2}$$

where $d_\xi(\mathbf{x}) = \mathbf{f}^T(\mathbf{x})\mathbf{M}^{-1}(\xi)\mathbf{f}(\mathbf{x})$. The function $d_\xi(\cdot)$ is called the "variance function" because it is proportional to the variance of the least squares estimator of $\mathbf{f}^T(\cdot)\theta$ in the linear regression model. Equation (2) implies the so-called equivalence theorem for D-optimality: A design ξ is D-optimal if and only if $\max_{\mathbf{x}\in\mathfrak{X}} d_\xi(\mathbf{x}) = m$; otherwise, $\max_{\mathbf{x}\in\mathfrak{X}} d_\xi(\mathbf{x}) > m$ (see Kiefer 1959). Therefore, provided that the maximum of $d_\xi(\cdot)$ over $\mathfrak{X}$ can be reliably determined, it is possible to construct a lower bound on the D-efficiency of ξ or prove its optimality. However, if $\mathfrak{X}$ is large and multi-dimensional, maximization can be challenging because $d_\xi(\cdot)$ is typically nonlinear and multimodal, even for linear regression models.

In the rest of this paper, we will drop the prefix "D-", e.g., a D-optimal design will simply be called an "optimal design".

We are not aware of any method that specifically targets optimal design problems on very large discrete grids. For the case of factors that are discrete with an enormous number of practically possible levels, the usual approach is to consider the factors to be fully continuous. Therefore, methods that operate on continuous spaces are natural competitors for the method proposed in this paper, and we briefly discuss them in the rest of this section.

If one or more factors are assumed to be continuous, the simplest approach is to discretize those factors and turn the problem into one that can be solved using discrete-space methods. Naturally, discrete-space solvers then generally cannot find a design that is perfectly optimal on the original, continuous $\mathfrak{X}$. Nonetheless, our experience shows that the loss in efficiency is usually negligible if a continuous factor is discretized to, say, 1000 levels. Such discretization can be handled with modern hardware and state-of-the-art discrete-space methods for problems with only one or two continuous factors or problems with one continuous factor and a few binary factors.

There are also other situations with continuous factors in which it is possible to directly apply the existing discrete-space methods. For instance, for some models, there exist theoretical results that limit the support of an optimal design to a relatively small finite subset. As an example, for a fully quadratic linear model on a cube, the search can be restricted to the set of centers of the faces of the cube of all dimensions (see Heiligers 1992).

[2] For nonlinear models, this statement is valid asymptotically.

[3] In the sense that the problem consists of maximizing a concave objective function over a convex set. Note, however, that if some of the factors are continuous, then the D-optimal design problem, viewed as a problem of convex optimization, is infinite-dimensional.

However, direct discretization may not be sufficient if there are more than two continuous factors or if we require a fine resolution of the levels. In such cases, choosing a method such as the one proposed in this paper can be of significant benefit.

For computing optimal designs when all or some of the factors are continuous, there are many methods that do not employ a fixed discretization of $\mathfrak{X}$. To present a brief survey thereof, we choose to split them into two categories.

1.1 Methods that use standard continuous-space algorithms

The problem of optimal design with continuous factors can be directly solved by means of common nonlinear programming methods; for example, the Nelder–Mead algorithm is used in Chaloner and Larntz (1989), a quasi-Newton method is applied in Atkinson et al. (2007) (see Section 13.4), semi-infinite programming is used in Duarte and Wong (2014), and multiple nonlinear programming methods are applied in García-Ródenas et al. (2020). Some papers utilize nonlinear programming methods in a specific way or for a particular class of models; see, for instance, Gribik and Kortanek (1977) and Papp (2012).

In addition to classical nonlinear programming methods, various metaheuristics have been recently proposed to compute optimal designs with continuous factors; see, e.g., Wong et al. (2015) for particle swarm optimization, Xu et al. (2019) or Stokes et al. (2020) for differential evolution, Lukemire et al. (2019) for quantum-behaved particle swarm optimization, and Zhang et al. (2020) for competitive swarm optimization. A thorough empirical evaluation of the performance of nonlinear programming methods and metaheuristics, including genetic algorithms, is given by García-Ródenas et al. (2020).

A typical feature of nonlinear programming methods and metaheuristics is that a fixed-size support of the design is merged with a vector of weights to form a single feasible solution to the optimization problem at hand. This transforms an infinite-dimensional convex problem into a finite one, which is, however, not convex in the coordinates of the support points. Then, these algorithms simultaneously adjust the support and the corresponding weights. A major advantage is that this approach can be used with almost any criterion; it is necessary only to implement the evaluation of the criterion as a subroutine. A disadvantage is that these methods either require a search for an appropriate support size or are rendered less efficient by using a relatively large support size as provided by the Carathéodory theorem (see, e.g., Theorem 2.2.3 in Fedorov 1972). In addition, because the problem is nonconvex, convergence to a good design is usually not guaranteed, and even when convergence is reached, it can be slow. One reason for the slow speed is that with the direct application of nonlinear programming and metaheuristics, the convexity of the problem in the design weights is not exploited.

1.2 Methods that use a discrete-space solver on "adaptive grids"

An alternative idea is to use a discrete-space method applied to a finite set, which is sequentially updated in the continuous space to (hopefully) approach the support of the optimal design. We call this strategy "adaptive support". The general idea of an adaptive search for the optimal support has been present in the literature from the beginning; for instance, the classical Fedorov-Wynn algorithm is based on iterative improvements to the support of the design, although for each new support, the design weights are adjusted only by means of a simple transformation (Fedorov 1972, Chapter 4). The general possibility of using adaptive supports has also been mentioned in Wu (1978a) and Wu (1978b). More recently, a randomized adaptive support approach has been suggested by Dror and Steinberg (2006).[4] Furthermore, Stufken and Yang (2012) (Section 6) have proposed "...a multi-stage grid search that starts with a coarse grid that is made increasingly finer in later stages". Adaptation of the support is also central to the computational methods of Yang et al. (2013) and Pronzato and Zhigljavsky (2014). These methods apply classical convex optimization to find optimal weights on a fixed discrete set, and in each step, they enrich the support set with the global maximum of the variance function. Most recently, Duarte et al. (2018) have employed semidefinite programming and an adaptive replacement of the support with a specific set of local maximizers of the variance function.

Our proposed algorithm, GEX, is more closely related to the class of continuous-space methods reviewed in Sect. 1.2 than to those discussed in Sect. 1.1.

2 The algorithm

In GEX, we apply a discrete-space optimal design algorithm to suitably chosen exploration sets $\mathfrak{X}_{exp}$. In the course of the computation, the exploration sets are sequentially updated with the aim of approaching the support of the true optimal design for the model at hand. An outline of GEX is presented as Algorithm 1 below. The input to the algorithm includes the problem itself in the form of a subroutine that, for each permissible $\mathbf{x} \in \mathfrak{X}$, computes the vector $\mathbf{f}(\mathbf{x})$. The values $\text{eff}_{opt} < 1$, $\text{eff}_{grp} \leq 1$, $N_{loc} \in \{0, 1, \ldots\}$ and $\text{eff}_{stop} < 1$ are

[4] Dror and Steinberg (2006) consider support adaptation for optimal *exact* designs, but the idea can also be applied to optimal *approximate* designs, which are central to this paper.

parameters chosen by the user; their meaning is explained later in this section.

Algorithm 1: GEX (outline)

input: $\mathfrak{X}$, $\mathbf{f}$, eff_{opt}, eff_{grp}, N_{loc}, eff_{stop}
1 $\mathfrak{X}_{exp} \leftarrow$ INI ($\mathfrak{X}$)
2 $\xi_{new} \leftarrow$ OPT ($\mathfrak{X}_{exp}$, $\mathbf{f}$, eff_{opt}, eff_{grp})
3 **repeat**
3a $\quad \xi_{old} \leftarrow \xi_{new}$
3b $\quad \mathfrak{X}_{exp} \leftarrow$ EXP ($\mathfrak{X}$, $\mathbf{f}$, ξ_{old}, N_{loc})
3c $\quad \xi_{new} \leftarrow$ OPT ($\mathfrak{X}_{exp}$, $\mathbf{f}$, eff_{opt}, eff_{grp})
until $\phi(\xi_{old})/\phi(\xi_{new}) > \text{eff}_{stop}$
4 **return** ξ_{new}

In Steps 1 and 2, we select an initial finite exploration set $\mathfrak{X}_{exp} \subseteq \mathfrak{X}$ and use a discrete-space method to calculate an efficient design ξ_{new} on $\mathfrak{X}_{exp}$. In Step 3, we alternately construct a finite exploration set $\mathfrak{X}_{exp}$ based on the current design and compute a new efficient design on $\mathfrak{X}_{exp}$. The algorithm stops once the last optimization on $\mathfrak{X}_{exp}$ does not lead to a significant improvement. Note that this form of the stopping rule implies that the number of loops in Step 3 is bounded.

Although the general scheme of GEX is simple, judicious specification of the basic steps can make a crucial difference in the overall performance. In the following subsections, we detail our choices regarding the procedures INI, OPT and EXP.

2.1 INI

In Step 1 of Algorithm 1, we construct an initial exploration set $\mathfrak{X}_{exp} \subseteq \mathfrak{X}$. In our specification, $\mathfrak{X}_{exp}$ is constructed as the union of two sets, $\mathfrak{X}_{grid}$ and $\mathfrak{X}_{rnd}$. The set $\mathfrak{X}_{grid}$ is a grid formed by the combinations of the extreme levels of all factors and the median levels of all nonbinary factors. The set $\mathfrak{X}_{rnd}$ is a random selection of points in $\mathfrak{X}$. The size of $\mathfrak{X}_{grid}$ is at most 3^k, which is reasonably small for as many as $k \lesssim 13$ factors, and for many models used in practice, $\mathfrak{X}_{grid}$ constructed in the proposed way contains highly informative points. For our numerical study, we chose the size of $\mathfrak{X}_{rnd}$ to be 1000; a sensitivity analysis suggests that the performance of the algorithm is similar for a wide range of sizes of $\mathfrak{X}_{rnd}$.

2.2 OPT

In Steps 2 and 3c of Algorithm 1, we apply a discrete-space algorithm that computes a design ξ_{new} on $\mathfrak{X}_{exp}$ with an efficiency of at least eff_{opt} (among all designs on $\mathfrak{X}_{exp}$) and then prunes the support of ξ_{new} by means of a grouping procedure parametrized by the value of eff_{grp}. In more detail, the assignment process $\xi_{new} \leftarrow$ OPT ($\mathfrak{X}_{exp}$, $\mathbf{f}$, eff_{opt}, eff_{grp}) consists of the steps shown below.

begin OPT ($\mathfrak{X}_{exp}$, $\mathbf{f}$, eff_{opt}, eff_{grp})
$\quad \xi_{new} \leftarrow$ REX ($\mathfrak{X}_{exp}$, $\mathbf{f}$, eff_{opt})
$\quad \xi_{new} \leftarrow$ GRP (ξ_{new}, $\mathbf{f}$, eff_{grp})
end

As the underlying discrete-space optimization procedure, we choose the REX algorithm of Harman et al. (2020). It has several advantages compared to other methods: REX is not only fast and applicable to relatively large problems, but crucially, the resulting designs do not have a tendency to contain many design points with small "residual" weights. The parameter eff_{opt} is the lower bound on the efficiency that is used to stop the computation. In our numerical studies, we chose $\text{eff}_{opt} = 1 - 10^{-6}$.

In Step 2 of Algorithm 1, the initial design for REX is based on the modified Kumar-Yildirim method as described in Harman and Rosa (2020), which is rapid and always provides a design with a nonsingular information matrix. If this relatively advanced initialization method were to be replaced by the uniform random selection of m distinct points, the efficiency of the resulting design (and the speed of computation) would not be much affected. However, the Kumar-Yildirim method guarantees nonsingularity of the information matrix of the initial design, which is an important aspect of the algorithm.[5] In contrast to Step 2, in Step 3c, the REX algorithm is initialized with ξ_{old}.

Each REX computation is followed by a specific form of grouping of nearby support points. For continuous spaces, a grouping method has previously been recommended by Fedorov (1972), page 109. In our case, the factors are discrete, but the levels can be very close to each other; consequently, grouping nearby support points generally improves the performance. We have identified that for all studied models, it is sufficient to use a nearest-distance approach to decrease the support size of the constructed design on a discrete space.

More precisely, let ξ_0 be the design obtained via the original discrete-space procedure, such as REX. Let $\mathbf{x}_1, \ldots, \mathbf{x}_s$ be the support points of ξ_0.[6] The procedure GRP determines the pair $(\mathbf{x}_k, \mathbf{x}_l)$, $k < l$, consisting of the two nearest support points and assigns the pooled weight $\xi_0(\mathbf{x}_k) + \xi_0(\mathbf{x}_l)$ to the point $\mathbf{x}_k$ if $\xi_0(\mathbf{x}_k) \geq \xi_0(\mathbf{x}_l)$ or to the point $\mathbf{x}_l$ if

[5] In many models with a finite number of factor levels, there is a nonzero probability that a a design with a randomly selected m-point support will have a singular information matrix, even if the model itself admits a design with a nonsingular information matrix.

[6] Note that using REX as the engine provides designs with relatively small support sizes, almost always smaller than m^2.

$\xi_0(\mathbf{x}_k) < \xi_0(\mathbf{x}_l)$, which results in a design ξ_1. If the efficiency of ξ_1 is at least eff_{grp} relative to the design originally returned by REX, then the pooling operation is accepted, and the process is repeated. In our study, we chose $\text{eff}_{grp} = 1 - 10^{-6}$. We remark that removing GRP (e.g., by setting $\text{eff}_{grp} = 1$) results in marginally more efficient designs, but it makes the computation slower. More importantly, not using GRP results in designs that are populated by a large number of points with small weights.

2.3 EXP

In the key Step 3b, we construct a new exploration set $\mathfrak{X}_{exp}$ based on a set of local maxima of the variance function as well as a long-range variation of the support of the current design. In detail, $\mathfrak{X}_{exp} \leftarrow \texttt{EXP}\ (\mathfrak{X}, \mathbf{f}, \xi_{old}, N_{loc})$ consists of the steps summarized below.

begin EXP $(\mathfrak{X}, \mathbf{f}, \xi_{old}, N_{loc})$
  $\mathfrak{X}_{loc} \leftarrow$ LOC $(\mathfrak{X}, \mathbf{f}, \xi_{old}, \mathrm{N}_{loc})$
  $\mathfrak{X}_{star} \leftarrow$ STAR $(\mathfrak{X}, \xi_{old})$
  $\mathfrak{X}_{exp} \leftarrow \mathfrak{X}_{loc} \cup \mathfrak{X}_{star}$
end

The procedure LOC returns the results of randomly initialized hill climbing maximizations of the current variance function $d\xi_{old}(\cdot)$ over $\mathfrak{X}$. The number of maximizations starting from random points in $\mathfrak{X}$ is given by the parameter N_{loc}; for our numerical study, we chose $\mathrm{N}_{loc} = 50$. Decreasing N_{loc} slightly worsens the final design but also makes the computation faster. Removing the $\mathfrak{X}_{loc}$ part of $\mathfrak{X}_{exp}$ altogether often makes the computation much faster, but for some models, it systematically leads to suboptimal designs.

As in all local search methods, the efficiency depends on the system of neighborhoods considered for each feasible solution $\mathbf{x} \in \mathfrak{X}$.

We use a "star-set" neighborhood $\mathfrak{S}(\mathbf{x})$, which consists of all points lying in a "star" centered at $\mathbf{x}$: $\mathfrak{S}(\mathbf{x})$ is the set of points in $\mathfrak{X}$ that differ from $\mathbf{x}$ in at most one coordinate. Formally,

$$\mathfrak{S}(\mathbf{x}) = \big\{\, \tilde{\mathbf{x}} \in \mathfrak{X} \mid \text{ There exists an } i \in \{1, \ldots, k\} \text{ such that } \tilde{x}_i \in \mathfrak{X}_i \text{ and } \tilde{x}_j = x_j \text{ for all } j \neq i \,\big\}.$$

We search the neighborhood of the current point $\mathbf{x}$, move to the point in $\mathfrak{S}(\mathbf{x})$ with the highest value of the variance function, and repeat this process as long as there is an increase in the variance function. Neighborhoods of the form of $\mathfrak{S}(\mathbf{x})$ seem to be suitable for optimal design problems on multidimensional grids, as they allow us to explore such grids quite thoroughly yet do not grow exponentially in size with respect to the dimensionality k of the design space. Moreover, the neighborhoods $\mathfrak{S}(\mathbf{x})$ have the added advantage that they are fast to computationally enumerate.

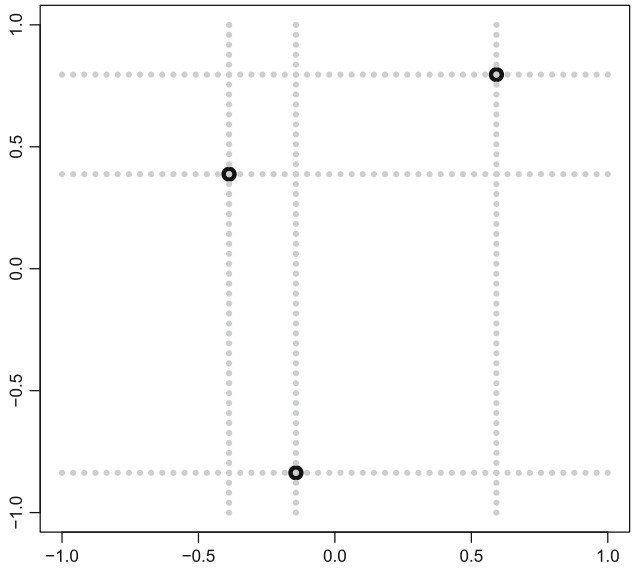

Fig. 1 $\mathfrak{X}_{star}$ for a design with 3 support points, where $\mathfrak{X}$ is a discretized square $[-1, 1]^2$. The support points are denoted by black circles, and the gray dots form the star sets

The crucial procedure for the construction of the exploration sets is STAR, which is specifically chosen to suit the grid-like structure of the design space studied in this paper. The best solution requires a balanced compromise between exploration and exploitation. Our experience leads us to choose star sets *again* (hence the name "galaxy exploration"). For a given ξ_{old}, the set $\mathfrak{X}_{star}$ is the union of all star-set neighborhoods centered at the support points of ξ_{old}. Formally,

$$\mathfrak{X}_{star} = \bigcup_{\mathbf{x} \in \text{supp}(\xi_{old})} \mathfrak{S}(\mathbf{x});$$

such a star set is illustrated in Fig. 1. Note that the REX algorithm produces efficient designs with "sparse" supports, which means that the size of $\mathfrak{X}_{star}$ does not grow greatly during the GEX computation.

2.4 Summary

Algorithm 2 shows the final formulation of GEX, with all steps described in detail.

Similar to the methods of Sect. 1.2, in particular Pronzato and Zhigljavsky (2014) or Duarte et al. (2018), our method applies a discrete-space solver to a subset of $\mathfrak{X}$ and adaptively modifies this subset. However, unlike the method of Pronzato and Zhigljavsky (2014) and many others, GEX does not attempt to calculate the global maximum of the variance

Algorithm 2: GEX (detailed)

input: $\mathfrak{X}$, $\mathbf{f}$, eff_{opt}, eff_{grp}, N_{loc}, eff_{stop}
1 $\mathfrak{X}_{exp} \leftarrow$ INI $(\mathfrak{X})$
2 **begin** OPT
  $\xi_{new} \leftarrow$ REX $(\mathfrak{X}_{exp}, \mathbf{f}, \text{eff}_{opt})$
  $\xi_{new} \leftarrow$ GRP $(\xi_{new}, \mathbf{f}, \text{eff}_{grp})$
  end
3 **repeat**
3a $\xi_{old} \leftarrow \xi_{new}$
3b **begin** EXP
    $\mathfrak{X}_{loc} \leftarrow$ LOC $(\mathfrak{X}, \mathbf{f}, \xi_{old}, \text{N}_{loc})$
    $\mathfrak{X}_{star} \leftarrow$ STAR $(\mathfrak{X}, \xi_{old})$
    $\mathfrak{X}_{exp} \leftarrow \mathfrak{X}_{loc} \cup \mathfrak{X}_{star}$
  end
3c **begin** OPT
    $\xi_{new} \leftarrow$ REX $(\mathfrak{X}_{exp}, \mathbf{f}, \text{eff}_{opt})$
    $\xi_{new} \leftarrow$ GRP $(\xi_{new}, \mathbf{f}, \text{eff}_{grp})$
  end
  until $\phi(\xi_{old})/\phi(\xi_{new}) > \text{eff}_{stop}$
4 **return** ξ_{new}

function but instead utilizes a set of local maximizers. The general idea of using a set of local maximizers is similar to that of Duarte et al. (2018), but in GEX, we accomplish it in a significantly different way. Importantly, we include the star sets around the support points of the current ξ, meaning that the exploration set is much larger than merely a set of local maximizers of the variance function, which enables GEX to better explore the design space. Note also that the $\mathfrak{X}_{star}$ part of the exploration set is similar to the set implicitly utilized in the coordinate-exchange method for exact designs (see Meyer and Nachtsheim 1995), where in each iteration, one coordinate of one design point is updated via a line search. Nonetheless, we compute approximate, not exact, designs and, crucially, do not consider the coordinates of the points one at a time; rather, all single-coordinate changes to all points are considered simultaneously as a batch (with a set of local maximizers of the variance function), to which an efficient internal discrete-space solver is applied.

In summary, GEX is practically and numerically simple to apply in the sense that it requires no theoretical analysis of the model at hand and no special numerical solvers or libraries, except for the standard procedures of linear algebra. The algorithm has only a few tunable parameters, which are directly interpretable and can be set to fixed default values that perform very well for a wide class of models (as we will demonstrate in Sect. 3).

In each iteration, GEX finds a design that is at least as good as the previous one, and is guaranteed to stop in a finite number of iterations. Nevertheless, GEX does not necessarily converge to the theoretical optimum, even if we let the terminating parameter eff_{stop} approach 1; it is possible to construct artificial problems for which it produces considerably suboptimal designs with high probability. Moreover, for extremely large design spaces $\mathfrak{X}$, it is difficult to verify that the design provided by GEX is optimal or, more generally, that it has an efficiency above a certain threshold. However, this difficulty is not specific to the proposed method; it is rather a feature of the optimization problem that we are attempting to solve. Obtaining a perfectly reliable lower bound on the efficiency of a design would require either computationally processing all vectors $\mathbf{f}(\mathbf{x})$, $\mathbf{x} \in \mathfrak{X}$ (cf. inequality (2)), or utilizing mathematical properties specific to the model, which might demand an exceedingly large amount of effort and/or expert knowledge.

Finally, we note that GEX automatically determines the support size of the resulting design. This property is beneficial in the sense that the application of GEX does not require a search for a proper support size. However, this also means that we do not have control over the support size.[7] If we require efficient designs with a strict upper bound on the support size, nonlinear programming techniques or heuristics (cf. Sect. 1.1) may be a better choice than GEX.

2.5 Notes

In some cases, the information matrix of the D-optimal design is ill-conditioned. If this occurs, we can use the well-known fact that regular reparametrization, i.e., replacing the vectors $\mathbf{f}(\mathbf{x})$ with $\tilde{\mathbf{f}}(\mathbf{x}) = \mathbf{R}\mathbf{f}(\mathbf{x})$, where $\mathbf{R}$ is nonsingular, does not alter the D-optimal design. If $\mathbf{R}$ is chosen to be close to the inverse of the square root of the optimal information matrix, such reparametrization may greatly improve the condition number of the matrices used in GEX. In particular, we applied this approach to enhance the numerical stability of the computation of optimal designs for Model 10 described in Sect. 3. We chose $\mathbf{R} = \mathbf{M}^{-1/2}$, where $\mathbf{M}$ is the information matrix of the design computed for the initial $\mathfrak{X}_{exp}$ via the modified Kumar-Yildirim method.

During the execution of the algorithm, points that cannot support optimal designs can be removed by using the results of Harman and Pronzato (2007) in order to decrease the size of the problem and potentially speed up the computations. However, in our experience, the removal of redundant points does not provide a significant increase in the speed of GEX. This is likely due to the nature of the discrete-space engine REX, which tends to work with designs with small supports and, as such, does not benefit much from the elimination of unpromising parts of the design space.

[7] Nevertheless, the support sizes determined by GEX for the benchmark models studied in Sect. 3 are reasonably small; see the column labeled s_{GEX} in Table 4.

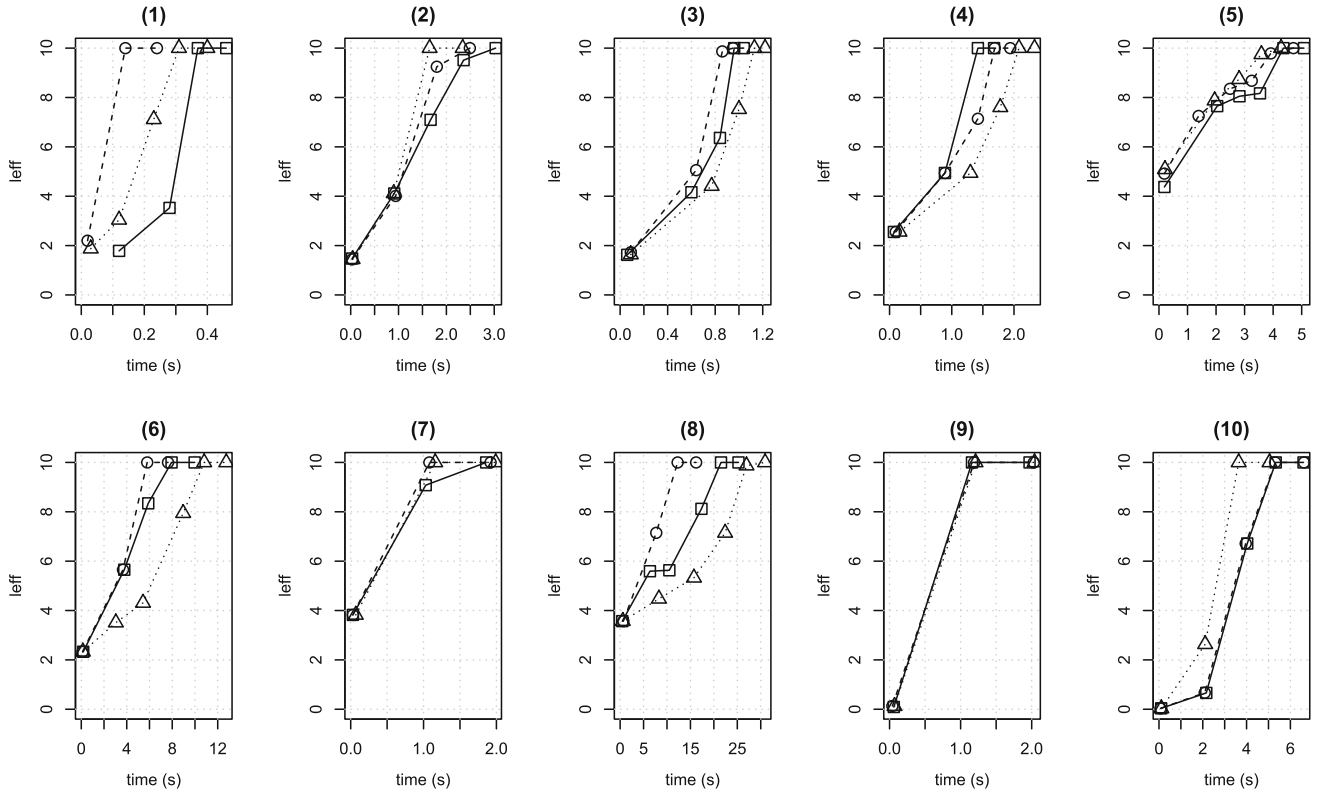

Fig. 2 The time profiles of the efficiencies of the designs ξ_{new} computed by REX in Steps 2 and 3c of GEX. For each problem (see the numbers above the panels), we executed 3 independent runs of GEX; each run is represented by a separate piecewise-linear curve. The efficiencies are expressed on a logarithmic scale, with efficiencies of 0, 0.9, 0.99, 0.999, ..., $1 - 10^{-9}$, and $> 1 - 10^{-9}$ corresponding to the values 0, 1, 2, 3, ..., 9, and 10, respectively, on the vertical axis

3 A numerical study

For a numerical study, we selected 5 prominently published recent papers that focus on presenting algorithms for computing D-optimal designs on continuous or mixed design spaces, namely,

- Pronzato and Zhigljavsky (2014) (adaptive grids combined with projected gradients and vertex exchange optimization),
- Duarte et al. (2018) (a different approach to adaptive grids combined with semidefinite programming),
- Lukemire et al. (2019) (quantum-behaved particle swarm optimization),
- Xu et al. (2019) (differential evolution), and
- García-Ródenas et al. (2020) (multiple nonlinear programming and metaheuristic methods).

From each paper, we chose two test problems, as summarized in Table 3. The chosen problems include a linear regression model, two nonlinear models with homoscedastic normal errors, a Poisson model, a probit model, and several logistic regression models. The number of factors, k, varies from 2 to 10, and the number of parametric dimensions, m, varies from 4 to 16. We used these test problems to illustrate the behavior of GEX and numerically compare its performance with that of the competing methods.

Figure 2 shows the typical behavior of GEX on the problems in Table 3. We see that the designs ξ_{new} generated in the main loop of GEX monotonically improve and ultimately converge to the optimum.[8] The variability of the computation time is moderate.[9] The usual number of discrete-space optimizations before the stopping rule is satisfied is 3 to 6. We observe that the efficiency growth generally does not decline with increasing computation time. Moreover, with increasing problem difficulty, the number of discrete-space optimization runs required to achieve the optimal design does not tend to grow.

Some additional general observations from the computational study of GEX can be summarized as follows:

[8] By the optimum, we mean the best design that we are aware of, either from our numerical studies or from the literature.

[9] The coefficient of variation of the computation times is less than 0.3 for all studied problems.

Table 2 The design produced by GEX for Model 6. The first column gives the index i of the identified support point, the columns labeled x_{i1} to x_{i5} list the values of the factors, and $\xi(\mathbf{x}_i)$ is the weight of the corresponding support point $\mathbf{x}_i$

i	x_{i1}	x_{i2}	x_{i3}	x_{i4}	x_{i5}	$\xi(\mathbf{x}_i)$
1	−2.000	−2	−2	−2.000	−2	0.093630
2	−2.000	2	2	−2.000	−2	0.092334
3	2.000	−2	−2	2.000	−2	0.062242
4	2.000	2	−2	2.000	−2	0.075893
5	2.000	2	2	2.000	−2	0.065934
6	−2.000	−2	2	−2.000	2	0.076828
7	−2.000	2	−2	2.000	2	0.089598
8	−0.930	2	−2	−2.000	−2	0.080512
9	−1.739	−2	−2	2.000	2	0.060395
10	−2.000	−2	−2	−1.340	2	0.024540
11	−0.528	2	−2	2.000	2	0.004427
12	−1.636	2	2	2.000	2	0.054245
13	2.000	−2	2	1.838	−2	0.055518
14	1.793	−2	2	2.000	2	0.083511
15	−2.000	2	2	−1.483	2	0.034830
16	1.391	−2	2	−2.000	−2	0.045563

- Depending on the test problem from Table 3, the runs of the subroutine `REX` consume between 10 and 60 percent, and the runs of the subroutine `LOC` consume between 15 and 85 percent of the overall computation time. Other subroutines take negligible amounts of time. Consequently, the speed of GEX is mostly determined by the speed of the underlying discrete-space optimization and the procedure that computes the sets of local optima of the variance functions.
- The results of GEX are highly robust with respect to the choice of the initial exploration set. Even if $\mathfrak{X}_{exp}$ constructed by the subroutine `INI` only consists of 1000 random points of $\mathfrak{X}$ (that is, if $\mathfrak{X}_{grid}$ is not used), there is only a small variation in the resulting designs. In particular, for each model from Table 3, the fully random initial exploration set leads, with a large probability, to the same size of the support and the same value of the criterion (within 6 significant digits); cf. Table 4.

We provide R (R Development Core Team 2020) codes that are completely free to use and require minimal technical expertise to be applied to any D-optimal design problem on a cuboid grid; see www.iam.fmph.uniba.sk/ospm/Harman/design/.

For illustration, in Table 2, we present the detailed design for Model 6 as computed by GEX. All other designs can be obtained within a few seconds by running the provided R script. Note that due to the nature of GEX, the resulting designs are "tidy", i.e., there is no need to remove design points with very small weights or round the positions of the support points to a reasonable number of decimal places.

In optimal experimental design, the numerical comparison of competing computational methods is a challenging task. The reason is that there are no generally adopted benchmark suites and no guidelines for reporting the results of such comparisons. Often, the source code is not available or is fine-tuned to the few problems studied in the source paper. Moreover, the quality of the resulting designs depends on the computation time in a method-specific way and can be strongly influenced by the choice of hardware, programming language and implementation details; a seemingly minor change can lead to a several-fold speed-up or slow-down. Despite these methodological difficulties, however, at least a brief comparison is necessary because there are a plethora of heuristic or theoretical ideas applicable to the computation of optimal designs, and failing to realize that a proposed method is much worse than existing alternatives can be detrimental to readers.

In Table 4, we provide the computation times, support sizes and criterion values for the optimal or nearly optimal designs as reported in the corresponding papers[10] and as provided by an R implementation of GEX. We used Microsoft R Open 3.5.3 and a 64-bit Windows 10 system with an Intel Core i7-9750H processor operating at 2.60 GHz.

The authors of the competing methods used either MATLAB or C++, both of which almost always permit a faster implementation of a given algorithm than R does. Except for Pronzato and Zhigljavsky (2014), the cited authors used standard modern hardware.

The benchmark results for Models 2 and 3 were obtained by running the algorithm of Pronzato and Zhigljavsky (2014) 11 times using our hardware and the MATLAB code kindly provided by the authors. For these two models, the authors also provide analytically calculated optimal designs based on the results of Schwabe (2012); thus, it can be shown that the designs obtained by GEX for these models have efficiencies of at least 99.999% relative to the provably optimal designs. For Models 1 and 4, the CPU times and benchmark design were taken from Table 9 in Duarte et al. (2018). The results for Models 5 and 9 from Lukemire et al. (2019) were obtained by running the compiled instance of the program written in C++ provided by the authors, with the number of iterations set to 2000 and the required number of support points predetermined by GEX. We ran each instance 11 times and recorded the computation times and criterion values for the resulting designs. The benchmark designs for Models 8 and 10 can be found in Tables 5 and 11 of Xu et al. (2019) and the computation times are based on personal communication with the lead author of Xu et al. (2019). Finally, the results for

[10] In selected cases, we used our own hardware to recompute the designs; see the following text.

Table 3 Descriptions of the benchmark models, the nominal parameter θ_0 used and the design space $\mathfrak{X}$

#	Model	θ_0	Design space $\mathfrak{X}$	k	m
1	$y(\mathbf{x}) \sim N(\eta, \sigma^2)$, $\eta = \frac{1}{1+e^{\mathbf{h}^T(\mathbf{x})\theta}}$, $\mathbf{h}^T(\mathbf{x}) = (1, x_1, x_2, x_1x_2)$	$(-2, 0.5, 0.5, 0.1)^T$	$[0, 5]_{0.001} \times [0, 1]_{0.001}$	2	4
2	$y(\mathbf{x}) \sim N(\eta, \sigma^2)$, $\eta = \theta_1 + \theta_2 e^{-\theta_3 x_1} + \frac{\theta_4}{\theta_4 - \theta_5}(e^{-\theta_5 x_2} - e^{-\theta_4 x_2})$	$(1, 1, 2, 0.7, 0.2)^T$	$[0, 2]_{0.001} \times [0, 10]_{0.001}$	2	5
3	$y(\mathbf{x}) \sim N(\eta, \sigma^2)$, $\eta = \mathbf{h}^T(\mathbf{x})\theta$, $\mathbf{h}^T(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3)$	does not affect the design	$[-1, 1]^2_{0.001}$	2	7
4	$y(\mathbf{x}) \sim Pois(\eta)$, $\eta = e^{\mathbf{h}^T(\mathbf{x})\theta}$, $\mathbf{h}^T(\mathbf{x}) = (1, \mathbf{x}^T, x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3)$	$(0.5, -0.2, 0.5, -0.2, -0.1, 0.2, -0.1, 0.2, -0.1, 0.2)^T$	$[-1, 1]^3_{0.001}$	3	10
5	$y(\mathbf{x}) \sim Bin(1, \eta)$, $\eta = \frac{e^{\mathbf{h}^T(\mathbf{x})\theta}}{1+e^{\mathbf{h}^T(\mathbf{x})\theta}}$, $\mathbf{h}^T(\mathbf{x}) = (1, \mathbf{x}^T)$	$(-1, 2, 0.5, -1, -0.25, 0.13)^T$	$\{-1, 1\}^4 \times [5, 35]_{0.001}$	5	6
6	$y(\mathbf{x}) \sim Bin(1, \eta)$, $\eta = F(\mathbf{h}^T(\mathbf{x})\theta)$, $\mathbf{h}^T(\mathbf{x}) = (1, \mathbf{x}^T)$	$(0.5, 0.7, 0.18, -0.2, -0.58, 0.51)^T$	$[-2, 2]^5_{0.001}$	5	6
7	$y(\mathbf{x}) \sim Bin(1, \eta)$, $\eta = \frac{e^{\mathbf{h}^T(\mathbf{x})\theta}}{1+e^{\mathbf{h}^T(\mathbf{x})\theta}}$, $\mathbf{h}^T(\mathbf{x}) = (1, \mathbf{x}^T)$	$(0.5, 0.7, 0.18, -0.2, -0.58, 0.51)^T$	$[-2, 2]^5_{0.001}$	5	6
8	$y(\mathbf{x}) \sim Bin(1, \eta)$, $\eta = \frac{e^{\mathbf{h}^T(\mathbf{x})\theta}}{1+e^{\mathbf{h}^T(\mathbf{x})\theta}}$, $\mathbf{h}^T(\mathbf{x}) = (1, \mathbf{x}^T)$	$(-0.4926, -0.628, -0.3283, 0.4378, 0.5283, -0.612, -0.6837, -0.2061)^T$	$[-3, 3]^7_{0.001}$	7	8
9	$y(\mathbf{x}) \sim Bin(1, \eta)$, $\eta = \frac{e^{\mathbf{h}^T(\mathbf{x})\theta}}{1+e^{\mathbf{h}^T(\mathbf{x})\theta}}$, $\mathbf{h}^T(\mathbf{x}) = (1, \mathbf{x}^T)$	$(3, 0.5, 0.75, 1.25, 0.8, 0.5, 0.8, -0.4, -1.0, 2.65, 0.65)^T$	$\{-1, 1\}^4 \times [50, 90]_{0.01} \times [30, 55]_{0.01} \times [0, 10]_{0.01} \times [18, 48]_{0.01} \times [0.125, 0.425]_{0.001} \times [5, 15]_{0.01}$	10	11
10	$y(\mathbf{x}) \sim Bin(1, \eta)$, $\eta = \frac{e^{\mathbf{h}^T(\mathbf{x})\theta}}{1+e^{\mathbf{h}^T(\mathbf{x})\theta}}$, $\mathbf{h}^T(\mathbf{x}) = (1, \mathbf{x}^T, x_1x_9, x_2x_5, x_3x_4, x_6x_7, x_8x_{10})$	$(3, 0.5, 0.75, 1.25, 0.8, 0.5, 0.8, -0.4, -1.0, 2.65, 0.65, 0.01, -0.02, 0.03, -0.04, 0.05)^T$	$\{-1, 1\}^4 \times [50, 90]_{0.01} \times [30, 55]_{0.01} \times [0, 10]_{0.01} \times [18, 48]_{0.01} \times [0.125, 0.425]_{0.001} \times [5, 15]_{0.01}$	10	16

The levels of discretization of the continuous intervals are indicated by subscripts. For instance, $[-1, 1]^2_{0.001}$ denotes two factors ranging from -1 to 1, each of them discretized with a step size of 0.001. The columns labeled k and m give the number of factors and the number of parameters, respectively. The symbol F in the definition of Model 6 denotes the standard normal cumulative distribution function

Table 4 Comparison of GEX with the competing methods on the benchmark models in Table 3

#	t_{GEX} (sec)	Φ^*_{GEX}	s_{GEX}	Source COM	t_{COM} (sec)	Φ^*_{COM}	s_{COM}	eff_{COM}
2	2.20 –3.67	0.117578	9	Pronzato and Zhigljavsky (2014)	3.68–3.92	0.117578–0.117578	9	1.0000
3	0.96 –1.51	0.221567	16	Pronzato and Zhigljavsky (2014)	3.05–3.33	0.221567–0.221567	16	1.0000
1	0.22 –0.45	0.0338935	4	Duarte et al. (2018)	12.00	0.0338904	4	0.9999
4	1.74 –2.25	0.870542	19	Duarte et al. (2018)	39.81	0.853086	15	0.9799
5	2.84 –5.19	0.351996	15	Lukemire et al. (2019)	61.05–63.76	0.348210–0.350217	15	0.9892-0.9949
9	2.05 –3.06	0.0381948	13	Lukemire et al. (2019)	257.89–321.38	0.0378399–0.0381872	13	0.9907-0.9998
8	15.04 –28.63	1.07287	26	Xu et al. (2019)	≤ 600	1.06962	33	0.9970
10	4.96 –6.89	0.0115145	19	Xu et al. (2019)	≤ 60	0.0114329	18	0.9929
6	7.53 –12.95	1.26609	16	García-Ródenas et al. (2020)	34.36–225.24	1.05263–1.23457	≤ 25	0.8314–0.9751
7	1.86 –2.88	0.539359	15	García-Ródenas et al. (2020)	9.82–83.23	0.465116–0.526316	≤ 25	0.8623–0.9758

The first column gives the model number, the column labeled t_{GEX} gives the minimum and maximum computation times out of 11 runs of GEX, Φ^*_{GEX} is the obtained criterion value, and s_{GEX} is the support size of the obtained design (for each model, the designs resulting from all 11 runs had the same criterion value as well as the same support size). The columns labeled t_{COM}, Φ^*_{COM} and s_{COM} present the computation times, criterion values and the support sizes, respectively, of the designs computed via the competing methods (specified in the column titled "Source COM"). The column labeled eff_{COM} lists the efficiencies $\Phi^*_{COM}/\Phi^*_{GEX}$ of the designs computed by the competing methods relative to the designs computed by GEX. More detailed descriptions are given in the text

Models 6 and 7 are available in Table 11 of García-Ródenas et al. (2020); note that the listed results encompass the time and efficiency performance of 5 different algorithms.

From Table 4, it is clear that GEX produces designs with higher efficiency than the competing methods in a shorter computation time, sometimes by several orders of magnitude. The only close competitor is the algorithm from Pronzato and Zhigljavsky (2014). In fact, the designs for Models 2 and 3 obtained by this algorithm are slightly better than the designs for the same models obtained by GEX, although the difference is not within 6 significant digits of the criterion values displayed in Table 4. However, the practical applicability of the algorithm from Pronzato and Zhigljavsky (2014) is limited to models with small numbers of factors; if the number of factors is greater than 2, the computation becomes exceedingly slow. Moreover, in contrast to GEX, the application of this algorithm may require theoretical analysis specific to the problem.

Nevertheless, we stress that all compared algorithms have their own advantages; for instance, they may be more appropriate for computing efficient designs for very general models or special optimality criteria.

4 Final comments

We have proposed a conceptually simple approach and its concrete and efficient specification, which we call the galaxy exploration method (GEX). The proposed algorithm can be used for computing D-optimal or nearly D-optimal approximate experimental designs on large grids and, by means of dense discretization, on continuous multidimensional cuboid spaces.

With a suitable transformation or embedding, GEX can also be applied to compute efficient designs on noncuboid design spaces. For instance, suppose that the experimental design space is the unit disk $\mathfrak{D}$ in a plane. Then, we have the following two options for facilitating the application of GEX:

1. Use the radial and angular coordinates of the design points as independent factors. More formally, transform $\mathfrak{D}$ into a cuboid design space $\mathfrak{X} = [0, 1] \times [0, 2\pi]$ by means of the polar transformation.
2. Embed $\mathfrak{D}$ into the larger design space $\mathfrak{X} = [-1, 1] \times [-1, 1]$ and extend the set $\mathbf{f}(\mathbf{x})$, $\mathbf{x} \in \mathfrak{D}$, with artificial $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ for all $\mathbf{x} \in \mathfrak{X} \setminus \mathfrak{D}$.

While feasible, assessing the actual performance of GEX under such transformations and embeddings, or adapting the principle of GEX itself to noncuboid design spaces, will require further research.

Although we have focused on D-optimality in this paper, GEX can be directly adapted to any other criterion for which there exist a simple analogue of the variance function and an appropriate discrete-space solver. This is the case for the popular c-, A-, and I-optimality criteria (see, e.g., Harman and Jurík 2008; Harman et al. 2020) as well as for other criteria.

In this paper, we have used the approach of local optimality to compute efficient designs for nonlinear models. Note that locally optimal designs not only are useful by themselves but also are crucial for several methods of constructing robust experimental designs; as examples, let us mention so-called maximin efficient designs (see, e.g., Müller and Pázman 1995; Dette et al. 2006) and the clustering methodology developed by Dror and Steinberg (2006). For other approaches to experimental design of nonlinear models, such as the pseudo-Bayesian and minimax approaches, the potential for utilizing GEX will require further research. In the meantime, we refer the reader to alternative algorithms, such as those proposed by Chen et al. (2015), Masoudi et al. (2017), Lukemire et al. (2019), and Masoudi et al. (2019).

The performance of GEX can be undoubtedly improved even further, for instance, through modification of the exploration sets, clever adaptive changes to the levels of each factor, or parallelization of the computation. These are also interesting topics for further research.

Acknowledgements This work was supported by Grant No. 1/0341/19 from the Slovak Scientific Grant Agency (VEGA). We are grateful to Dr. Luc Pronzato and Prof. Anatoly Zhigljavsky for the MATLAB codes implementing their algorithm from Pronzato and Zhigljavsky (2014) as well as to Dr. Weinan Xu for the information on the computation of the designs from Xu et al. (2019). We also thank anonymous reviewers and editors who significantly contributed to the clarity of the presentation.

Funding This study was funded by Grant No. 1/0341/19 from the Slovak Scientific Grant Agency (VEGA).

Availability of data and material Not applicable.

Declaration

Conflicts of interest The authors have no relevant financial or nonfinancial interests to disclose. The authors have no conflicts of interest to declare that are relevant to the content of this article.

Code availability Computer codes that can be used to replicate the numerical results can be downloaded from the web page of the authors, as provided in the article.

References

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

Atkinson, A.C., Woods, D.C. Designs for generalized linear models. Handbook of design and analysis of experiments, pp. 471–514 (2015)

Atwood, C.L.: Sequences converging to D-optimal designs of experiments. Ann. Stat. **1**, 342–352 (1973)

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

Chaloner, K., Larntz, K.: Optimal Bayesian design applied to logistic regression experiments. Journal of Statistical Planning and Inference **21**, 191–208 (1989)

Chen, R.B., Chang, S.P., Wang, W., Tung, H.C., Wong, W.K.: Minimax optimal designs via particle swarm optimization methods. Statistics and Computing 25(5), 975–988 (2015)

Chernoff, H.: Locally optimal designs for estimating parameters. The Annals of Mathematical Statistics 24, 586–602 (1953)

Dette, H., Martinez Lopez, I., Ortiz Rodriguez, I.M., Pepelyshev, A.: Maximin efficient design of experiment for exponential regression models. Journal of Statistical Planning and Inference **136**, 4397–4418 (2006)

Dror, H.A., Steinberg, D.M.: Robust Experimental Design for Multivariate Generalized Linear Models. Technometrics **48**, 520–529 (2006)

Duarte, B.P., Wong, W.K.: A semi-infinite programming based algorithm for finding minimax optimal designs for nonlinear models. Statistics and Computing 24, 1063–1080 (2014)

Duarte, B.P., Wong, W.K., Dette, H.: Adaptive grid semidefinite programming for finding optimal designs. Statistics and Computing 28, 441–460 (2018)

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

Filová, L., Harman, R.: Ascent with quadratic assistance for the construction of exact experimental designs. Comput. Statistics **35**, 775–801 (2020)

García-Ródenas, R., García-García, J.C., López-Fidalgo, J., Martín-Baos, J.A., Wong, W.K.: A comparison of general-purpose optimization algorithms for finding optimal approximate experimental designs. Comput. Stat. Data Anal. **144**, 106844 (2020)

Gribik, P.R., Kortanek, K.O.: Equivalence Theorems and Cutting Plane Algorithms for a Class of Experimental Design Problems. SIAM J. Appl. Math. **32**, 232–259 (1977)

Goos, P., Jones, B.: Optimal design of experiments: a case study approach. Wiley, New York (2011)

Harman, R., Bachratá, A., Filová, L.: Construction of efficient experimental designs under multiple resource constraints. Applied Stochastic Models in Business and Industry 32, 3–17 (2016)

Harman, R., Filová, L., Richtárik, P.: A randomized exchange algorithm for computing optimal approximate designs of experiments. Journal of the American Statistical Association 115, 348–361 (2020)

Harman, R., Jurík, T.: Computing c-optimal experimental designs using the simplex method of linear programming. Computational Statistics & Data Analysis **53**, 247–254 (2008)

Harman, R., Pronzato, L.: Improvements on removing nonoptimal support points in D-optimum design algorithms. Statistics & Probability Letters **77**, 90–94 (2007)

Harman, R., Rosa, S.: On greedy heuristics for computing D-efficient saturated subsets. Operations Research Letters 48, 122–129 (2020)

Heiligers, B.: Admissible experimental designs in multiple polynomial regression. Journal of Statistical Planning and Inference **31**, 219–233 (1992)

Kiefer, J.: Optimum experimental designs. J. Roy. Stat. Soc. Ser. B (Methodol.) **21**, 272–304 (1959)

Khuri, A.I., Mukherjee, B., Sinha, B.K., Ghosh, M.: Design issues for generalized linear models: A review. Stat. Sci. **21**, 376–399 (2006)

Lukemire, J., Mandal, A., Wong, W.K.: D-qpso: A quantum-behaved particle swarm technique for finding D-optimal designs with dis-

crete and continuous factors and a binary response. Technometrics **61**, 77–87 (2019)
Masoudi, E., Holling, H., Duarte, B.P., Wong, W.K.: A metaheuristic adaptive cubature based algorithm to find Bayesian optimal designs for nonlinear models. Journal of Computational and Graphical Statistics 28(4), 861–876 (2019)
Masoudi, E., Holling, H., Wong, W.K.: Application of imperialist competitive algorithm to find minimax and standardized maximin optimal designs. Computational Statistics & Data Analysis **113**, 330–345 (2017)
Meyer, R.K., Nachtsheim, C.J.: The coordinate-exchange algorithm for constructing exact optimal experimental designs. Technometrics 37, 60–69 (1995)
Müller, C., Pázman, A.: Applications of necessary and sufficient conditions for maximin efficient designs. Metrika 48, 1–19 (1995)
Pázman, A.: Foundations of optimum experimental design. Reidel, Dordrecht (1986)
Pronzato, L., Pázman, A.: Design of Experiments in Nonlinear Models. Springer, New York (2013)
Pronzato, L., Zhigljavsky, A.A.: Algorithmic construction of optimal designs on compact sets for concave and differentiable criteria. J. Stat. Plan. Inference **154**, 141–155 (2014)
Pukelsheim F (2006). Optimal Design of Experiments (Classics in Applied Mathematics). SIAM, Philadelphia
Pukelsheim, F., Rieder, S.: Efficient rounding of approximate designs. Biometrika 79, 763–770 (1992)
Papp, D.: Optimal Designs for Rational Function Regression. J. Am. Stat. Assoc. **107**, 400–411 (2012)
: , (2020)
Sagnol, G.: Computing optimal designs of multiresponse experiments reduces to second-order cone programming. Journal of Statistical Planning and Inference **121**, 1684–1708 (2011)
Sagnol, G., Harman, R.: Computing exact D-optimal designs by mixed integer second order cone programming. The Annals of Statistics 43, 2198–2224 (2015)
Silvey, S.D., Titterington, D.H., Torsney, B.: An algorithm for optimal designs on a design space. Communications in Statistics-Theory and Methods **7**, 1379–1389 (1978)
Schwabe, R.: Optimum Designs for Multi-factor Models, vol. 113. Springer, Berlin (2012)
Stokes, Z., Mandal, A., Wong, W.K.: Using Differential Evolution to design optimal experiments. Chemom. Intell. Lab. Syst. **99**, 103955 (2020)
Stufken, J., Yang, M.: On locally optimal designs for generalized linear models with group effects. Statistica Sinica 22, 1765–1786 (2012)
Uciński, D., Patan, M.: D-optimal design of a monitoring network for parameter estimation of distributed systems. Journal of Global Optimization 39, 291–322 (2007)
Vandenberghe, L., Boyd, S., Wu, S.: Determinant maximization with linear matrix inequality constraints. SIAM J. Matrix Anal. **19**, 499–533 (1998)
Xu, W., Wong, W.K., Tan, K.C., Xu, J.X.: Finding High-dimensional D-optimal designs for logistic models via differential evolution. IEEE Access **7**, 7133–7146 (2019)
Yang, M., Biedermann, S., Tang, E.: On optimal designs for nonlinear models: a general and efficient algorithm. Journal of the American Statistical Association 108, 1411–1420 (2013)
Yu, Y.: D-optimal designs via a cocktail algorithm. Statistics and Computing 21, 475–481 (2011)
Wong, W.K., Chen, R.B., Huang, C.C., Wang, W.: A modified particle swarm optimization technique for finding optimal designs for mixture models. PLoS ONE **10**, e0124720 (2015)
Wu, C.F.: Some algorithmic aspects of the theory of optimal designs. The Annals of Statistics 6, 1286–1301 (1978)
Wu, C.F.: Some iterative procedures for generating nonsingular optimal designs. Communications in Statistics-Theory and Methods **7**, 1399–1412 (1978)
Zhang, Z., Wong, W.K., Tan, K.C.: Competitive swarm optimizer with mutated agents for finding optimal designs for nonlinear regression models with multiple interacting factors. Memetic Computing **12**, 219–233 (2020)