Journal of Statistical Planning and Inference 171 (2016) 1–9

Contents lists available at ScienceDirect

Journal of Statistical Planning and Inference

journal homepage: www.elsevier.com/locate/jspi

Privacy sets for constrained space-filling

Eva Benková [a,b], Radoslav Harman [a], Werner G. Müller [b,*]

[a] Department of Applied Mathematics and Statistics, Comenius University Bratislava, Slovakia
[b] Department of Applied Statistics, Johannes Kepler University Linz, Austria

ARTICLE INFO

Article history:
Received 20 November 2015
Received in revised form 10 December 2015
Accepted 11 December 2015
Available online 17 December 2015

Keywords:
Computer experiments
Design algorithms
Optimal design

ABSTRACT

Utilizing a typology for space filling into what we call "soft" and "hard" methods, we introduce the central notion of "privacy sets" for dealing with the latter. This notion provides a unifying framework for standard designs without replication, Latin hypercube designs, and Bridge designs, among many others. We introduce a heuristic algorithm based on privacy sets and compare its performance on some well-known examples. For instance, we demonstrate that for the computation of Bridge designs this algorithm performs significantly better than the state-of-the-art method. Moreover, the application of privacy sets is not restricted to cuboid design spaces and promises improvements for many other situations.

1. Introduction

Most of the literature on space-filling designs attempts to achieve its aim by optimizing a prescribed objective measuring a degree of space-fillingness, such as maximin, minimax, etc., sometimes combined with an estimation or prediction oriented criterion (like suggested in Morris and Mitchell (1995)). Let us label those as "soft" space-filling methods. In contrast, "hard" space-filling methods ensure desirable properties by enforcing constraints on the designs, such that a secondary criterion can be used for optimization (e.g. D-optimality in the case of the Bridge-designs of Jones et al. (2014)). In this paper we intend to propose a general framework for the latter methods based on the central notion of so-called privacy sets, which allows us elegant and efficient formulations of design algorithms. Eventually we will compare our "hard" techniques to the more established "soft" procedures.

Formally, let $\mathcal{X} \subseteq \mathbb{R}^d$ be a non-empty design space and let ξ be a finite subset of $\mathcal{X}$ representing an experimental design, with each point of ξ being a design point of a single trial of the corresponding experiment. The notion of a "set" automatically implies that the order of design points is irrelevant, as well as that the replicated observations in the same design point are not allowed. Without loss of generality, the standard problem of optimal designs of experiments (cf. e.g. Pázman, 1986) is to maximize some criterion in the set of all permissible designs Γ, that is, to find $\xi^* \in \operatorname{argmax}_{\xi\in\Gamma}\{\Phi(\xi)\}$, where $\Gamma \subseteq \Xi$ and Ξ denotes the set of all designs. Here, the soft methods of space-filling focus on proposing and tuning the criterion Φ and then optimize on the set Γ of all designs of a given size. Hard methods, in contrast, pose restrictions on the design, such that all designs from Γ satisfy a required minimum level of space-fillingness and Φ can play the role of a criterion taking statistical efficiency of the design into account.

The statistical quality of one design compared to another can be assessed by calculating their mutual efficiency, which is defined as $\mathrm{eff}(\xi|\eta) = \Phi(\xi)/\Phi(\eta)$ for any $\xi, \eta \in \Xi$, $\Phi(\eta) > 0$.

* Corresponding author.
E-mail address: werner.mueller@jku.at (W.G. Müller).

http://dx.doi.org/10.1016/j.jspi.2015.12.004

2. Privacy sets algorithm

Let us now define the central notion for our approach to the generation of (constrained) space-filling designs.

Definition 1. *For each $x \in \mathcal{X}$, let $\mathcal{P}(x) \subseteq \mathcal{X}$, $x \in \mathcal{P}(x)$, be a given privacy set of the point x. For any design $\xi \in \Xi$, let $\mathcal{P}(\xi) = \cup_{x\in\xi} \mathcal{P}(x)$ be the privacy set of the design ξ.*

We assume that there is a given upper limit on the size of the experiment $N \in \mathbb{N}$. A design ξ will then be called permissible, if $|\xi| \leq N$ and $x \notin \mathcal{P}(y)$ for all $x, y \in \xi$, $x \neq y$. A permissible design ξ will be called maximal permissible if it cannot be augmented without violation of some of the constraints, i.e., if $\xi \cup \{x\}$ is not permissible for all $x \in \mathcal{X} \setminus \xi$.

Assumption 1. We assume that $|\xi| = N$ for any maximal permissible design.

We would like to emphasize that the idea of privacy sets is not artificial, but can be used for example to ensure various space-filling properties. In fact, many of the widely-used designs, starting from the classical exact designs (cf. e.g. Casella, 2008) restricted by at most one trial at each point, to popular Latin hypercube designs (LHD, see McKay et al., 1979), to Bridge designs (BD, see Jones et al., 2014), can be formulated in the terms of privacy sets. More specifically, we can set $\mathcal{P}(x) = \{x\}$ in the case of classical designs. For Latin hypercube designs the design space $\mathcal{X}$ is usually a finite grid and we have $\mathcal{P}(x) = \{y : \exists i \in \{1, \ldots, d\} : x_i = y_i\}$, where $x, y \in \mathbb{R}^d$ and x_i, y_i denote their ith coordinates. Privacy sets for Bridge designs are given by Eq. (2) in Section 3.

Note that the use of privacy sets is meaningful not only for computer, but also for physical experiments. It covers, for example, time-separation constraints, where the designs space represents time, and consecutive trials must be performed at least δ time units apart (see, e.g., the second example in Section 5 of the paper Sagnol and Harman, 2015). In this case, $\mathcal{X} = \mathbb{R}_0^+$ and $\mathcal{P}(x) = (x - \delta, x + \delta) \cap \mathcal{X}$ for all $x \in \mathcal{X}$.

The set $\mathcal{P}(x)$ is typically some kind of a neighbourhood of x (containing also x itself) securing some "privacy" for it, although this is not strictly required by the definition itself. Using the privacy sets defined above we obtain the optimization problem

$$\xi^* \in \operatorname*{argmax}_{\xi\in\Xi} \{\Phi(\xi);\ |\xi| \leq N \text{ and } x \notin \mathcal{P}(y) \text{ for all } x, y \in \xi, x \neq y\}. \tag{1}$$

In the following, we present a framework for an exchange-type algorithm – Privacy Sets Algorithm (PSA) – for solving optimization problem (1). In general, the specification of individual steps depends greatly on the design space $\mathcal{X}$ and on the constraints given by the sets $\mathcal{P}(x)$, $x \in \mathcal{X}$, as well as on the optimization criterion Φ.

A characteristic feature of PSA is the ability to temporarily violate "privacy" of one or more design points. This offers a wider range of possibilities than when performing only permissible changes and prevents the algorithm from getting stuck, for instance when the privacy constraints are very strict. Let $\mathcal{A}(\xi) \subseteq \mathcal{X} \setminus \xi$ denote a set of "candidate points" that can possibly augment a maximal design ξ. The set $\mathcal{A}(\xi)$, in contrast to $\mathcal{P}(\xi)$, is not an attribute of the problem itself, but can be adjusted in order to ensure the optimum performance of the algorithm. Note that we do not require $\mathcal{A}(\xi)$ to contain solely permissible points $x \notin \mathcal{P}(\xi)$.

PSA does not pose any restrictions on the design space $\mathcal{X}$. Due to implementation reasons, however, we always assume a finite design space of size $n \in \mathbb{N}$. If n is relatively small (up to thousands of design points, say), the implementation of PSA is rather simple for all kinds of privacy sets. This is mainly true because of the ability to store information about availability of each individual point of the design space. However, with n increasing, it becomes computationally intensive or even unfeasible to keep n-dimensional vectors in the computer memory and certain specific features of a particular class of privacy sets have to be considered.

One of the key parts of PSA is the efficient augmentation of a design that is not maximal with the remaining runs to achieve the full size N. This is done by employing the following forward-type procedure (Algorithm 1) which adds permissible design points one-by-one until a maximal design is obtained.

Algorithm 1: Greedy Procedure (GrP)

Input : A permissible design ξ, $|\xi| = N^* < N$.
Output: A permissible design ξ, $|\xi| = N$

1 **for** $i = 1 : N - N^*$ **do**
2 | Augment ξ with the point x, which maximizes $\Phi(\xi \cup \{x\})$ subject to $x \notin P(\xi)$.
3 **end**
4 **return** ξ

One-point permissible augmentation from Step 2 can be crucial in effective implementing of PSA algorithm. One of the straightforward solutions is to use the exhaustive enumeration of $\mathcal{X} \setminus \mathcal{P}(\xi)$ (for smaller problems) or to use a blind random search, that is, to choose the best point from a set of candidates sampled independently from $\mathcal{X} \setminus \mathcal{P}(\xi)$. This