CASE STUDIES
Design Strategies and Approximation Methods for High-Performance Computing Variability Management

Yueyao Wang[1], Li Xu[1], Yili Hong[1], Rong Pan[2], Tyler Chang[3], Thomas Lux[3], Jon Bernard[3], Layne Watson[3], and Kirk Cameron[3]

[1]Department of Statistics, Virginia Tech, Blacksburg, VA 24061
[2]School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85281
[3]Department of Computer Science, Virginia Tech, Blacksburg, VA 24060

Abstract

Problem: Performance variability management is an active research area in high-performance computing (HPC). In this paper, we focus on input/output (I/O) variability, which is a complicated function that is affected by many system factors. To study the performance variability, computer scientists often use grid-based designs (i.e., full factorial designs) to collect I/O variability data and use mathematical approximation methods to build a prediction model. In statistics literature, space-filling designs (SFDs) and surrogates are popular for data collection and building predictive models. The applicability of SFDs and surrogates in HPC variability management setting, however, needs investigation. In this case study, we investigate their applicability in HPC setting in terms of design efficiency, prediction accuracy, and scalability.

Approach: We first customize the existing SFDs so that they can be applied in the HPC setting. We conduct a comprehensive investigation of design strategies and the prediction ability of approximation methods. We use both synthetic data simulated from three test functions and the real data from HPC setting. We then compare different methods in terms of design efficiency, prediction accuracy, and scalability.

Results: In our synthetic and real data analysis, GP with SFDs outperforms in most scenarios. With respect to the choice of approximation models, GP is recommended if the data are collected by SFDs. If data are collected in grid-based way, both GP and Delaunay are worth trying. With the best choice of approximation method, the

performance of SFDs and GBDs depends on the property of underlying surface. For the cases where SFDs are better, we can use about half or less the design budget for the GBD to achieve the same prediction accuracy. Although we observed that the GBD can also beat SFDs for one case, GBD is not scalable to high-dimensional experiment regions. Therefore, SFDs are recommended especially when large numbers of input factors need to be considered in the model.

Key Words: Computer Experiment; Delaunay Triangulation; Gaussian Process; Linear Shepard's Method; MARS; Space-Filling Design.

1 Problem Description

The computing scale and complexity in modern technologies and scientific areas make high-performance computing (HPC) increasingly important. Performance variability, however, is an important challenge in the research of HPC systems that has been observed for a long time (e.g., Giampapa et al. 2010, Akkan, Lang, and Liebrock 2012, Cameron et al. 2019). High variability in HPC systems can lead to unstable system performance and potentially high energy costs. Therefore, variability management is crucial for system performance optimization. The performance variability is affected by many complicated interactions of factors in the system. In this study, we focus on input/output (I/O) performance variability. The relationship between system configurations (e.g., CPU frequency, file size, record size, the number of I/O threads, and I/O operation modes) and the I/O performance variability is of interest.

One framework that has been used for HPC variability management involves three steps Cameron et al. (2019), which are data collection on performance variability for a set of system configurations, building an approximation model to make predictions for new configurations, and using the prediction to do optimization for future designs. Statistical research is usually involved in the data collection and prediction steps. Although there is vast research on designs for data collections and approximation models for predictions, the applicability of those statistical methods in the setting of HPC performance management needs investigation, which motivates us to do a case study based on the HPC performance data.

In the data collection stage, researchers identify HPC system settings for which I/O throughput data should be collected. Computer scientists often use grid-based designs (GBDs) to collect data under numerous possible system configurations, when the number of factors is relatively small (Cameron et al. 2019, Xu et al. 2020). Note that the GBDs are equivalent to full factorial designs. In statistics literature, space-filling designs (SFDs) are often used, which assign design points far apart to fill the whole experiment region. In the prediction stage, an approximation model is built based on collected data for various system configurations. Mathematical approximation methods such as the linear Shepard's method (Shepard

1968) and Delaunay triangulation (Delaunay 1934) have been used. In statistics literature, Gaussian process (GP) models are popular for building approximation models.

From HPC application point of view, there are several questions that need to be addressed. First, due to HPC system constraints, the design region could be irregularly spaced. For example, in our experiments, the file size needs to be larger than or equal to the record size, causing complexity in the experimental design. Therefore, existing SFDs need to be tailored so that they are suitable for the HPC setting. Second, it is desirable to demonstrate SFDs can collect data more efficiently than the grid-based design (GBD) in a way that is accessible to computer scientists. Third, there is little research on the interaction between design strategies and approximation methods, especially in the setting of HPC variability management. However, it is possible that the prediction accuracy of a design may depend on the chosen approximation method.

Motivated by the needs in HPC performance variability management, we perform thorough comparisons between the prediction abilities of different approximation methods under different design strategies. We aim to recommend some efficient and scalable ways for computer scientists to collect performance data and provide a few practical guidelines in the HPC setting.

In literature, SFDs are widely used for experimental design when little information is known about the phenomenon to be studied. The uniform design proposed by Fang et al. (2000) has the natural idea of placing design points uniformly in the experimental region. Latin hypercube designs (McKay, Beckman, and Conover 1979) ensure good one dimensional projection properties. Some design strategies are based on the distance measure, such as the maximin and minimax designs (Johnson, Moore, and Ylvisaker 1990). There are also several designs constructed based on the variants or combinations of the aforementioned designs, such as the maximin Latin hypercube design (Morris and Mitchell 1995) and the maximum projection design (Joseph, Gul, and Ba 2015).

Non-regular and constrained design regions are quite common in industrial experiments and physical sciences. Some efforts have been made to allocate design points within such regions. Lekivetz and Jones (2015) proposed the fast flexible space filling algorithm which constructs designs based on hierarchical clustering. Pratola et al. (2017) map the original dimension to a higher dimensional space to convert the geodesic distance to Euclidean distance. Golchi and Loeppky (2015) adopt the idea of sampling from constrained distributions and propose a sequential constrained Monte Carlo algorithm to sample design points uniformly from the constrained input region.

When the physical experiment or the corresponding computer simulation model is complex and time-consuming, a surrogate model is needed to describe the underlying process. Many smooth techniques, such as response surface models, Kriging methods, kernel estimation, and

neural networks, can be used to approximate the true surface. Response surface methodology, originally introduced by Box and Wilson (1951), is a traditional technique for modeling the response variables given input variables. Another commonly-used statistical approximation model is Gaussian process regression (GP) (e.g., Sacks et al. 1989, Currin et al. 1991), which can generate a smooth surface and be capable to deal with the heteroscedasticity (Goldberg, Williams, and Bishop 1998) in the response variable. In the HPC community, mixture models have been used to study the multimodal behavior of the throughput distribution (Xu et al. 2020). Some novel numerical techniques including max box mesh, iterative box mesh, and Voronoi mesh methods for interpolation are investigated by Lux et al. (2018).

In previous work, the SFDs and approximation models have been compared, separately, in terms of prediction performance. The prediction accuracy of SFDs is studied by Johnson, Montgomery, and Bradley (2011) with Gaussian process surrogates. Multiple approximation methods are compared under GBDs by Lux et al. (2018) and Cameron et al. (2019). Those compared methods include regression methods, such as multivariate adaptive regression spline model, support vector regression, multilayered perceptron regression, and some numerical methods, such as linear Shepard's method and Delaunay triangulation. However, little study has been done regarding the interaction between design strategies and approximation methods in the HPC performance setting. This paper aims to investigate the prediction ability of different kinds of design strategies and approximation methods as a case study. So, when a particular design strategy is given, HPC engineers can choose the best approximation method to achieve a higher prediction accuracy, or vice versa.

The rest of the paper is presented as follows. Section 2 provides a detailed background of the motivating example and introduces the underlying problem that we are interested in. Sections 3.1 and 3.2 briefly summarize the SFDs and approximation methods that are under investigation in this paper. Section 3.3 conducts synthetic data analyses where the performance of all combinations of the five designs and five approximation methods are compared under three test functions. In Section 3.4, an HPC variability experiment is introduced and real-world comparison results are presented. Section 4 provides conclusions of the comparisons, practical guidelines, and areas for future work.

2 Data Collection and Preparation

We first introduce some notation for the HPC data. To define a generic experiment process, let $\boldsymbol{X}$ be a d-dimensional space of input factors with a set of constraints $C_{\boldsymbol{X}}(\boldsymbol{x})$, where $\boldsymbol{x} = (x_1, \ldots, x_d) \in \boldsymbol{X}$. Let $\boldsymbol{Y}$ be the random vector of the experiment outputs. The first step to start exploring the relationship between the input factors and the output is to efficiently

allocate design points within $\boldsymbol{X}$, which will be evaluated to obtain the corresponding output vectors. Suppose $\boldsymbol{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ is the set of selected design points, where $\boldsymbol{x}_i \in \boldsymbol{X}$ is the ith design point. Let y_i be the corresponding observed output at the ith design point. Then $\boldsymbol{y} = (y_1, \ldots, y_n)$ is the collected output vector containing the observed outputs at all $\boldsymbol{x}_i \in \boldsymbol{D}$.

In the HPC data, there are four numeric factors, including one hardware factor: CPU frequency (GHz); and three application factors: the number of I/O threads, the file size (KB), and the record size (KB). For our experiments, we consider CPU frequencies in the range $[2.0, 3.5]$ GHz, numbers of I/O threads in the range $[1, 64]$, and file sizes and record sizes in the range $[4, 16384]$ KB. Additionally, we have the constraint that for each system configuration, the file size must be greater than or equal to the record size, and the file size and record size must be of the form of $\sum_l 2^l$ for some positive integer l. To make the design region uniform, we apply a log transformation to the file size and record size: *log file size* $= \log_2$(*file size*), *log record size* $= \log_2$(*record size*). Then the experiment region becomes $\boldsymbol{X} = [2.0, 3.5] \times [1, 64] \times [2, 14] \times [2, 14]$ with the constraint that *log file size* $\geq$ *log record size*. In this application, we define the response of interest $\boldsymbol{Y}$ as the performance variability measurement (PVM). The PVM is given by the standard deviation of I/O throughputs (in the units of KB/s) at each input configuration (Cameron et al. 2019).

To collect data for this application, the I/O throughput of hard disk is collected in a grid-based pattern, using the IOzone benchmark (IOzone projects contributors 2016) to produce the workload with each system setting. Each factor is divided into k_i levels, $i = 1, \ldots, 4$ and a configuration is obtain by taking a possible combination of levels in each factor. Figures 1(a) and (b) show the two-dimensional projections of the real 4-dimension grid space where data are collected. At each configuration, the IOzone benchmark is run multiple times and the throughputs are gathered as an HPC performance measurement. The configurations are collected under 6 IO operation mode: initial_writers, rewriters, readers, re_readers, random_readers, random_writers. In total, we have 2658 configurations and each configuration has 300 replications to capture the performance variability. Since the data collection procedure is time-consuming, we are interested in whether SFDs can select points more efficiently than the GBDs.

After obtaining the data, another problem that we are interested in is the problem of accurately predicting HPC performance variability at a new configuration. With the collected dataset $\{\boldsymbol{D}, \boldsymbol{y}\}$, we want to build models that describe the relationship between system factors and performance variability. Based on the previous literature and studies, we adopt both statistical models and numerical models in computer science to explore the underlying relationship.

An example of the data structure is presented in Table 1. The 2D surface plots of the PVM under two pairs of factors are shown in Figure 2. From the surface plot, we gain a rough

Table 1: Example of data structure.

Frequency	Threads	File Size	Record Size	PVM
2.0	1	2	2	17411.29
2.0	1	4	2	58014.19
2.0	1	4	3	46393.96
2.0	1	4	4	43238.60
2.0	1	6	2	49839.42
2.0	1	6	3	109721.24

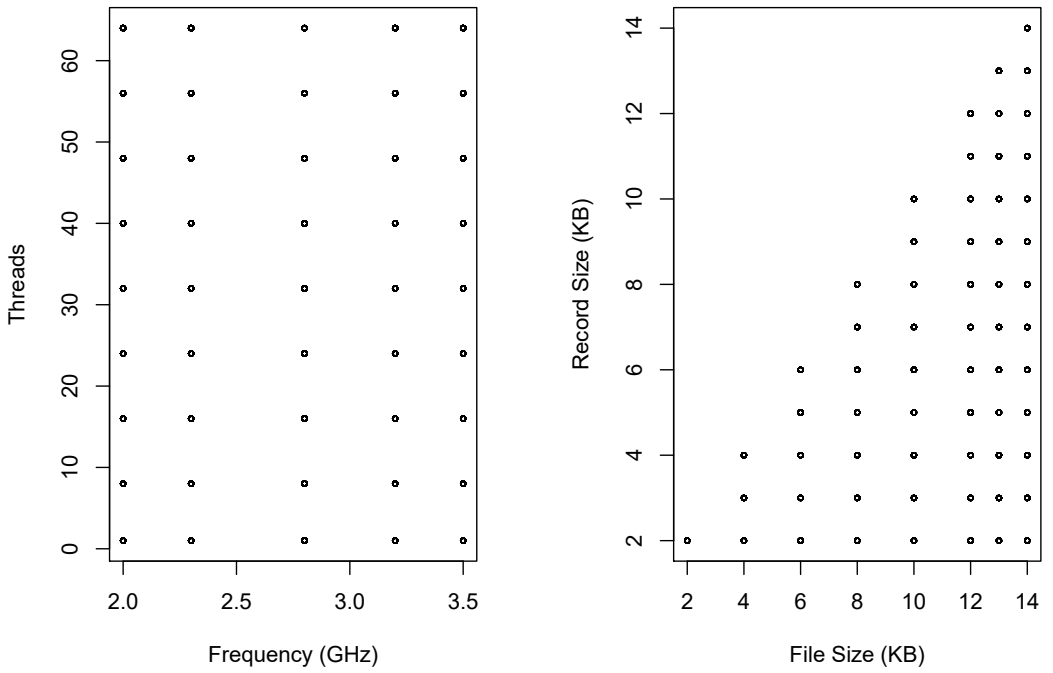
idea of the relationship between response and input factors. In Figure 2(a), we can see that the number of I/O threads and frequency have a positive relationship with the throughput variability; the larger the number of threads and frequency, the larger the PVM. Figure 2(b) shows that when the file size and record size are closer to each other, (i.e., near the boundary of the constraint in the plot), the variability is relatively small. When the file size is much larger than the record size, the performance varies a lot. These relationships are consistent with our intuitions.

3 Analysis and Interpretation

In this section, we conduct analysis on the effectiveness of designs and approximation methods in HPC setting. We first give brief descriptions on the design strategies and approximations in Sections 3.1 and 3.2, respectively. We then examine the the effectiveness of designs and approximation methods using synthetic data simulated from three different test functions. Finally, we study the designs and approximation methods using the real data from the HPC study.

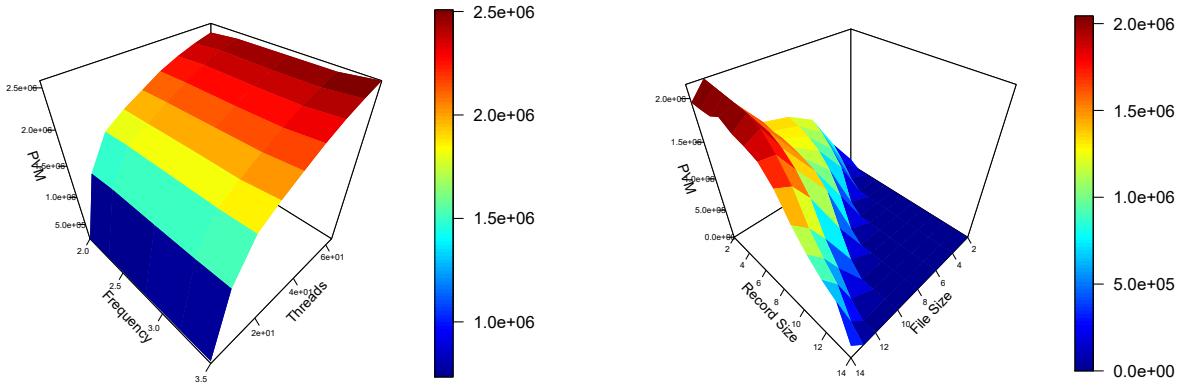
3.1 Design Strategies

Space-filling designs (SFDs), as the name suggests, spread out design points evenly in the experiment region in order to gather information from the whole experiment region. SFDs can assign points based on distance measures or sampling strategies. One reason to propose SFDs as an alternative to GBDs is that spreading points evenly across the entire design region is ideal when prediction accuracy is our primary goal. This is because the prediction error at a particular point depends on its location relative to the design points. If a design allocates points only near the center of the experiment region, a large error may result in prediction for inputs on the boundary of the experiment region. For convenience, we first assume that the experiment region is a unit hypercube $\boldsymbol{X} = [0, 1]^d$. This assumption can easily be relaxed



(a) Frequency & Number of Threads (b) File Size & Record Size

Figure 1: Two dimensional projection grids of design points where real data are collected.



(a) Frequency & Threads (b) File Size & Record Size

Figure 2: 3D surface plots of performance variability measure (PVM) under two pairs of factors.

with linear transformations. This section introduces several common criteria for constructing SFDs, as well as a list of designs we will study in this paper.

SFDs Constructed by Sampling Strategies

Several sampling strategies to construct SFDs are discussed in this section. An intuitive idea for a spread out design is to scatter points uniformly in the design region. Designs built in this way are referred to as uniform designs (UDs) (Fang et al. 2000). UDs can gather a sufficient amount of information to explore the relationship between the response variable and the input factors with relatively small runs (Li, Lin, and Chen 2004). UDs consider the whole design region equally important. However, when some portions of the domain are of more interest than others, stratified random sampling (SRS) can be used to enhance the design performance. Suppose n design points are desired. The SRS partitions the design region $\boldsymbol{X}$ into s strata and in stratum j, n_j points are selected based on a certain input distribution, where $j = 1, \ldots, s$ and $\sum_{j=1}^{s} n_j = n$. The size and position of each stratum depends on different experiment scenarios. When we know that some input factors are important to the response, we also want the design points' projections on to those factors to be spread out. This can be achieved by Latin hypercube design (LHD) (McKay, Beckman, and Conover 1979), which is a popular SFD and has been combined with various other designs. To construct an LHD of size n, the range of each input factor is equally divided into n intervals $[0, 1/n), \ldots, [(n-1)/n, 1]$. For each of the d coordinates, exactly one design point projection is sampled from each interval. In this way, it can be guaranteed that the design points are spread out across the range of each input factor. One favorable property of LHD is that any lower dimension projection of an LHD is also an LHD. Although LHDs have good properties for projection, it is not guaranteed that the design points will be spread out evenly in the entire experiment region. For example, it is possible that all design points could be located along the diagonal of the d-dimensional unit hypercube. To avoid this drawback, Latin hypercube sampling is often used together with other designs such as maximin design (Johnson, Moore, and Ylvisaker 1990) or maximum projection design (Joseph, Gul, and Ba 2015). These SFDs are based on a distance metric, which will be discussed in the next section.

SFDs Based on Distance Measures

In this section, design strategies based on distance measures and metrics are briefly introduced. One idea for SFDs is that no point in the experiment region $\boldsymbol{X}$ should be too far from its nearest neighbor in $\boldsymbol{D}$. Let

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\sum_{k=1}^{d} |x_{ik} - x_{jk}|^s \right)^{1/s}$$

be the distance metric. Usually, the Euclidean distance, (i.e., $s = 2$) is used. For an arbitrary point $\boldsymbol{x}$ in $\boldsymbol{X}$, we determine its closest design point $\boldsymbol{x}_i$ and the minimum distance $\min_i d(\boldsymbol{x}, \boldsymbol{x}_i)$. To guarantee that no point is too far from the design points, we choose the point $\boldsymbol{x} \in \boldsymbol{X}$ with the maximum distance to its closest design point, which is $\max_{\boldsymbol{x} \in \boldsymbol{X}} \min_i d(\boldsymbol{x}, \boldsymbol{x}_i)$. Then we find the design to minimize this distance,

$$\min_{\boldsymbol{D}} \max_{\boldsymbol{x} \in \boldsymbol{X}} \min_i d(\boldsymbol{x}, \boldsymbol{x}_i),$$

which is called the minimax distance design (Johnson, Moore, and Ylvisaker 1990). The second way to spread out points in $\boldsymbol{D}$ is to allocate the design points as far apart as possible. This can be realized by finding the design that maximizes the minimum distance between two design points, which is the criterion of the maximin distance design,

$$\max_{\boldsymbol{D}} \min_{i,j} d(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

The maximin distance design ensures that the design points are spread as far apart from each other as possible in the full dimension but does not guarantee that the design is space filling for each projection on to a subspace. Morris and Mitchell (1995) propose the maximin Latin hypercube (MmLh) design, which incorporates the structure of an LHD with the maximin design. The criterion is

$$\min_{\boldsymbol{D}} \left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{d^m(\boldsymbol{x}_i, \boldsymbol{x}_j)} \right]^{1/m}. \tag{1}$$

The MmLh design ensures good projection properties when projecting into one dimension, but fails to consider the projection onto other subdimensions. Joseph, Gul, and Ba (2015) propose maximum projection design (MaxPro) that ensures good space filling on all subspaces. It considers a weighted Euclidean distance

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{\theta}) = \left[\sum_{k=1}^{d} \theta_k (x_{ik} - x_{jk})^2 \right]^{1/2}, \tag{2}$$

where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)$ is a vector of weight factors. Let $\theta_k = 1$ for those factors that construct the subspace and $\theta_k = 0$ for other factors, then (2) calculates the distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ after projection into subspaces. The criterion of maximum projection design can be modified based on (1) as

$$\min_{\boldsymbol{D}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{d^m(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{\theta})}.$$

Here, the weight factors satisfy $\sum_{k=1}^{d} \theta_k = 1$. To properly choose the weight factor $\boldsymbol{\theta}$, authors adopt the Bayesian framework. A prior distribution is assigned to $\boldsymbol{\theta}$ and the expected value

of the objective function is minimized. The criterion can be further simplified with uniform prior and $m = 2d$:

$$\min_{\boldsymbol{D}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{\prod_{k=1}^{d} (x_{ik} - x_{jk})^2}. \tag{3}$$

It is clear from (3) that any two design points in $\boldsymbol{D}$ can not have the same value in any dimension. Otherwise, $x_{ik} - x_{jk} = 0$, which will cause the objective function to be infinite. The criterion automatically guarantees that the design also has the LHD property.

Another design strategy, the maximum entropy design is also included in this section, although generally it is considered to be a model-based design. The maximum entropy design (Shewry and Wynn 1987) maximizes the negative entropy function, which captures the amount of information gained from an experiment. The entropy is defined as $H(\boldsymbol{X}) = -\int_{\boldsymbol{X}} f(\boldsymbol{x}) \log[f(\boldsymbol{x})] d\boldsymbol{x}$, where $f(\boldsymbol{x})$ is a density function. The information gained is $I(\boldsymbol{X}) = -H(\boldsymbol{X})$. We want the experimental design that causes the largest change in the information, which is equivalent to maximizing the entropy:

$$\max_{\boldsymbol{D}} H(\boldsymbol{X}).$$

This can be simplified as $\max_{\boldsymbol{D}} \log |\Sigma_n|$ if we assume that the underlying surface is a Gaussian process, where Σ_n is the variance-covariance matrix of $\boldsymbol{D}$. Then the construction of the maximum entropy design depends on the choice of the correlation function. In this study, we adopt the format that is implemented in the *DiceDesign*, where the correlation matrix is C with elements $r(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1 - 1.5d/a - 0.5(d/a)^3$ if $d > a$ and 0 otherwise. Here, d is the Euclidean distance between two points.

Customizing SFDs for HPC Setting

Four of the above designs are considered in this paper. They are uniform sampling, maximum Latin hypercube design (MnLh), maximum entropy design (MaxEnt), and maximum projection LHD (MaxPro). In the HPC performance variability modeling application, the additional constraint is that the file size needs to be larger than or equal to the record size. To deal with the constraint in the real application, we want to use an approach that can tailor all different designs. So above designs must be adjusted in order to accommodate the constrained region. In this study, a rejection sampling strategy, as a simple way that can be easily applied across different design strategies, is used. Suppose we would like to generate a design with size n. If points in the initial size n design fail to meet the constraint, then we generate designs with size $2n$, $3n$, ... until there are at least n points in the design that satisfy the constraint. Then we randomly select n points from the large design, which satisfy our constraint.

Note that this idea is different from rejection sampling, because we reject points that are obtained from a specific optimization problem. After discard the points that does not meet

the constraint, the remaining design points may not hold all the original properties of that type of design. However, this is a uniform, simple approach that can be applied to any type of design. We do not need to solve extra optimization problems or alter the existing sampling algorithm. Approaches that can propose discretized designs that maintain the space filling prosperities and also meet the constraint are desirable in the future work.

Besides tailoring SFDs for the constraint in the HPC problem, we also need to adjust designs to include boundary points so that we will not have bad extrapolation problems when we using the data to build numerical prediction models. The details are explained in 3.2. In this study, we use central composite designs to augment origin SFDs. A CCD consists of a factorial (fractional factorial) design with factors of two levels, i.e., the vertices of the experiment region, a set of center points, and a set of axial points. We add the vertex points, the center point of the axis, and the center point of the hypercube to the SFDs. If the experiment region is irregular, then the augmented points that do not meet the constraint are excluded.

3.2 Approximation Methods

To conduct the prediction accuracy comparison, we need to select several representative surrogates to approximate the underlying function. The approximation methods often used in both computer science and statistics are briefly introduced here.

Response Surface Methodology

Response surface methodology (RSM) is a field of statistics that investigates the relationship between a response variable and input factors through design experiments. It is a traditional method for studying computer experiments (**?**, **?**). Low-order polynomial models such as first- or second-degree polynomial models are commonly used in applications. In this study, we use a second-order model to approximate the true response surface.

In applied mathematics, numerical methods are often used to approximate the true surface. In this section, two numerical methods are used; they are Delaunay triangulation and Linear Shepard's method.

Delaunay Triangulation Interpolation

Delaunay triangulation interpolation is a numerical method that approximates the underlying function $f : \boldsymbol{R}^d \to \boldsymbol{R}$ based on the values $f(\boldsymbol{p})$ for a given vertex set $\boldsymbol{P}$ and the corresponding Delaunay triangulation. A Delaunay triangulation (Delaunay 1934) is given by any triangulation that satisfies the Delaunay properties. Suppose $\boldsymbol{P} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$ is a set of n points

in $\boldsymbol{R}^d$. Then a d-dimensional triangulation of $\boldsymbol{P}$, denoted $\mathrm{T}(\boldsymbol{P})$, is a set of d-simplices that satisfies the following criteria: 1) The vertex set of $\mathrm{T}(\boldsymbol{P})$ is $\boldsymbol{P}$; 2) the union of all simplices in $\mathrm{T}(\boldsymbol{P})$ is the convex hull of $\boldsymbol{P}$, denoted as $\mathrm{CH}(\boldsymbol{P})$; and 3) the d-simplices are disjoint besides their common boundaries (vertices, edges and facets). A Delaunay triangulation, denoted as $\mathrm{DT}(\boldsymbol{P})$, is the geometric dual of the Voronoi diagram. A Voronoi diagram divides $\boldsymbol{R}^d$ into n regions with each containing one Delaunay vertex in $\boldsymbol{P}$, such that all points within each region are closer to the vertex in their region than to any other vertex. The Delaunay triangulation is given by connecting all vertices whose Voronoi cells share a boundary with an edge.

Let $\boldsymbol{x} \in \mathrm{CH}(\boldsymbol{P})$ be an interpolation point and $S \in \mathrm{DT}(\boldsymbol{P})$ be the simplex with vertices $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{d+1}$ that contains $\boldsymbol{x}$. Find the weights $w_1, \ldots, w_{d+1}$ that satisfy $\sum_{i=1}^{d+1} w_i = 1$, $w_i \geq 0$, for $i = 1, \ldots, d+1$ and $\boldsymbol{x} = \sum_{i=1}^{d+1} w_i \boldsymbol{s}_i$. Then the estimated function value $\widehat{f}_{DT}(\boldsymbol{x})$ based on $\mathrm{DT}(\boldsymbol{P})$ is

$$\widehat{f}_{DT}(\boldsymbol{x}) = w_1 f(\boldsymbol{s}_1) + w_2 f(\boldsymbol{s}_2) + \cdots + w_{d+1} f(s_{d+1}).$$

In this paper, the Fortran 2003 package DELAUNAYSPARSE is used to perform the interpolations. In order to achieve computational efficiency, the algorithm in DELAUNAYSPARSE only computes a necessary, sparse subset of the Delaunay triangulation given pre-specified interpolation points (Chang et al. 2018).

In the HPC variability management application, in order to compute the predictions, we need to approximate the response values for a test set based on the Delaunay triangulation $\mathrm{DT}(\boldsymbol{D})$ of the proposed design $\boldsymbol{D}$. However, the points in the test set might not always fall inside $\mathrm{CH}(\boldsymbol{D})$, which results in an extrapolation problem instead of an interpolation problem. DELAUNAYSPARSE can handle extrapolation problems by projecting each test point onto $\mathrm{CH}(\boldsymbol{D})$ when the test point is close to $\mathrm{CH}(\boldsymbol{D})$, but this solution can perform badly if the test point is far outside of $\mathrm{CH}(\boldsymbol{D})$. Since we do not know whether a test point is inside $\mathrm{CH}(\boldsymbol{D})$ beforehand, necessary adjustments of the design strategies need to be made to avoid bad extrapolation problems.

In order to avoid bad extrapolation problems when using the Delaunay method to approximate the true surface, we need to augment the proposed SFDs so that the convex hull of each augmented SFD covers the entire experimental region. In this way, wherever a test point falls in the experimental region, it always results in an interpolation problem for the Delaunay method. To realize this idea, intuitively one can augment each SFD with those boundary points. In this study, we choose to use CCD to augment the proposed SFDs.

Linear Shepard Method

Shepard's method is a form of inverse distance weighting, originally proposed by Shepard (1968). It is an interpolation method based on the weighted average of basis functions, each

centered on a point in the data set $\boldsymbol{P} = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n)$, where the weight is calculated in terms of inverse distance from the interpolation point to points in $\boldsymbol{P}$. Several variations of Shepard's method are available such as, quadratic and cubic Shepard's method. However, in our application, linear Shepard's method is used for the sake of efficiency. Given the same setting as in Section 3.2, the original Shepard approximation of $f(\boldsymbol{x})$ at point $\boldsymbol{x}$ is:

$$\widehat{f}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} W_i(\boldsymbol{x}) f(\boldsymbol{p}_i)}{\sum_{i=1}^{n} W_i(\boldsymbol{x})},$$

where $W_i(\boldsymbol{x}) = 1/\|\boldsymbol{x} - \boldsymbol{p}_i\|_2^2$. This weight is nonzero for all the data points even those that are far away from the interpolation point $\boldsymbol{x}$. In order to achieve better approximation performance, a modified linear Shepard's method considers only the local points within an R-sphere of $\boldsymbol{x}$ and replaces the original $f(\boldsymbol{p}_i)$ with a linear approximation function $B(\boldsymbol{p}_i)$. The modified linear Shepard's method has the form

$$\widehat{f}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} W_i(\boldsymbol{x}) B(\boldsymbol{p}_i)}{\sum_{i=1}^{n} W_i(\boldsymbol{x})},$$

where the modified version weights are given by

$$W_i(\boldsymbol{x}) = \left[\frac{\max\left(0, R_i - d(\boldsymbol{x}, \boldsymbol{p}_i)\right)}{R_i d(\boldsymbol{x}, \boldsymbol{p}_i)} \right]^2.$$

Here, R_i is the radius of a sphere centered at $\boldsymbol{x}_i$ that reflects influence scope of $\boldsymbol{x}_i$. The Fortran package SHEPPACK (Thacker et al. 2010) is used in our study to perform the linear Shepard interpolation.

Besides the numerical methods, two semi-/non-parametric statistical models are considered in this study: multivariate adaptive regression splines (MARS) and Gaussian processes (GPs).

Multivariate Adaptive Regression Splines

Multivariate adaptive regression splines (MARS) were introduced by Friedman (1991). They are nonparametric regression models formed by spline basis product expansion. MARS can automatically capture nonlinearity and interaction effects. The MARS model is given by:

$$\widehat{f}(x) = \sum_{m=1}^{M} c_m B_m(x),$$

where $B_m(x)$ are the basis functions. These basis functions can be constants, hinge functions, or products of hinge functions, where each hinge function is of the form $\max(0, x - c)$ or $\max(0, c - x)$, where c is a constant. As a flexible nonparametric model, MARS tends to overfit without pruning. The algorithm for constructing MARS model is based on a modification of

recursive partition trees that requires a forward and backward pass. In the forward pass, the MARS model is initialized as a constant valued function (whose value is the intercept), then basis functions are gradually added to the model until the maximum number of terms is reached or the loss in sum of squares residual error is small. Next, a backward pass is used to prune the model based on the generalized cross validation (GCV) criterion, which is a trade off between goodness-of-fit and model complexity. In this study, we use the *earth* (Milborrow 2019) package in R to build the MARS model. A grid of hyper-parameter: the number of maximum terms in model and the maximum degree of interactions, is used to train the model in order to the select model with the highest Rsquare value.

Gaussian Process

Gaussian process (GP) interpolation is a commonly used approximation method in computer experiments. GP is often defined as a stochastic process where every finite collection of n observations follows a multivariate normal (MVN) distribution. A GP is determined by its mean function $\mu(\boldsymbol{x})$ and its covariance function $C(\boldsymbol{x}, \boldsymbol{x}')$. When a zero-mean GP is assumed, it can be completely determined by the covariance function, which is also referred to as the kernel function. There are several common kernel functions, including Gaussian $C(\boldsymbol{x}, \boldsymbol{x}') = \exp(-(\boldsymbol{x} - \boldsymbol{x}')^2/\theta)$ and Matérn kernels. Let $\boldsymbol{Y}_n = \{y_1, \ldots, y_n\}$ be the n observations at the proposed design points $\boldsymbol{X}_n = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and $Y \sim \mathrm{N}_n(0, \Sigma_n)$, where Σ_n is the covariance matrix with correlation elements $C(\boldsymbol{x}, \boldsymbol{x}')$. Then for an arbitrary point $\boldsymbol{x}$, the value of $Y(\boldsymbol{x})$ given the design and corresponding observations can be obtained by the conditional distribution $Y(\boldsymbol{x})|\,\{\boldsymbol{Y}_n, \boldsymbol{X}_n\}$. This can be calculated based on the conditional distribution of MVN:

$$Y(\boldsymbol{x})|\,\{\boldsymbol{Y}_n, \boldsymbol{X}_n\} \sim \mathrm{N}[\mu(\boldsymbol{x}), \sigma^2(\boldsymbol{x})],$$

where $\mu(\boldsymbol{x}) = \Sigma(\boldsymbol{x}, \boldsymbol{X}_n)\Sigma_n^{-1}\boldsymbol{Y}_n$ and $\sigma^2(\boldsymbol{x}) = \Sigma(\boldsymbol{x}, \boldsymbol{x}) - \Sigma(\boldsymbol{x}, \boldsymbol{X}_n)\Sigma_n^{-1}\Sigma(\boldsymbol{X}_n, \boldsymbol{x})$. It is obvious that the mean function is a linear combination of $\boldsymbol{Y}_n$ while the covariance function does not involve information of observations. A maximum likelihood estimator can be used for parameter estimation. We use the R package *laGP* (Gramacy 2016) to implement the local Gaussian process approximation. In the local GP model, if one wants to predict at $\boldsymbol{x}$, instead of using the whole design $\boldsymbol{D}$, a subset of $\boldsymbol{D}$ close to $\boldsymbol{x}$ is selected sequentially to increase the computing speed.

3.3 Synthetic Data Analyses

In order to investigate the performance of each design strategy and approximation method combination, we conduct synthetic data analyses using three test functions. Specifically, we

compare the root mean square error (RMSE) and the mean absolute percentage error (MAPE) of the predictions for each combination with each test function.

We have two goals when conducting the synthetic data analyses. First, we want to perform simulations with test functions that are representative of the HPC performance. Since this application has four input factors, we choose two test functions that have four input variables each, and for each of these test functions, we apply the same linear constraint function as in the HPC performance variance problem. Second, we would like to explore the prediction behavior and computing time of GBD and SFD with a high-dimension test function. So the 8-dimensional Borehole function is adopted to illustrate the design performance when the experiment region is of high-dimension.

Test Functions

The Friedman function was proposed by Friedman, Grosse, and Stuetzle (1983). It is a 5-dimensional function, and in our study, we map the first four variables to our experiment region and fix the last variable x_5. The function is:

$$f(x) = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5.$$

This function's domain is the unit hypercube: $x_i \in [0, 1]$, $i = 1, \ldots, 4$ with $x_5 = 0.5$. We apply the constraint $x_3 \geq x_4$ to the input domain to emulate the HPC performance variance problem.

The Colville function is a four-dimensional function with the formula:

$$f(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + \\ 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1).$$

The Colville function's input domain is $x_i \in [-10, 10]$, $i = 1, \ldots, 4$. The same constraint as with Friedman function was applied here.

The Borehole function is an eight-dimensional function that models water flow rate through a borehole. Let $x = \{r_w, r, T_u, H_u, T_l, H_l, L, K_w\}$ be the input variables, then the water flow rate is:

$$f(x) = \frac{2\pi T_u(H_u - H_l)}{\log(r/r_w)(1 + \frac{T_u}{T_l} + \frac{2LT_u}{\log(r/r_w)K_w r_w^2})}.$$

The input ranges are listed in Table 2. For this test function, no constraint is applied because the goal of this test function is to investigate the high-dimensional performance of the design strategy and the approximation method combinations.

Table 2: Ranges of parameters for the Borehole function.

Variable	minimum	maximum
r_w	0.05	0.15
r	100	50000
T_u	63070	115600
H_u	990	1110
T_l	63.1	116
H_l	700	820
L	1120	1680
K_w	9855	12045

Comparison Procedures

Using the above test functions, we want to compare the prediction accuracy of GBDs with that of proposed SFDs for each test function using various design sizes and approximation methods. Since the GBD is built by selecting n levels on each factor and then enumerate all possible combinations of four factors, the design size needs to be n^4. In a real application, it is possible that the number of levels at each factor are different. However since our test functions are continuous function, we assume that if we can take n levels on one factor, we can also take the same number of levels on all other factors, which means we only consider the fine "regular grid" in the synthetic data analyses. After deciding the size of GBD, we can generate SFDs correspondingly. The simulation procedure is as follow:

1. Choose a test function. Uniformly select n_g random points within the input region as the test set g.
2. For $n = 3, \ldots, 7$
   (a) Create a GBD D_g by choosing n points in each dimension and expanding into a grid via cartesian product.
   (b) To generate the SFDs, create designs $D_{maximin}$, D_{maxpro}, D_{maxent}, and $D_{uniform}$ each of size $N = n^4 - n_a$, where n_a is the size of the augmented design.
   (c) For each test function, find the corresponding values of points in the above designs.
   (d) Use five approximation methods to generate five predictive surfaces for each design:
      i. Linear regression: use backward selection to determine the second order linear regression model with minimum Bayesian information criterion (BIC).
      ii. Delaunay triangulation: use the DELAUNAYSPARSE package to build the model.

iii. Linear Shepard: use the SHEPPACK package to build the model.

iv. MARS: use cross validation to tune the MARS model on a hyper-parameter grid, and select the model with the highest Rsquare value.

v. Gaussian Process: use the separable Gaussian kernel with a nugget effect included in the model.

(e) Repeat Steps 2c - 2d B times.

(f) Compute average root mean squared error (RMSE) and mean absolute percentage error (MAPE) over repetitions for each SFD and approximation method.

Although in Section 3.1 we introduced that the SFDs are obtained by solving different optimization problems, those optimization problems often don't have analytic solutions if large number of points are desired in a high-dimensional experiment region. Numerical algorithms are usually used to obtain SFDs, which lead to non-unique solutions for a certain type of SFD. In order to understand the overall prediction performance for a certain type of SFD, we repeat the step of generating design and making prediction for B times. In the above procedures, the test set size n_g and the repetition number B is changed according to the dimension of the test function and the approximation method. For relatively smooth test functions or stable approximation methods, (e.g., Delaunay and MARS), we do not require a large number of replications or a large test set to obtain a stable and representative result. However for other models, such as using the linear Shepard's method under a non-smooth test function, we need to increase the replication number in order to get a reliable result. The summary of the test sizes n_g and replication numbers B are listed in Table 3.

For the Borehole function, since the dimension is relatively high, it is difficult to compute the prediction performance for a series of GBDs within a reasonable time and with reasonable computer resources. Therefore, we skip the step of generating multiple GBDs of the same size as each SFD. Instead, we consider one GBD of size $3^8 = 6561$ GBD and compare its performance with other SFDs varying sizes. For each test function, we plot the average RMSE and MAPE versus design size for each combination of approximation methods and design strategies. The results are shown in Figures 3, 4, and 5. The results show that the overall error decreases as the design size increases, but at different rates for different methods and problems. For the two 4-dimensional test functions, we can see an interaction effect between the approximation method and design strategy. Under numerical approximation methods, GBD has a smaller prediction error as size increasing compared to the SFDs. However, under statistical models, the trend is the opposite. One possible explanation for this behavior is that GBDs have good geometric properties and both numerical approximation methods, Delaunay and linear Shepard, rely on geometric properties to make predictions. For the 8-dimensional

Table 3: Test set size n_g and replication time B in the synthetic data analyses.

Surrogate Model	Fried		Colville		Borehole	
	n_g	B	n_g	B	n_g	B
RSM	10000	30	10000	30	5000	60
Delaunay	10000	30	10000	30	5000	60
LSP	10000	30	10000	30	5000	180
MARS	10000	30	10000	30	5000	60
GP	10000	30	10000	30	5000	60

Borehole function, the 3^8 GBD is shown as a horizontal line in Figure 5. In general for the Borehole function, the GBD does not perform as well as the SFDs, despite the fact that each SFD has a smaller design size.

3.4 HPC Data Analysis

In this section, we conduct a data analysis using the real data as described in Section 2.

Model Fittings

In HPC application, the data were collected using a full factorial design. Each of the four input factors has several unique levels and the throughput data was gathered at each combination of levels. In total there are 2658 possible configurations. Since the real data itself is a GBD, we construct SFDs with size increasing from 100 to 2700 with 200 interval and compare them to the real data. In this way, we investigate whether SFDs can achieve the same prediction accuracy as the GBD with a smaller design size.

A similar procedure as in Section 3.3 is used except for a few modifications. Since the experiment region in the HPC performance variance problem is a numerical discrete space, the SFD points are binned to the nearest feasible value after generation. Also because we do not know the underlying true surface in the real application as in the synthetic data analysis, we need to decide a underlying surface that can describe the real data well and also suitable for conducting comparisons. We choose to use a fitted model with methods in Section 3.2 using the real data as the truth. In order to choose the most appropriate model, we first use a 10-folds cross validation (CV) to see the average CV prediction errors for different models. The result is summarized in Table 4. Choosing the model that has the smallest CV prediction error is a natural idea. In this case, it's either Delaunay which has the smallest MAPE or GP with the smallest RMSE. However, one property of Delaunay method is that the fitted model traverse every training data point. This means if we use the fitted Delaunay trained from the real data as the underlying truth, the response value we take from the fitted surface for the

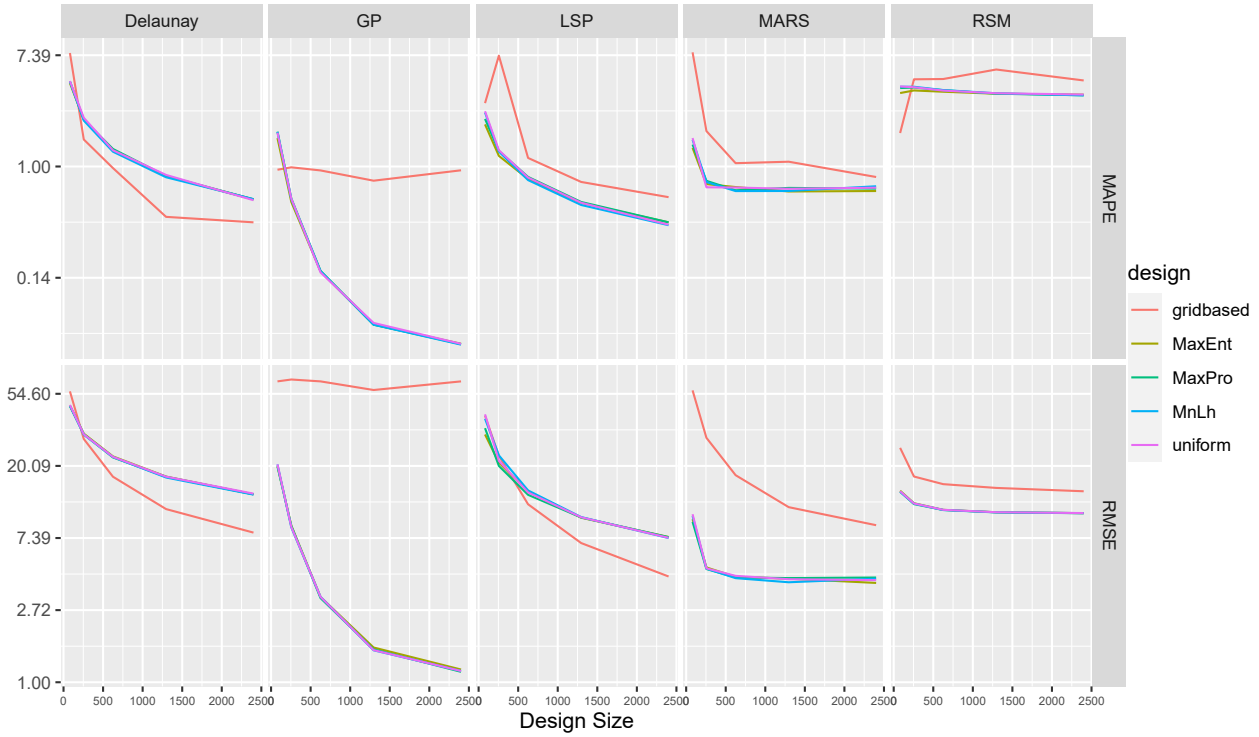


Figure 3: Colville test function RMSE and MAPE. RMSE is on the magnitude of 10^4.

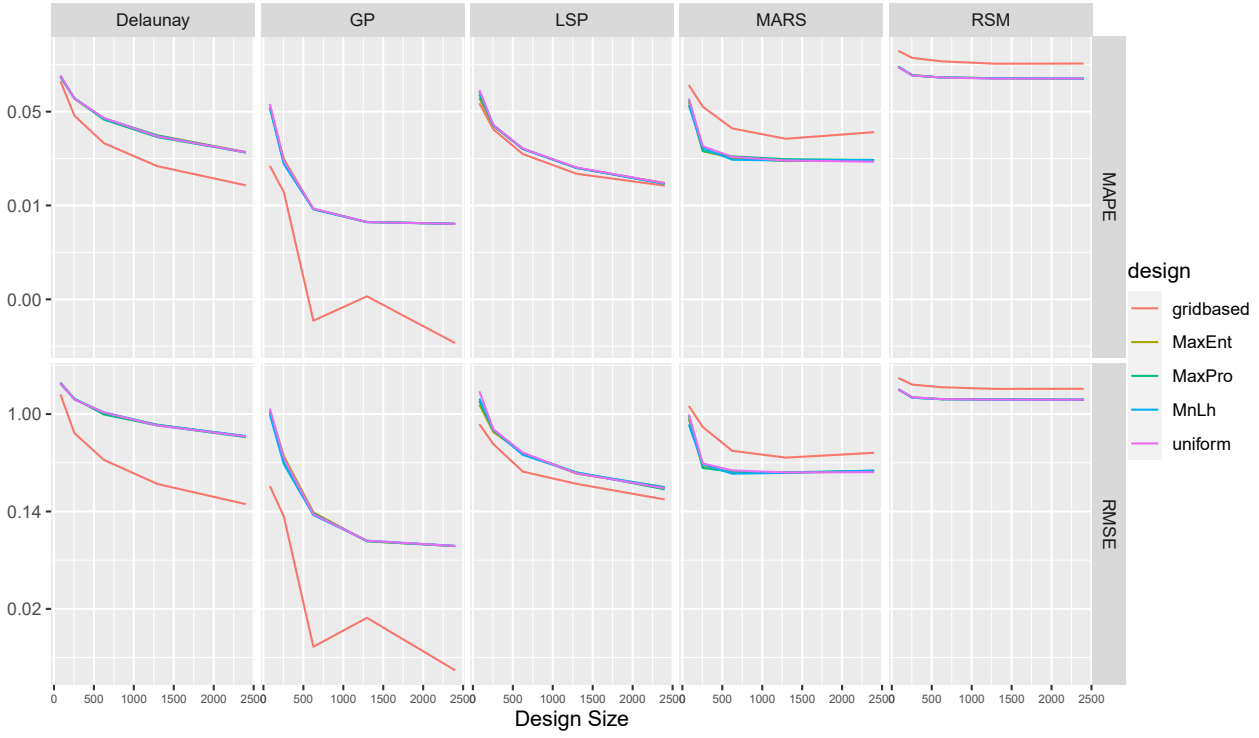


Figure 4: Friedman test function RMSE and MAPE.

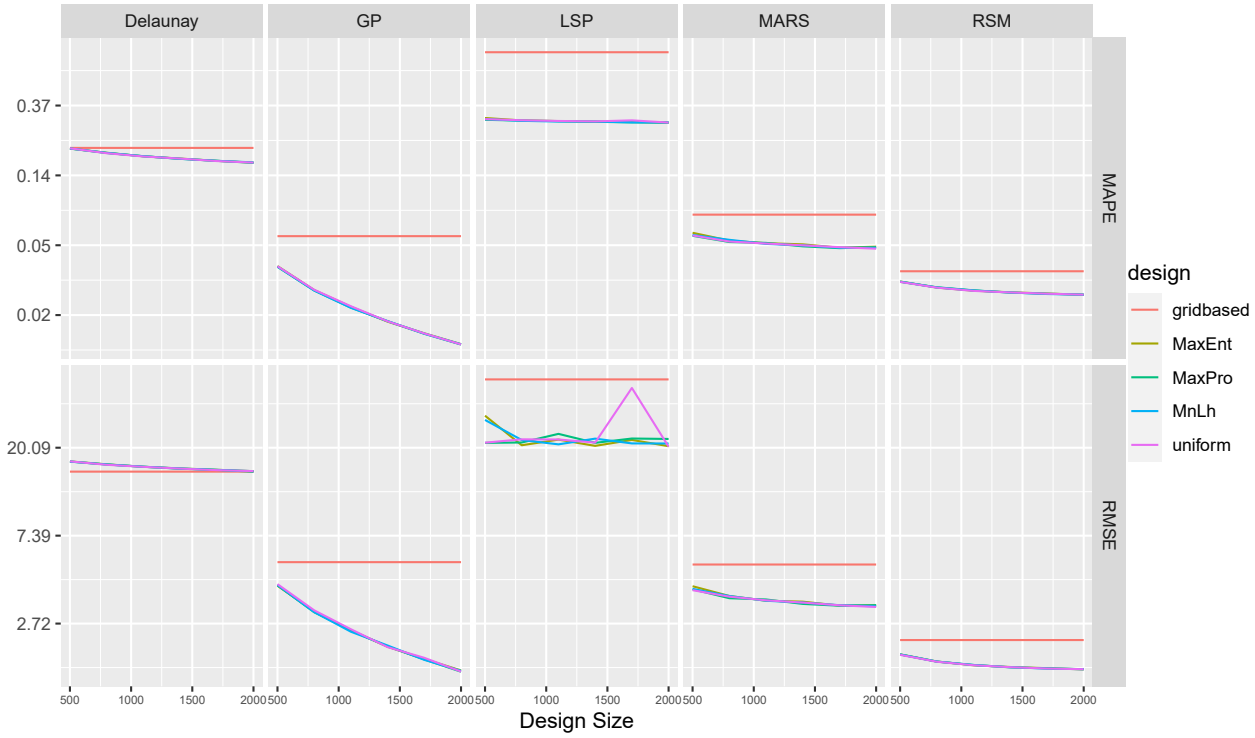


Figure 5: Borehole test function RMSE and MAPE. Here, the GBD has a fixed size of $3^8 = 6561$ due to the computational limit.

GBD, is exactly the real collected PVM. This benefits the GBD since the GBD has the exactly same data we used to build the true surface. When building approximation models to make predictions, the GBD has advantages compare to SFDs. Especially if we use the Delaunay approximation method, we would get 0 prediction error for the GBD. In this way, we can not make a fair comparison for the performance of the GBD and the SFD. If we considering the GP model, although in our analysis, we include nugget effect so the GP model won't go through every training point. The variance-covariance matrix of the fitted GP model trained with the real data is featured by the evenly spaced design points. Potentially, using fitted GP model as the truth also benefit the GBD which has good geometric property. Similarly, LSP model also has the same problem as the Delaunay. Therefore, in order to have a fair comparison, we decided to use the MARS fitted model. It will not return the exactly same PVM for the GBD, also is not affected by the geometric property. So although it's ability to describe the real data is not as good as the above three methods, we still would like to choose it as the truth in the real data analysis.

Comparisons

The results are presented in Figure 6.

Table 4: Average 10-folds CV error.

Model	RSM	MARS	Delaunay	LSP	GP
MAPE	0.47	0.39	0.20	0.26	0.22
RMSE	89476.99	80468.44	63394.17	76488.13	54854.81

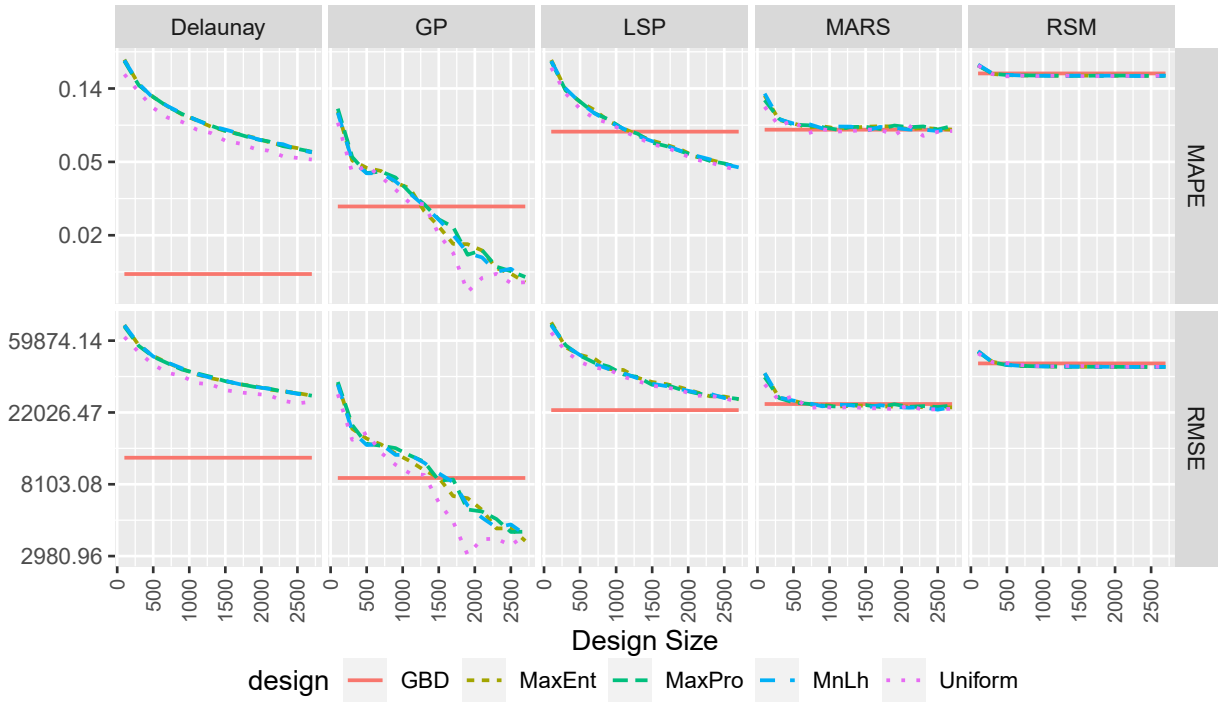


Figure 6: Error trends versus design size in the HPC application. The GBD is given by the locations where true data were collected. Underlying truth surface is fitted by MARS.

Figure 6 describes the MAPE and RMSE of different methods and design combinations when using MARS fitted model as the underlying truth. The error trends of all SFDs are similar to each other under every approximation method. GP with SFDs can achieve the smallest prediction error under both criteria. Like most cases in synthetic analysis, the GBD outperforms SFDs with Delaunay method. While with GP, MARS and RSM models, SFDs can achieve the same prediction error as the GBD with smaller budget. For the LSP method, the SFDs beat the GBD under MAPE criterion and quite close with the GBD under the RMSE criterion.

4 Conclusions and Recommendations

In this paper, comparisons are conducted for synthetic data analysis and HPC variability management application to explore the prediction accuracy of GBDs and SFDs under different approximation method. Overall the prediction error decreases as the design size increases. We find that no design outperforms all others uniformly. GP with SFD generates best results under most scenarios. The GBD outperforms the SFD under the two numerical approximation methods Delaunay and linear Shepard's method for most cases. One possible reason is that GBDs have better geometric properties and these two methods depend on those properties. For the two statistical method, MARS and RSM methods, SFDs have higher prediction accuracy compare to the GBD. One thing interesting with these two statistical method is that, for SFDs, the increasing of design size brings quite small improvement in prediction accuracy after the design size exceed a certain number. I.e., with MARS and RSM method, the prediction accuracy does benefit from the increasing of design size in early stage but after the design size achieve a relatively small budget, increasing design size can not guarantee decreasing in prediction error. While in contrast, under the GP surrogate, increasing the design size of SFDs always results in an obvious decrease in prediction errors.

In previous work, the Delaunay method with GBD tends to have the smallest relative error (Lux et al. 2018). In our analysis , with the maximum design budget, the best approximation method and design combination under each test function and real application is summarized in Table 5. From this table we can see that, GP outperforms other models under both error criteria. This is consistent with the CV prediction ability analysis with the real HPC data in Table 4. Therefore, GP model is recommended when choosing the approximation method. If the data are collected in a grid-based manner, Delaunay method can also be considered.

With the best approximation for each scenario, the design budget of SFDs and GBDs to have the same or higher prediction accuracy are summarized in Table 6. We allow the design

Table 5: Best design and approximation method combination with different test functions under two criteria.

Test Function	RMSE		MAPE	
	Best Method	Best Design	Best Method	Best Design
Colville	GP	SFDs	GP	SFDs
Fried	GP	GBD	GP	GBD
Borehole	GP	SFDs	GP	SFDs
HPC application	GP	SFDs	GP	SFDs

type that have weaker performance to take the maximum design budget in our analysis, and see how many runs the other type need to have the same or higher precision. From Table 6 we can see that by choosing the right design type, we can save a large amount of time and cost. For the cases where SFDs are better, we can use around 50% or less of the design budget for the GBD to achieve the same prediction accuracy. Although we observed that the GBD can also beat SFDs in one case, GBD is not scalable to high-dimensional experiment regions. When the experiment region is of high dimension, building prediction models based on GBD will be time consuming even if we only consider a few unique points in each dimension. So SFDs are recommended when large numbers of input factors need to be considered in the model.

In the future, it will be interesting to investigate designs that can maximize the prediction accuracy based on a good choice of surrogates in the HPC setting. For example, G-optimal designs aim to minimize the maximum element on the diagonal of the hat matrix, which has the effect of minimizing the maximum variance among the predicted values. V-optimal designs minimize the average prediction variance among a set of points. G-/V-optimal designs could be considered because they minimize variance predictions. Another direction is that in our study, we used SFDs for continuous inputs and using a heuristic way to exclude points that do not satisfy the application constraint. A more refined approach that can propose discretized designs that maintain the space filling prosperities and also meet the constraint are desirable to better solve the real application. One further step after obtaining desirable designs can be determining the system configuration that optimize the HPC variabilities. For example, in **?**) work, the optimal system configuration is determined as configuration that can minimize the HPC variability while maintaining the HPC performance, i.e., the computing speed. This can provide insights in choosing system configurations in real HPC applications.

Supplementary Materials

The following supplementary materials are available online.

Table 6: Design budget of the GBD and SFDs to have the same or higher precision under both criterion.

Test Function and Model	RMSE		MAPE	
	SFDs	GBD	SFDs	GBD
Colville with GP	81	2401	256	2401
Fried with GP	2401	625	2401	625
Borehole with GP	500	6561	500	6561
HPC application with GP	1500	2658	1300	2658

Code and data: Computing codes for the synthetic and real data analyses. The HPC data are also included (zip file).

Acknowledgments

The authors acknowledge Advanced Research Computing at Virginia Tech for providing computational resources and technical support that have contributed to the statistical results reported within this paper. The work is supported by funds from NSF under Grant CNS-1565314 and CNS-1838271 to Virginia Tech.

About the Authors

Yueyao Wang is a Ph.D. candidate in Department of Statistics at Virginia Tech. Her email address is yueyao94@vt.edu.

Li Xu is a Ph.D. candidate in Department of Statistics at Virginia Tech. His email address is li1992@vt.edu.

Yili Hong is a Professor of Statistics at Virginia Tech. He is a member of ASQ. His email address is yilihong@vt.edu.

Rong Pan is Associate Professor, Program Chair of Industrial Engineering and Engineering Management at Arizona State University. He is a senior member of ASQ. His email address is Rong.Pan@asu.edu.

Tyler Chang is a Postdoctoral Appointee at Argonne National Laboratory. His email address is tchang@anl.gov.

Thomax Lux is a Senior Research Engineer at Amobee. His email address is tchlux@vt.edu.

Jon Bernard is a Ph.D. student in computer science at Virginia Tech. His email address is jobernar@vt.edu.

Layne Watson is Professor of Computer Science, Mathematics, and Aerospace and Ocean Engineering at Virginia Tech. His email is ltw@cs.vt.edu.

Kirk Cameron is Professor of Computer Science at Virginia Tech and Director of the Center for Computer Systems. His email is cameron@cs.vt.edu.

References

Akkan, H., M. Lang, and L. M. Liebrock (2012). Stepping towards noiseless linux environment. In *Proceedings of the 2nd international workshop on runtime and operating systems for supercomputers*.

Box, G. E. and K. B. Wilson (1951). On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society: Series B (Methodological) 13*(1), 1–38.

Cameron, K. W., A. Anwar, Y. Cheng, L. Xu, B. Li, U. Ananth, J. Bernard, C. Jearls, T. Lux, Y. Hong, L. T. Waston, and A. R. Butt (2019). MOANA: Modeling and analyzing I/O variability in parallel system experimental design. *IEEE Transactions on Parallel and Distributed Systems 30*(8), 1843–1856.

Chang, T. H., L. T. Watson, T. C. Lux, B. Li, L. Xu, A. R. Butt, K. W. Cameron, and Y. Hong (2018). A polynomial time algorithm for multivariate interpolation in arbitrary dimension via the Delaunay triangulation. In *Proceedings of the ACMSE 2018 Conference*, Number 12. ACM.

Currin, C., T. Mitchell, M. Morris, and D. Ylvisaker (1991). Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association 86*(416), 953–963.

Delaunay, B. (1934). Sur la sphère vide. a la mémoire de georges voronoï. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, 793–800.

Fang, K.-T., D. K. Lin, P. Winker, and Y. Zhang (2000). Uniform design: theory and application. *Technometrics 42*(3), 237–248.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics 19*(1), 1–67.

Friedman, J. H., E. Grosse, and W. Stuetzle (1983). Multidimensional additive spline approximation. *SIAM Journal on Scientific and Statistical Computing 4*(2), 291–301.

Giampapa, M., T. Gooding, T. Inglett, and R. W. Wisniewski (2010). Experiences with a lightweight supercomputer kernel: Lessons learned from blue gene's cnk. In *SC'10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE.

Golchi, S. and J. L. Loeppky (2015). Monte Carlo based designs for constrained domains. *arXiv preprint arXiv:1512.07328*.

Goldberg, P. W., C. K. Williams, and C. M. Bishop (1998). Regression with input-dependent noise: A gaussian process treatment. pp. 493–499.

Gramacy, R. B. (2016). laGP: Large-scale spatial modeling via local approximate Gaussian processes in R. *Journal of Statistical Software 72*(1), 1–46. doi: 10.18637/jss.v072.i01.

IOzone projects contributors (2016). IOzone filesystem benchmark. http://www.iozone.org/.

Johnson, M. E., L. M. Moore, and D. Ylvisaker (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference 26*(2), 131–148.

Johnson, R. T., D. C. Montgomery, and J. Bradley (2011). An empirical study of the prediction performance of space-filling designs. *International Journal of Experimental Design and Process Optimisation 2*(1), 1–18.

Joseph, V. R., E. Gul, and S. Ba (2015). Maximum projection designs for computer experiments. *Biometrika 102*(2), 371–380.

Lekivetz, R. and B. Jones (2015). Fast flexible space-filling designs for nonrectangular regions. *Quality and Reliability Engineering International 31*(5), 829–837.

Li, R., D. K. Lin, and Y. Chen (2004). Uniform design: design, analysis and applications. *International Journal of Materials and Product Technology 20*(1-3), 101–114.

Lux, T. C., L. T. Watson, T. H. Chang, J. Bernard, B. Li, L. Xu, G. Back, A. R. Butt, K. W. Cameron, and Y. Hong (2018). Predictive modeling of I/O characteristics in high performance computing systems. In *Proceedings of the High Performance Computing Symposium*, Number 8. Society for Computer Simulation International.

Lux, T. C., L. T. Watson, T. H. Chang, J. Bernard, B. Li, X. Yu, L. Xu, G. Back, A. R. Butt, and K. W. Cameron (2018). Novel meshes for multivariate interpolation and approximation. In *Proceedings of the ACMSE 2018 Conference*, Number 13. ACM.

McKay, M. D., R. J. Beckman, and W. J. Conover (1979). Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics 21*(2), 239–245.

Milborrow, S. (2019). earth: Multivariate adaptive regression splines. R package version 5.1.2.

Morris, M. D. and T. J. Mitchell (1995). Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference 43*(3), 381–402.

Pratola, M. T., O. Harari, D. Bingham, and G. E. Flowers (2017). Design and analysis of experiments on nonconvex regions. *Technometrics 59*(1), 36–47.

Sacks, J., W. J. Welch, T. J. Mitchell, and H. P. Wynn (1989). Design and analysis of computer experiments. *Statistical Science 4*(4), 409–423.

Shepard, D. (1968). A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pp. 517–524. ACM.

Shewry, M. C. and H. P. Wynn (1987). Maximum entropy sampling. *Journal of Applied Statistics 14*(2), 165–170.

Thacker, W. I., J. Zhang, L. T. Watson, J. B. Birch, M. A. Iyer, and M. W. Berry (2010). Algorithm 905: Sheppack: Modified shepard algorithm for interpolation of scattered multivariate data. *ACM Transactions on Mathematical Software (TOMS) 37*(3), 1–20.

Xu, L., Y. Wang, T. C. Lux, T. Chang, J. Bernard, B. Li, Y. Hong, L. T. Watson, and K. W. Cameron (2020). Modeling I/O performance variability in high-performance computing systems using mixture distributions. *Journal of Parallel and Distributed Computing*. doi: 10.1016/j.jpdc.2020.01.005.