Projected Gaussian Markov Improvement Algorithm for High-Dimensional Discrete Optimization via Simulation

XINRU LI, General Motors, Warren, USA
EUNHYE SONG, Georgia Institute of Technology, Atlanta, USA

This article considers a discrete optimization via simulation (DOvS) problem defined on a graph embedded in the high-dimensional integer grid. Several DOvS algorithms that model the responses at the solutions as a realization of a Gaussian Markov random field (GMRF) have been proposed exploiting its inferential power and computational benefits. However, the computational cost of inference increases exponentially in dimension. We propose the projected Gaussian Markov improvement algorithm (pGMIA), which projects the solution space onto a lower-dimensional space creating the region-layer graph to reduce the cost of inference. Each node on the region-layer graph can be mapped to a set of solutions projected to the node; these solutions form a lower-dimensional solution-layer graph. We define the response at each region-layer node to be the average of the responses within the corresponding solution-layer graph. From this relation, we derive the region-layer GMRF to model the region-layer responses. The pGMIA alternates between the two layers to make a sampling decision at each iteration. It first selects a region-layer node based on the lower-resolution inference provided by the region-layer GMRF, then makes a sampling decision among the solutions within the solution-layer graph of the node based on the higher-resolution inference from the solution-layer GMRF. To solve even higher-dimensional problems (e.g., 100 dimensions), we also propose the pGMIA+: a multi-layer extension of the pGMIA. We show that both pGMIA and pGMIA+ converge to the optimum almost surely asymptotically and empirically demonstrate their competitiveness against state-of-the-art high-dimensional Bayesian optimization algorithms.

CCS Concepts: • **Theory of computation → Discrete optimization**; **Bayesian analysis**; • **Computing methodologies → Modeling and simulation;**

Additional Key Words and Phrases: Gaussian Markov random field, high-dimensional discrete optimization via simulation, projection, Bayesian optimization

ACM Reference Format:
Xinru Li and Eunhye Song. 2024. Projected Gaussian Markov Improvement Algorithm for High-Dimensional Discrete Optimization via Simulation. *ACM Trans. Model. Comput. Simul.* 34, 3, Article 14 (May 2024), 29 pages. https://doi.org/10.1145/3649463

1 INTRODUCTION

Optimization via simulation (OvS) refers to the class of methodologies for optimizing a problem whose objective function and/or constraints at a feasible solution must be estimated via stochastic simulation. When an OvS problem has a discrete solution space, the problem is further categorized

This research is supported by the National Science Foundation under grants no. DMS-1854659 and no. CMMI-2045400.
Authors' addresses: X. Li, General Motors, 30500 Mound Rd, Warren, MI, 48092; email: xinru.li@gm.com; E. Song, Georgia Institute of Technology, 755 Ferst Drive, Atlanta, GA; e-mail: eunhye.song@isye.gatech.edu.

ACM 1558-1195/2024/05-ART14
https://doi.org/10.1145/3649463

as a **discrete OvS (DOvS)** problem. This article focuses on a DOvS problem whose feasible solution space is a high-dimensional hyperbox defined on the integer lattice. To motivate our research, we illustrate two practical applications below.

(i) Cevik et al. [2016] study calibration of the **University of Wisconsin breast cancer simulation (UWBCS)** model. The input parameters of the UWBCS model include breast cancer natural history, detection, treatment parameters, and non-breast cancer mortality parameters. Calibration targets include the cancer incidence and mortality rates. Each combination of parameters is assigned a score that measures the difference between the simulation results and the targets. The UWBCS model has ten natural history parameters that amount to 378,000 feasible combinations in total.

(ii) Hoffman et al. [2018] study condition-based maintenance scheduling for a manufacturing line. Each machine's state is represented by an integer-valued health index, which stochastically degrades over time. The vector of decision variables consists of a health index threshold at which maintenance is to be scheduled for each machine. Maintenance resources are shared across all machines. The objective is to minimize the expected cost of operation. The problem dimension is determined by the number of machines, which may be large for a complex manufacturing system.

In both examples, each solution can only be evaluated via stochastic simulation. The stochastic error variance of simulation output tends to be large and the simulation runtime is nontrivial. Both examples have high-dimensional discrete solution spaces that can be embedded in the integer lattice, where the number of feasible solutions increases exponentially in the problem dimension. As such, it is difficult to apply a DOvS algorithm that requires all solutions to be simulated (e.g., ranking and selection). Meanwhile, it is reasonable to assume that two solutions that are close in the integer lattice are likely to have similar responses, e.g., the expected operating costs of two maintenance policies are similar if their health index thresholds are close. Therefore, a DOvS method that exploits *spatial inference* may significantly improve efficiency in solving these types of problems.

There are several existing inferential DOvS algorithms in which the responses at the feasible solutions are modeled as a realization of a stochastic process. Taking the Bayesian view, a prior distribution is assumed for the stochastic process, which is then updated to the posterior as the algorithm simulates solutions. The updated posterior distribution provides inference on the responses at any feasible solutions—including those that have not yet been simulated—and the algorithm selects which solution to simulate next by evaluating a sampling criterion. A popular choice for the prior is a **Gaussian process (GP)** as its posterior update is convenient; combined with the assumption that the simulation output has Gaussian noise, the posterior is still a GP. Inferential OvS is closely related to **Bayesian Optimization (BO)**, which has become popular for global optimization of a black-box function. BO also adopts GP as a workhorse model while mainly focusing on continuous-domain problems with deterministic objective functions.

Compared with the continuous counterpart, the literature on inferential DOvS (or discrete BO) is rather slim. Baptista and Poloczek [2018] focus on the binary variables while Oh et al. [2019] and Roustant et al. [2020] consider a combinatorial solution space with categorical variables. Mes et al. [2011] also study categorical solutions and have commonality with this work in that they build hierarchical Bayesian models to represent the solution space with decreasing resolution. However, they do not model spatial correlation among solutions and their target problem is much smaller scale (thousands of solutions) than what is considered here. Closer to our work are Sun et al. [2014] and Xie et al. [2016], which study integer solution spaces embedded in the Euclidean space and adopt a continuous GP covariance kernel to model the spatial correlation among the discrete solutions. In the same setting, Garrido-Merchán and Hernández-Lobato [2020] discuss a way to improve the covariance kernel's inference on the discrete solution space via rounding.

Salemi et al. [2019] empirically demonstrate that some popular choices of the covariance kernel may result in poor inference when applied to solutions on the integer lattice. Instead, they model the responses at the solutions on the d-dimensional integer lattice as a realization of a **Gaussian Markov random field (GMRF)** and propose the **Gaussian Markov improvement algorithm (GMIA)**. A GMRF is a GP defined on a graph of nodes, in which the connectivity between nodes on the graph determines the correlation structure of the GMRF. Its precision matrix (inverse of the covariance matrix) has nonzero elements if the corresponding nodes are connected by an edge on the graph. Thus, the precision matrix tends to be sparse when the graph is sparse and updating the precision matrix is extremely cheap. However, to evaluate a sampling criterion at solutions, the posterior variances of the solutions are needed. Even if the precision matrix is sparse, computing its inverse is expensive for a large-scale DOvS problem; the cost of matrix operations involved in sampling criterion evaluation increases at least quadratically in the number of feasible solutions.

To reduce the computational complexity of GMIA, improvements have been proposed. Semelhago et al. [2017] adopt sparse linear algebra techniques to significantly reduce the cost of precision matrix inversion. For further speed-up, Semelhago et al. [2021] restrict the sampling decisions within a promising subset of solutions for several rapid-search iterations. In contrast, Li and Song [2020] achieve the same order of computational cost as in Semelhago et al. [2021] while making global inference by exploiting the linear algebraic techniques tailored for sparse matrices. However, all of these approaches require inverting a precision matrix at least intermittently. As the problem dimension increases, the GMIA eventually hits the limit on the size of the precision matrix that can be stored and inverted.

Reducing the computational overhead of a high-dimensional BO problem has been actively researched in recent years. These approaches can be categorized into two groups as summarized below. For a more comprehensive review on high-dimensional BO, see Binois and Wycoff [2022].

The first direction is *batching* the dimensions into groups and constructing an additive GP model from the batches. Kandasamy et al. [2015] decompose the solution space into disjoint groups of dimensions and represent the objective function as the sum of independent GPs defined for each group. They apply the sampling criterion in each group separately, then aggregate them to decide the next solution to sample. Wang et al. [2017] propose an adaptive decomposition of dimensions using a multinomial distribution whose posterior is updated throughout the algorithm. Rolland et al. [2018] and Mutny and Krause [2018] extend the idea to overlapping dimension groups. Wang et al. [2018] partition the solution space using a Mondrian process and learn a local GP for each subspace and the additive structure.

The second approach is to reduce the problem dimension by *projection*, assuming that the black-box function has a lower-dimensional active subspace. Wang et al. [2013] first apply the projection idea to high-dimensional BO using a random projection matrix. On the other hand, Djolonga et al. [2013] apply a low-rank recovery algorithm to compute the projection matrix from an initial design. However, when the feasible solution space is bounded, the projection-based method does not perform well when a substantial chunk of the embedded subspace is projected to outside of the feasible solution space of the original problem. Letham et al. [2020] and Binois et al. [2020] each discuss lower-dimensional embedding approaches that avoid this phenomenon. Eriksson and Jankowiak [2021] only consider a sparse axis-aligned subspace, arguing that inferring a non-axis-aligned subspace is substantially more expensive and leads to overfitting. Lu et al. [2018] and Moriconi et al. [2020] consider nonlinear embedding through **variational autoencoders (VAEs)**.

Both batching and projection approaches have limitations. In batching, the objective function is assumed to be a sum of functions of the batched dimensions. Therefore, unless two dimensions are in the same batch, their interaction effects cannot be modeled. Moreover, the additive model's covariance can be constructed easily from the batch GPs. However, for a GMRF, spatial correlation

is implicitly determined by the precision matrix and it is not straightforward to construct the precision matrix of the additive model from the batch GMRFs. For the projection methods, it is typically assumed that an upper bound of the active subspace's dimension is known, but without prior knowledge on the objective function, such an upper bound is difficult to obtain. Mathesen et al. [2019] point out that projection-based BO algorithms may perform poorly when a lower-dimensional active subspace does not exist for the given problem; the same can be observed in our numerical experiments.

In this article, we propose the ***projected* Gaussian Markov Improvement Algorithm (pGMIA)** and pGMIA+. The former establishes a framework for the GMIA to scale up in the problem dimension, whereas the latter extends the framework to solve significantly higher-dimensional problems than the former. The pGMIA and pGMIA+ reap the benefits of both batching and projection approaches while tackling their limitations.

The pGMIA partitions the dimensions into two batches, *solution-layer* and *region-layer* dimensions, then projects the solution space onto the latter. Since this is an axis-aligned projection, the integer lattice (graph) structure of the solution space is preserved after projection. The name *region-layer* comes from the fact that a node in the projected graph can be mapped to a region of solutions in the original solution space. We refer to the projected graph and its node as *region-layer graph* and *region-layer node*, respectively. Inversely, each region-layer node corresponds to a *solution-layer graph* that contains all solutions projected to the node. Note that only the solution-layer dimensions are represented in the solution-layer graph, which makes it lower dimensional than the original solution space.

There are two major differences between the pGMIA and other projection methods. First, the pGMIA does not assume an active lower-dimensional subspace. Instead, it regards the response at each region-layer node to be the average of the responses at all solutions in the corresponding solution-layer graph. Second, unlike other methods that make sampling decisions only in the projected space, the pGMIA makes hierarchical sampling decisions in both region and solution layers. The pGMIA first selects a region-layer node, then zooms into its solution-layer graph to select a solution to sample. Essentially, *the pGMIA treats all dimensions to be active*, not just the region-layer dimensions.

As in the batching approaches, we model both region-layer and solution-layer responses with GMRFs; however, they do not fit an independent model for each batch of dimensions. Instead, we derive both region- and solution-layer GMRFs from a single GMRF representing the entire solution space. We refer to the last as the *single-layer* GMRF to distinguish it from the other two. One region-layer GMRF is defined to represent the region-layer graph's responses, and for each region-layer node, a solution-layer GMRF is defined for the corresponding solution-layer graph.

Unlike the batching methods, the pGMIA does not assume the objective function to be a sum of lower-dimensional functions. Rather, our model has a hierarchical structure—the region-layer GMRF captures the lower-resolution trend in the response, whereas each solution-layer GMRF provides a local, higher-resolution model. In addition, the pGMIA periodically updates the batches according to a user-chosen criterion. This allows re-batching of the dimensions and gives the pGMIA an opportunity to learn all interaction terms in the response as the simulation budget increases.

The pGMIA+ extends the two-layer scheme of the pGMIA to multi-layers; it partitions the dimensions into $m > 2$ batches and constructs m layers by hierarchically projecting the batches of dimensions. Starting from the top layer, the pGMIA+ selects a node on each layer's graph and zooms into the lower-layer graph to make a sampling decision. As m increases, fewer dimensions are included in each layer, reducing the computational overhead of the sampling decisions. This allows the pGMIA+ to tackle much higher–dimensional problems than the pGMIA.

The idea of hierarchical search between the two layers of GMRFs has also been explored by Salemi et al. [2019] in their algorithm, **multi-resolution GMIA (MR-GMIA)**. However, the MR-GMIA has several theoretical/practical limitations. First, their region-layer GMRF is fitted independently from the solution-layer GMRFs for computational convenience, although there is clear dependence due to averaging of the responses. Second, it is assumed that the precision matrix of the region-layer GMRF has the same sparsity pattern as that of the solution-layer GMRF whereas, in truth, the region-layer precision matrix becomes a dense matrix due to averaging. Moreover, the batches of dimensions as well as the hyperparameters of the GMRFs remain unchanged after initialization throughout the algorithm, which significantly slows down the search progress in later iterations, as our empirical study shows.

Unlike the MR-GMIA, the pGMIA's region- and solution-layer GMRFs are statistically consistent; the region-layer GMRF is defined by projecting the single-layer GMRF. This makes the region-layer precision matrix dense; however, we prove that the dense precision matrix can be effectively approximated with an easy-to-compute sparse matrix. We derive an exact bound on the approximation error and show that it becomes negligible as the feasible range in each dimension increases. Moreover, the resulting approximate precision matrix has the same sparsity pattern as the single-layer GMRF, what MR-GMIA assumes without mathematical justification. As such, efficient linear algebraic techniques tailored for the single-layer GMRF can still be applied at the region layer. Moreover, exploiting the Markov property of the single-layer model, we show that each solution-layer GMRF is independent from the rest of the field conditional on the fact that the responses at the neighboring solutions of the region are equal to the single-layer prior means. This approximation significantly reduces the computational overhead of the solution-layer sampling decisions as we can focus on the solutions included in the selected region-layer node while ignoring the rest of the solution space.

Another benefit of the pGMIA is that the hyperparameters of the GMRFs can be updated cheaply. Under our region-layer GMRF approximation, the region-layer hyperparameters can be written as linear functions of the single-layer GMRF parameters. Thus, instead of computing the **maximum likelihood estimator (MLE)** of the single-layer GMRF, the pGMIA first solves the region-layer MLE problem and then solves the solution-layer problem conditional on the region-layer estimates. Since each layer contains fewer dimensions than the original space, the computational overhead is significantly reduced. Thanks to this feature, the pGMIA can quickly update the hyperparameters whenever the batch of dimensions is updated.

Under mild assumptions, we prove that both pGMIA and pGMIA+ converge to a global optimum of the DOvS problem almost surely when run without stopping. We compare our algorithms with MR-GMIA as well as some state-of-the-art high-dimensional BO algorithms. The empirical results demonstrate competitiveness of the pGMIA and the pGMIA+ in both search progress and computation time.

The remainder of the article is structured as follows. In Section 2, we provide background on GMRFs and discuss the projected GMRF model. Section 3 discusses the hyperparameter estimation at both region and solution layers. Section 4 provides the algorithmic details of the pGMIA and the pGMIA+. An extensive empirical study is presented in Section 5 to demonstrate the performances of the pGMIA and the pGMIA+, followed by conclusions in Section 6. Due to the space limit, all appendices are included in the Supplementary Material.

2 GAUSSIAN MARKOV RANDOM FIELDS FOR DOVS

The DOvS problem of our interest is $\min_{\mathbf{x}\in\mathscr{X}} y(\mathbf{x}) \triangleq \mathrm{E}[Y(\mathbf{x})]$, where the feasible solution space, $\mathscr{X}$, is a finite subset of d-dimensional integer lattice and $Y(\mathbf{x})$ is the stochastic simulation output at feasible solution $\mathbf{x}$. For any $\mathbf{x} \in \mathscr{X}$, $Y_j(\mathbf{x}) = y(\mathbf{x}) + \epsilon_j(\mathbf{x})$ can be observed for replications $j = 1, 2, \ldots,$

where simulation error $\epsilon_j(\mathbf{x})$ is **independent and identically distributed (i.i.d.)** with zero mean and unknown variance $\sigma^2(\mathbf{x})$.

We further assume $\mathscr{X}$ to be a hyperbox. The number of feasible solutions, $n \triangleq |\mathscr{X}|$, increases exponentially in d in this setting. When n is large and simulation runtime is nonnegligible, only a small fraction of $\mathscr{X}$ can be simulated. To make inference at the solutions not simulated yet, we model $y(\cdot)$ as a realization of a GMRF. In Section 2.1, we introduce the single-layer GMRF, then define the region- and solution-layer GMRFs in Section 2.2.

2.1 Single-layer GMRF

A GMRF is a multivariate Gaussian random vector $\mathbb{Y} = (\mathbb{Y}_1, \mathbb{Y}_2, \ldots, \mathbb{Y}_n)^\top$ defined on undirected labeled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of n nodes and $\mathcal{E}$ is the set of edges connecting neighboring nodes. In the context of DOvS, $\mathcal{V}$ corresponds to $\mathscr{X}$, and each $\mathbb{Y}_i$ models the value of $y(\cdot)$ at the i-th feasible solution. We refer to $\mathbb{Y}$ as the single-layer GMRF model. We also adopt $\mathbb{Y}(\mathbf{x})$ to denote the element of $\mathbb{Y}$ corresponding to solution $\mathbf{x}$ to make the dependence explicit.

The prior imposed on $\mathbb{Y}$ is $f(\mathbb{Y}|\boldsymbol{\mu}, \mathbf{Q}) = (2\pi)^{-n/2}|\mathbf{Q}|^{1/2}\exp(-\frac{1}{2}(\mathbb{Y}-\boldsymbol{\mu})^\top\mathbf{Q}(\mathbb{Y}-\boldsymbol{\mu}))$, where $\boldsymbol{\mu}$ and $\mathbf{Q}$ are the mean vector and precision matrix, respectively. The covariance matrix of $\mathbb{Y}$ is $\boldsymbol{\Sigma} \triangleq \mathbf{Q}^{-1}$. Let Q_{ij} be the (i,j)-th element of $\mathbf{Q}$. Then, $\mathrm{Q}_{ii} = \mathrm{Prec}(\mathbb{Y}_i|\mathbb{Y}_{\mathcal{V}\setminus\{i\}}) = 1/\mathrm{Var}(\mathbb{Y}_i|\mathbb{Y}_{\mathcal{V}\setminus\{i\}})$ and

$$\mathrm{Q}_{ij} = -\mathrm{Corr}(\mathbb{Y}_i, \mathbb{Y}_j|\mathbb{Y}_{\mathcal{V}\setminus\{i,j\}})\sqrt{\mathrm{Q}_{ii}\mathrm{Q}_{jj}}, \quad i \neq j, \tag{1}$$

where $\mathbb{Y}_S$ is the subvector of $\mathbb{Y}$ corresponding to the nodes in $S \subset \mathcal{V}$. Let $\mathscr{N}(i)$ be the set of the neighbors of Node i in $\mathcal{G}$. Then, for every $i \in \mathcal{V}$, we have that $\mathbb{Y}_i \perp \mathbb{Y}_{\mathcal{V}\setminus(\mathscr{N}(i)\cup\{i\})}|\mathbb{Y}_{\mathscr{N}(i)}$ [Rue and Held 2005]. Namely, conditional on the responses at its neighbors, the response at each node is independent of the rest of the field, which makes $\mathbb{Y}$ Markovian. Combining this property with Equation (1), we have that $\mathrm{Q}_{ij} = 0$, if $j \notin \mathscr{N}(i)$. Thus, the sparsity of $\mathbf{Q}$ is determined by $\mathscr{N}(\cdot)$. As in Salemi et al. [2019], we define the set of neighbors of $\mathbf{x} \in \mathscr{X}$ as $\mathscr{N}(\mathbf{x}) = \{\mathbf{x}' \in \mathscr{X} : \|\mathbf{x}-\mathbf{x}'\|_2 = 1\}$, where $\|\cdot\|_2$ is the Euclidean norm. This makes $\mathbf{Q}$ very sparse, with the fill-in rate (the fraction of nonzero elements) less than $(2d+1)/n$.

We assume that $\boldsymbol{\mu} = \beta\mathbf{1}_n$, where β is a hyperparameter and $\mathbf{1}_n$ is the n-dimensional column vector of ones assuming that there is no prior information about the objective function values at the solutions. The precision matrix, $\mathbf{Q}$, is parameterized by $\boldsymbol{\theta} = (\theta_0, \theta_1, \ldots, \theta_d)$, whose (i,j)-th entry is

$$\mathrm{Q}_{ij} \triangleq \begin{cases} \theta_0, & \text{if } i = j, \\ -\theta_0\theta_\ell, & \text{if } |\mathbf{x}_i - \mathbf{x}_j|_{\mathrm{abs}} = \mathbf{e}_\ell, \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where $\mathbf{e}_\ell$ is the ℓth standard basis vector and $|\cdot|_{\mathrm{abs}}$ is the element-wise absolute value operator. From Equations (1) and (2), observe that $\theta_\ell = \mathrm{Corr}(\mathbb{Y}_i, \mathbb{Y}_j|\mathbb{Y}_{\mathcal{V}\setminus\{i,j\}})$ if $|\mathbf{x}_i - \mathbf{x}_j|_{\mathrm{abs}} = \mathbf{e}_\ell$, i.e., θ_ℓ determines the prior conditional correlation between neighboring solutions in the ℓth dimension. The conditional precision of any solution must be positive; thus, $\theta_0 = \mathrm{Q}_{ii} > 0$. We assume that the prior conditional correlation between two solutions is nonnegative, i.e., $\theta_\ell \geq 0$ for all $1 \leq \ell \leq d$. Moreover, we impose that $\mathbf{Q}$ is diagonally dominant, i.e., $\sum_{\ell=1}^{d}\theta_\ell < 0.5$, to make $\mathbf{Q}$ positive definite.

Since $y(\mathbf{x})$ cannot be evaluated directly, we do not observe the realization of $\mathbb{Y}(\mathbf{x})$. Instead, we define a new GMRF that models the stochastic simulation output at $\mathbf{x}$. Let $\mathscr{X}_2 \subset \mathscr{X}$ be the set of simulated solutions and $\mathscr{X}_1 \triangleq \mathscr{X}\setminus\mathscr{X}_2$. Let $\mathbb{Y}_1$ and $\mathbb{Y}_2$ be the subvectors of $\mathbb{Y}$ corresponding to $\mathscr{X}_1$ and $\mathscr{X}_2$, respectively. For $\mathbf{x} \in \mathscr{X}_2$, we observe that $\bar{Y}(\mathbf{x}) = \frac{1}{r(\mathbf{x})}\sum_{j=1}^{r(\mathbf{x})} Y_j(\mathbf{x})$, where $r(\mathbf{x})$ is the number of replications made at $\mathbf{x}$. Let $\mathcal{Y}_2$ be the vector of $\bar{Y}(\mathbf{x})$ at all $\mathbf{x} \in \mathscr{X}_2$; then, $\mathcal{Y}_2$ is a realization of the GMRF, $\mathbb{Y}_2^\epsilon = \mathbb{Y}_2 + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}|\mathbb{Y}_2 \sim \mathcal{N}(\mathbf{0}_{|\mathscr{X}_2|}, \mathbf{Q}_\epsilon^{-1})$, $\mathbf{0}_{|\mathscr{X}_2|}$ is a $|\mathscr{X}_2|$-dimensional

column vector of zeros, and $\mathbf{Q}_\epsilon$ is a diagonal matrix whose diagonal element corresponding to $\mathbf{x}$ is $r(\mathbf{x})/\sigma^2(\mathbf{x})$. Because $\sigma^2(\mathbf{x})$ is unknown, we adopt $S^2(\mathbf{x}) = \frac{1}{r(\mathbf{x})-1}\sum_{j=1}^{r(\mathbf{x})}\left(Y_j(\mathbf{x}) - \bar{Y}(\mathbf{x})\right)^2$ as a plug-in estimate. Salemi et al. [2019] show that

$$\mathbb{Y}|\mathbb{Y}_2^\epsilon = \mathcal{Y}_2 \sim \mathcal{N}\left(\beta\mathbf{1}_n + \bar{\mathbf{Q}}^{-1}\begin{pmatrix}\mathbf{0}_{|\mathscr{X}_1|}\\ \mathbf{Q}_\epsilon(\mathcal{Y}_2 - \beta\mathbf{1}_{|\mathscr{X}_2|})\end{pmatrix}, \bar{\mathbf{Q}}^{-1}\right), \text{ where } \bar{\mathbf{Q}} = \begin{bmatrix}\mathbf{Q}_{11} & \mathbf{Q}_{12}\\ \mathbf{Q}_{12}^\top & \mathbf{Q}_{22}+\mathbf{Q}_\epsilon\end{bmatrix}, \quad (3)$$

with $\mathbf{Q}_{ij}$ representing the submatrix of $\mathbf{Q}$ corresponding to $\mathscr{X}_i$ and $\mathscr{X}_j$ for $1 \le i, j \le 2$. Updating $\bar{\mathbf{Q}}$ after simulating a new solution is extremely cheap, as we only need to update the corresponding diagonal element of $\mathbf{Q}_\epsilon$. Moreover, the sparsity pattern of $\mathbf{Q}$ is preserved in $\bar{\mathbf{Q}}$.

We adopt the complete expected improvement (CEI) proposed by Salemi et al. [2019] as the sampling criterion. Let $\tilde{\mathbf{x}}$ be the sample-best solution, i.e., $\tilde{\mathbf{x}} = \operatorname{argmin}_{\mathbf{x}\in\mathscr{X}_2}\bar{Y}(\mathbf{x})$; at any point in the algorithm, we refer to $\tilde{\mathbf{x}}$ as the current best solution. Then, the CEI of $\mathbf{x}$ relative to $\tilde{\mathbf{x}}$ is defined as $\mathrm{CEI}(\tilde{\mathbf{x}}, \mathbf{x}) = \mathrm{E}[\max(\mathbb{Y}(\tilde{\mathbf{x}}) - \mathbb{Y}(\mathbf{x}), 0)]$, where the expectation is with respect to the posterior distribution in (3). Let $\mathbf{M}(\mathbf{x})$ and $\mathbf{V}(\mathbf{x})$ denote the posterior mean and variance of $\mathbb{Y}(\mathbf{x})$, respectively, and let $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x})$ be the posterior covariance between $\mathbb{Y}(\mathbf{x})$ and $\mathbb{Y}(\tilde{\mathbf{x}})$. Note that $\mathbf{M}(\mathbf{x})$ is an element of the mean vector of (3) and $\mathbf{V}(\mathbf{x})$ and $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x})$ can be obtained from the posterior covariance matrix, $\bar{\mathbf{Q}}^{-1}$. Then, the posterior variance of $\mathbb{Y}(\tilde{\mathbf{x}}) - \mathbb{Y}(\mathbf{x})$ is $\mathbf{V}(\tilde{\mathbf{x}}, \mathbf{x}) = \mathbf{V}(\tilde{\mathbf{x}}) + \mathbf{V}(\mathbf{x}) - 2\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x})$. The CEI of $\mathbf{x}$ can be computed as [Jones et al. 1998]

$$\mathrm{CEI}(\tilde{\mathbf{x}}, \mathbf{x}) = (\mathbf{M}(\tilde{\mathbf{x}}) - \mathbf{M}(\mathbf{x}))\Phi\left(\frac{\mathbf{M}(\tilde{\mathbf{x}})-\mathbf{M}(\mathbf{x})}{\sqrt{\mathbf{V}(\tilde{\mathbf{x}},\mathbf{x})}}\right) + \sqrt{\mathbf{V}(\tilde{\mathbf{x}}, \mathbf{x})}\phi\left(\frac{\mathbf{M}(\tilde{\mathbf{x}})-\mathbf{M}(\mathbf{x})}{\sqrt{\mathbf{V}(\tilde{\mathbf{x}},\mathbf{x})}}\right), \quad (4)$$

where ϕ and Φ are the standard normal probability density and cumulative distribution functions, respectively. Therefore, to compute CEI at all solutions, we need the posterior mean vector and $2n-1$ elements of $\bar{\mathbf{Q}}^{-1}$: namely, its diagonal and the column corresponding to $\tilde{\mathbf{x}}$. This observation combined with the sparsity of $\bar{\mathbf{Q}}$ has led to a series of computational improvements of GMIA [Li and Song 2020; Semelhago et al. 2021, 2017], as reviewed in Section 1. However, they eventually hit computational limits as n and d increase. The memory space required to store the precision matrix is another challenge. For instance, MATLAB requires more than 4.46e+12 GB of RAM to store $\bar{\mathbf{Q}}$ with $n = 10^{20}$.

2.2 Projected GMRF

As mentioned in Section 1, the pGMIA batches the dimensions of $\mathscr{X}$ into two groups, the region-layer and the solution-layer dimensions, then projects the solution space onto the region-layer dimensions to create the region-layer graph. We illustrate the projection scheme with a simple two-dimensional example in Figure 1, where the first dimension has coordinate values from 1 to 4 and the second from 5 to 8. Suppose we chose the second dimension to be the region-layer dimension. By projecting the solution space onto the second, we create a region-layer graph with four nodes (solid circles), where each node corresponds to four solutions in each dashed box. Note that all solutions in each dashed box have the same second coordinate since they are projected to the same node in the region-layer graph. The corresponding solution-layer graph defined for each dashed box is therefore one-dimensional, with only the first dimension active. In total, the projection creates one region-layer graph and four solution-layer graphs.

The pGMIA defines the response at each region-layer node as the average of the responses at the solutions in the solution-layer graph corresponding to the node. Just as the single-layer GMRF models the responses at all solutions, the pGMIA defines a region-layer GMRF to model the vector of region-layer responses and a solution-layer GMRF to model the solutions' responses included in each solution-layer graph. For the example in Figure 1, one region-layer GMRF and four solution-

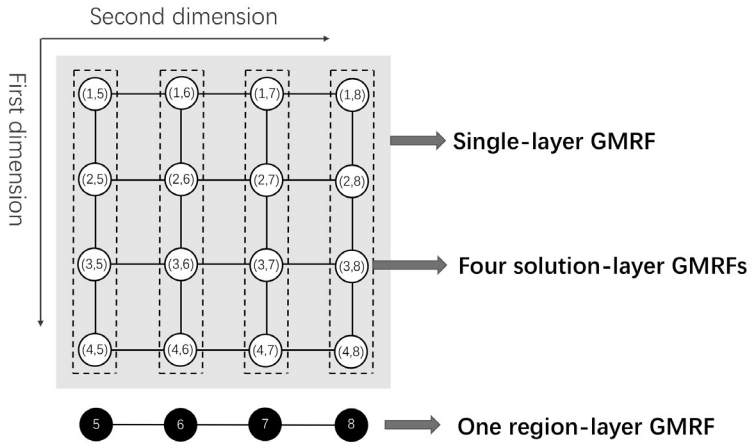


Fig. 1. Illustration of the solution-layer and region-layer GMRFs for a simple 2-dimensional example. The first dimension is included in the solution layer, whereas the second dimension is in the region layer.

layer GMRFs can be defined. In the remainder of this section, we mathematically formalize how the region- and solution-layer GMRFs are derived from the single-layer GMRF.

Let k_i denote the number of feasible values that the i-th coordinate of $\mathbf{x}$ can take. Consequently, we have that $n = \prod_{i=1}^{d} k_i$. Let d_2 represent the number of region-layer dimensions and $d_1 \triangleq d - d_2$. Without loss of generality, we label the dimensions in the region layer as $i = d_1 + 1, d_1 + 2, \ldots, d$. The region layer graph has $K_r = \prod_{i=d_1+1}^{d} k_i$ nodes and $K_s = \prod_{i=1}^{d_1} k_i$ solutions are projected to each region-layer node. Clearly, $K_r \ll n$. Let $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_{K_r}$ denote the region-layer nodes; then, $\mathscr{X} = \bigcup_{\ell=1}^{K_r} \mathscr{X}(\mathcal{R}_\ell)$, where $\mathscr{X}(\mathcal{R}_\ell)$ is the set of solutions in the solution-layer graph of $\mathcal{R}_\ell$.

The pGMIA allows the projection to be updated throughout the algorithm. Let $\mathscr{P}$ denote the partition of dimensions that determines the projection. Given d_1, there are $\binom{d}{d_1}$ candidates for $\mathscr{P}$. Each time $\mathscr{P}$ is updated, the dimensions are relabeled so that the first d_1 dimensions always belong in the solution layer. Other quantities such as K_r and K_s that depend on $\mathscr{P}$ are also updated accordingly. The pGMIA updates $\mathscr{P}$ with mild computational overhead. In the remainder of this section and Section 3, we assume $\mathscr{P}$ to be fixed and defer the discussion on its update to Section 4.

For each region-layer node R_ℓ, its response is defined as $z(\mathcal{R}_\ell) \triangleq K_s^{-1} \sum_{\mathbf{x} \in \mathscr{X}(\mathcal{R}_\ell)} y(\mathbf{x})$. The pGMIA models $(z(\mathcal{R}_1), z(\mathcal{R}_2), \ldots, z(\mathcal{R}_{K_r}))^\top$ as a realization of the region-layer GMRF, $\mathbb{Z} \triangleq (\mathbb{Z}(\mathcal{R}_1), \mathbb{Z}(\mathcal{R}_2), \ldots, \mathbb{Z}(\mathcal{R}_{K_r}))^\top$. To derive the distribution of $\mathbb{Z}$ from that of the single-layer GMRF, $\mathbb{Y}$, let us first define $K_r \times n$ matrix

$$\mathbf{P} \triangleq \mathbf{I}_{K_r} \bigotimes \mathbf{1}_{K_s}^\top, \tag{5}$$

where $\mathbf{I}_{K_r}$ is the $K_r \times K_r$ identity matrix and $\bigotimes$ represents the Kronecker product. Then, we have that $\mathbb{Z} = K_s^{-1}\mathbf{P}\mathbb{Y}$. We assume that the elements of $\mathbb{Y}$ are sorted such that the corresponding solutions are in ascending order of coordinates from the first to last dimensions. For instance, in Figure 1, the first five elements of $\mathbb{Y}$ correspond to Solutions $(1,5), (2,5), (3,5), (4,5)$, and $(1,6)$. Each time $\mathscr{P}$ is updated and the dimensions are relabeled, $\mathbb{Y}$ is sorted accordingly.

The prior mean vector and covariance matrix for the region layer are $\mathrm{E}[\mathbb{Z}] = K_s^{-1}\mathbf{P}\mathrm{E}[\mathbb{Y}] = K_s^{-1}\mathbf{P}\boldsymbol{\mu} = \beta \mathbf{1}_{K_r}$ and $\mathrm{Var}(\mathbb{Z}) = K_s^{-2}\mathbf{P}\mathrm{Var}(\mathbb{Y})\mathbf{P}^\top = K_s^{-2}\mathbf{P}\mathbf{Q}^{-1}\mathbf{P}^\top$, respectively. Although $\mathbb{Z}$ is a GMRF, it does not have the same sparse neighborhood structure as the single-layer GMRF. The inverse of $\mathrm{Var}(\mathbb{Z})$ turns out to be a dense matrix, implying that all region-layer nodes neighbor each other in the region-layer graph. To compute the exact region-layer precision matrix, we need $\mathbf{Q}^{-1}$; however,

for the scale of problems of our interest, inverting $\mathbf{Q}$ is computationally infeasible. Moreover, the computational benefit of GMRF is lost with a dense precision matrix. To tackle this challenge, we propose approximating the precision matrix of $\mathbb{Z}$ with an easy-to-compute sparse matrix, as shown in Theorem 2.1.

THEOREM 2.1. *We have that* $(\frac{1}{K_s^2}\mathbf{P}\mathbf{Q}^{-1}\mathbf{P}^\top)\mathbf{P}\mathbf{Q}\mathbf{P}^\top = \mathbf{P}\mathbf{Q}\mathbf{P}^\top(\frac{1}{K_s^2}\mathbf{P}\mathbf{Q}^{-1}\mathbf{P}^\top)$. *Furthermore, if* $k_i > 2$ *for* $i = 1, 2, \ldots, d$, *then*

$$\left|\left(\tfrac{1}{K_s^2}\mathbf{P}\mathbf{Q}^{-1}\mathbf{P}^\top\right)\mathbf{P}\mathbf{Q}\mathbf{P}^\top - \mathbf{I}_{K_r}\right|_{abs} < \frac{4c_2^2}{k_{min}}\mathbf{I}_{K_r} + \frac{2a_1c_1c_2}{1-2c_1}\left(\frac{2}{k_{min}} + 1 - \prod_{i=1}^{d_1}\frac{k_i-2}{k_i}\right)\mathbf{1}_{K_r\times K_r},$$

where $<$ *is taken element-wise,* $k_{min} = \min_{i=1}^{d_1}\{k_i\}$, $a_1 = \sum_{i=1}^{d_1}\theta_i$, $c_1 = \sum_{i=1}^{d}\theta_i$, $c_2 = \max\{2a_1, \theta_{d_1+1}, \ldots, \theta_d\}$, *and* $\mathbf{1}_{K_r\times K_r}$ *is the* $K_r \times K_r$ *matrix of ones.*

The proof of Theorem 2.1 is included in Appendix **??** of the Supplementary Material. Theorem 2.1 provides an element-wise absolute value bound on the difference between $\mathbf{I}_{K_r}$ and the matrix product of Var($\mathbb{Z}$) and $\mathbf{P}\mathbf{Q}\mathbf{P}^\top$, and thereby quantifies the error in approximating the precision matrix of $\mathbb{Z}$ with $\mathbf{P}\mathbf{Q}\mathbf{P}^\top$. From Equation (5), $\mathbf{P}\mathbf{Q}\mathbf{P}^\top$ can be computed easily. Note that $a_1 < c_1 < 0.5$ and $c_2 < 1$ from the diagonal dominance constraint. Because $\frac{1}{k_{\min}}$ and $1 - \prod_{i=1}^{d_1}\frac{k_i-2}{k_i}$ are monotonically decreasing in each k_i, the error bound decreases in k_i. Moreover, when $k_i \to \infty$ for $i = 1, \ldots, d_1$, we have that $\frac{1}{k_{\min}} \to 0$ and $1 - \prod_{i=1}^{d_1}\frac{k_i-2}{k_i} \to 0$. Thus, each element of $(\frac{1}{K_s^2}\mathbf{P}\mathbf{Q}^{-1}\mathbf{P}^\top)\mathbf{P}\mathbf{Q}\mathbf{P}^\top - \mathbf{I}_{K_r}$ converges to 0. We also empirically observed that $\mathbf{P}\mathbf{Q}\mathbf{P}^\top$ approximates $(\text{Var}(\mathbb{Z}))^{-1}$ well even for moderate k_is.

From Theorem 2.1, the prior distribution of $\mathbb{Z}$ can be approximated with $\mathcal{N}(\beta\mathbf{1}_{K_r}, \mathbf{T}^{-1})$, where $\mathbf{T} \triangleq \mathbf{P}\mathbf{Q}\mathbf{P}^\top$. Because $\mathbf{P}\mathbf{Q}\mathbf{P}^\top$ has the same sparsity pattern as $\mathbf{Q}$, all sparse linear algebraic operations tailored for the single-layer GMRF are applicable to the region-layer GMRF. Henceforth, we refer to the approximate model as the region-layer GMRF.

Similar to the single-layer GMRF, we derive the posterior distribution of the region-layer GMRF conditional on the simulation history. Any region-layer node is said to be simulated if at least one of the solutions in its solution-layer graph is simulated. Let $\mathscr{R}_2 \subset \mathscr{R}$ be the set of simulated region-layer nodes and $\mathscr{R}_1 \triangleq \mathscr{R}\backslash\mathscr{R}_2$, and partition $\mathbb{Z}$ to $(\mathbb{Z}_1, \mathbb{Z}_2)^\top$, where $\mathbb{Z}_1$ and $\mathbb{Z}_2$ correspond to $\mathscr{R}_1$ and $\mathscr{R}_2$, respectively. Also, let $\mathscr{X}_2(\mathcal{R}_\ell) \subseteq \mathscr{X}(\mathcal{R}_\ell)$ be the set of simulated solutions in the solution-layer graph of $\mathcal{R}_\ell$ and $\mathscr{X}_1(\mathcal{R}_\ell) \triangleq \mathscr{X}(\mathcal{R}_\ell)\backslash\mathscr{X}_2(\mathcal{R}_\ell)$. For any $\mathcal{R}_\ell \in \mathscr{R}_2$, we cannot observe $z(\mathcal{R}_\ell)$ directly because (i) even if $\mathcal{R}_\ell$ is simulated, it is unlikely that all $\mathbf{x} \in \mathscr{X}(\mathcal{R}_\ell)$ are simulated; and (ii) even if all $\mathbf{x} \in \mathscr{X}(\mathcal{R}_\ell)$ are simulated, we do not observe $y(\mathbf{x})$ without stochastic noise. Instead, we define the stochastic observation of $z(\mathcal{R})$ as

$$\bar{Z}(\mathcal{R}_\ell) = |\mathscr{X}_2(\mathcal{R}_\ell)|^{-1}\sum\nolimits_{\mathbf{x}\in\mathscr{X}_2(\mathcal{R}_\ell)}\bar{Y}(\mathbf{x}). \tag{6}$$

$\bar{Z}(\mathcal{R}_\ell)$ is the average of the stochastic observations at the simulated solutions in the solution-layer graph of $\mathcal{R}_\ell$. Let $\mathcal{Z}_2$ be the vector of $\bar{Z}(\mathcal{R}_\ell)$ at all $\mathcal{R}_\ell \in \mathscr{R}_2$. To model $\mathcal{Z}_2$, we define $\mathbb{Z}_2^\xi = \mathbb{Z}_2 + \boldsymbol{\xi}$, where $\boldsymbol{\xi}|\mathbb{Z}_2 \sim \mathcal{N}(\mathbf{0}_{|\mathscr{R}_2|}, \mathbf{T}_\xi^{-1})$ and $\mathbf{T}_\xi$ is the precision matrix of the stochastic noise for the region-layer GMRF. Since all solutions are simulated independently, $\mathbb{Z}_2^\xi(\mathcal{R}_\ell)$ and $\mathbb{Z}_2^\xi(\mathcal{R}_h)$ for $\mathcal{R}_\ell \neq \mathcal{R}_h$ are independent given $\mathbb{Z}(\mathcal{R}_\ell)$ and $\mathbb{Z}(\mathcal{R}_h)$, which makes T_ξ a diagonal matrix. The diagonal element of $\mathbf{T}_\xi$ corresponding to $\mathcal{R}_\ell$ is $1/\text{Var}(\bar{Z}(\mathcal{R}_\ell))$, where

$$\text{Var}(\bar{Z}(\mathcal{R}_\ell)) = \text{Var}\left(\tfrac{1}{|\mathscr{X}_2(\mathcal{R}_\ell)|}\sum\nolimits_{\mathbf{x}\in\mathscr{X}_2(\mathcal{R}_\ell)}y(\mathbf{x})\right) + \text{E}\left(\tfrac{1}{|\mathscr{X}_2(\mathcal{R}_\ell)|^2}\sum\nolimits_{\mathbf{x}\in\mathscr{X}_2(\mathcal{R}_\ell)}\tfrac{\sigma^2(\mathbf{x})}{r(\mathbf{x})}\right). \tag{7}$$

In Equation (7), we treat $\mathscr{X}_2(\mathcal{R}_\ell)$ as a set of uniformly randomly selected solutions without replacement from $\mathscr{X}(\mathcal{R}_\ell)$. This allows $\mathrm{Var}(\bar{Z}(\mathcal{R}_\ell))$ to be estimated by [Salemi et al. 2019]

$$V(\mathcal{R}_\ell) = \frac{|\mathscr{X}_1(\mathcal{R}_\ell)|/|\mathscr{X}(\mathcal{R}_\ell)|}{|\mathscr{X}_2(\mathcal{R}_\ell)|(|\mathscr{X}_2(\mathcal{R}_\ell)|-1)} \sum_{\mathbf{x}\in\mathscr{X}_2(\mathcal{R}_\ell)} (\bar{Y}(\mathbf{x}) - \bar{Z}(\mathcal{R}_\ell))^2 + \frac{1}{|\mathscr{X}_2(\mathcal{R}_\ell)|^2} \sum_{\mathbf{x}\in\mathscr{X}_2(\mathcal{R}_\ell)} \frac{S^2(\mathbf{x})}{r(\mathbf{x})}. \quad (8)$$

Note that $|\mathscr{X}_1(\mathcal{R}_\ell)|/|\mathscr{X}(\mathcal{R}_\ell)|$ corrects for the fact that $\mathscr{X}(\mathcal{R}_\ell)$ is finite [Cochran 1977]; when all solutions in $\mathcal{R}_\ell$ are simulated, the first term of Equation (8) is 0. We adopt $1/V(\mathcal{R}_\ell)$ as a plug-in estimate of the diagonal element of $\mathbf{T}_\xi$ that corresponds to $\mathcal{R}_\ell$. Given $\mathbb{Z}_2^\xi = \mathcal{Z}_2$, the posterior distribution of $(\mathbb{Z}_1, \mathbb{Z}_2)^\top$ is

$$\mathcal{N}\left(\beta\mathbf{1}_{K_r} + \bar{\mathbf{T}}^{-1}\begin{pmatrix} \mathbf{0}_{|\mathscr{R}_1|} \\ \mathbf{T}_\xi\left(\mathcal{Z}_2 - \beta\mathbf{1}_{|\mathscr{R}_2|}\right)\end{pmatrix}, \bar{\mathbf{T}}^{-1}\right), \text{ where } \bar{\mathbf{T}} \triangleq \begin{bmatrix} \mathbf{T}_{11} & \mathbf{T}_{12} \\ \mathbf{T}_{12}^\top & \mathbf{T}_{22} + \mathbf{T}_\xi \end{bmatrix}. \quad (9)$$

Note that $\mathbf{T}_{ij}$ represents the submatrix of $\mathbf{T}$ corresponding to $\mathscr{R}_i$ and $\mathscr{R}_j$ for $1 \le i, j \le 2$.

The posterior distribution of $\mathbb{Z}$ in Equation (9) provides low-resolution inference to find a region-layer node to sample. This fixes the d_2 region-layer coordinates. To determine the rest of coordinates for the solution to sample, we rely on high-resolution solution-layer inference. Lemma 2.2 lets us update the posterior distribution of the solution-layer GMRF; see Appendix **??** for its proof.

Lemma 2.2. *Let $\mathcal{R}_* \in \mathscr{R}$ and $\mathcal{N}(\mathcal{R}_*) \triangleq (\cup_{\mathbf{x}\in\mathscr{X}(\mathcal{R}_*)}\mathcal{N}(\mathbf{x}))\backslash\mathscr{X}(\mathcal{R}_*)$. Suppose that $\mathbb{Y}^*$ and $\mathbb{Y}_{\mathcal{N}(\mathcal{R}_*)}$ are the subvectors of $\mathbb{Y}$ corresponding to the solutions in $\mathscr{X}(\mathcal{R}_*)$ and $\mathcal{N}(\mathcal{R}_*)$, respectively. Then, $\mathbb{Y}^*|\mathbb{Y}_{\mathcal{N}(\mathcal{R}_*)} = \beta\mathbf{1}_{|\mathcal{N}(\mathcal{R}_*)|} \sim \mathcal{N}(\beta\mathbf{1}_{|\mathscr{X}(\mathcal{R}_*)|}, \mathbf{Q}_*^{-1})$, where $\mathbf{Q}_*$ is the submatrix of $\mathbf{Q}$ corresponding to $\mathbb{Y}^*$.*

Thus, conditional on the fact that the responses at the neighboring solutions of $\mathcal{R}_*$ are equal to their prior means, $\mathbb{Y}^*$ is independent of $\mathbb{Y}\backslash\mathbb{Y}^*$. Known as the mean-field approximation in general, this approximation is adopted in various statistical models with a complex spatial correlation structure to ease computation involved in the analysis. For instance, Hensman et al. [2015] apply the mean-field approximation for variational inference of a GP.

Based on Lemma 2.2, the posterior distribution of $\mathbb{Y}^*$ can be derived similarly as in Equation (3). Let $\mathbb{Y}_1^*$ and $\mathbb{Y}_2^*$ be the GMRF vectors corresponding to $\mathscr{X}_1(\mathcal{R}_*)$ and $\mathscr{X}_2(\mathcal{R}_*)$, respectively. Let $\mathbf{Q}_{ij}^*$ be the submatrix of $\mathbf{Q}_*$ corresponding to $\mathbb{Y}_i^*$ and $\mathbb{Y}_j^*$ for $1 \le i, j \le 2$, and let $\mathcal{Y}_2^*$ be the vector of sample means of the simulation outputs at $\mathscr{X}_2(\mathcal{R}_*)$. We model $\mathcal{Y}_2^*$ as a realization of $\mathbb{Y}_{2,\epsilon}^* = \mathbb{Y}_2^* + \boldsymbol{\epsilon}^*$, where $\boldsymbol{\epsilon}^* \sim \mathcal{N}(\mathbf{0}_{|\mathscr{X}_2(\mathcal{R}_*)|}, \mathbf{Q}_{*,\epsilon}^{-1})$ and $\mathbf{Q}_{*,\epsilon}$ is the precision matrix of stochastic noise. Proposition 2.3 provides the posterior distribution for $\mathbb{Y}^*$ given $\mathbb{Y}_{\mathcal{N}(\mathcal{R}_*)} = \beta\mathbf{1}_{|\mathcal{N}(\mathcal{R}_*)|}$ and $\mathbb{Y}_{2,\epsilon}^* = \mathcal{Y}_2^*$.

Proposition 2.3. *The posterior distribution of $\mathbb{Y}^* \mid \mathbb{Y}_{2,\epsilon}^* = \mathcal{Y}_2^*, \mathbb{Y}_{\mathcal{N}(\mathcal{R}_*)} = \beta\mathbf{1}_{|\mathcal{N}(\mathcal{R}_*)|}$ is*

$$\mathcal{N}\left(\beta\mathbf{1}_{|\mathscr{X}(\mathcal{R}_*)|} + \bar{\mathbf{Q}}_*^{-1}\begin{pmatrix} \mathbf{0}_{|\mathscr{X}_1(\mathcal{R}_*)|} \\ \mathbf{Q}_{*,\epsilon}(\mathcal{Y}_2^* - \beta\mathbf{1}_{|\mathscr{X}_2(\mathcal{R}_*)|})\end{pmatrix}, \bar{\mathbf{Q}}_*^{-1}\right), \text{ where } \bar{\mathbf{Q}}_* = \begin{bmatrix} \mathbf{Q}_{11}^* & \mathbf{Q}_{12}^* \\ \mathbf{Q}_{12}^{*\top} & \mathbf{Q}_{22}^* + \mathbf{Q}_{*,\epsilon} \end{bmatrix}. \quad (10)$$

See Appendix **??** for the proof. We exploit Proposition 2.3 in two ways to improve the efficiency of the pGMIA. First, it lets us estimate the hyperparameters of the GMRF efficiently by separating the region- and solution-layer estimation problems, as we describe in Section 3. Second, Proposition 2.3 enables us to update the solution-layer GMRF's posterior distribution locally for $\mathcal{R}_*$ without updating the posterior distribution of the entire single-layer GMRF. Recall that updating the single-layer model is computationally infeasible for large $\mathscr{X}$.

The pGMIA makes the sampling decisions utilizing the posterior distributions of the two-layer models. The CEI is applied in the region layer to select the next node to "sample". Recall that sampling a region-layer node means that a solution projected to the node will be selected for

sampling. The CEI of $\mathcal{R} \in \mathscr{R}$ is

$$\mathrm{CEI}(\tilde{\mathcal{R}}, \mathcal{R}) = \mathrm{E}[\max(\mathbb{Z}(\tilde{\mathcal{R}}) - \mathbb{Z}(\mathcal{R}), 0)] = (\mathbf{M}(\tilde{\mathcal{R}}) - \mathbf{M}(\mathcal{R}))\Phi\left(\frac{\mathbf{M}(\tilde{\mathcal{R}}) - \mathbf{M}(\mathcal{R})}{\sqrt{\mathbf{V}(\tilde{\mathcal{R}}, \mathcal{R})}}\right) + \sqrt{\mathbf{V}(\tilde{\mathcal{R}}, \mathcal{R})}\phi\left(\frac{\mathbf{M}(\tilde{\mathcal{R}}) - \mathbf{M}(\mathcal{R})}{\sqrt{\mathbf{V}(\tilde{\mathcal{R}}, \mathcal{R})}}\right), \tag{11}$$

where $\tilde{\mathcal{R}} \in \mathscr{R}_2$ is the node with the smallest observed region-layer output, $\mathbf{M}(\mathcal{R})$ and $\mathbf{V}(\mathcal{R})$ denote the posterior mean and variance of $\mathbb{Z}(\mathcal{R})$, respectively, $\mathbf{C}(\tilde{\mathcal{R}}, \mathcal{R})$ is the posterior covariance between $\mathbb{Z}(\mathcal{R})$ and $\mathbb{Z}(\tilde{\mathcal{R}})$, and $\mathbf{V}(\tilde{\mathcal{R}}, \mathcal{R}) = \mathbf{V}(\tilde{\mathcal{R}}) + \mathbf{V}(\mathcal{R}) - 2\mathbf{C}(\tilde{\mathcal{R}}, \mathcal{R})$. Note that $\mathbf{M}(\mathcal{R})$ and $\mathbf{V}(\mathcal{R})$ are the elements of the mean vector and the diagonal of $\bar{\mathbf{T}}^{-1}$ in Equation (9), respectively, corresponding to $\mathcal{R}$ and $\mathbf{C}(\tilde{\mathcal{R}}, \mathcal{R})$ is the off-diagonal element in $\bar{\mathbf{T}}^{-1}$ corresponding to $\mathcal{R}$ and $\tilde{\mathcal{R}}$.

Once $\mathcal{R}_\ell = \arg\max_{\mathcal{R} \in \mathscr{R}} \mathrm{CEI}(\tilde{\mathcal{R}}, \mathcal{R})$ is selected, we compute the following conditional CEI at the solutions in $\mathscr{X}(\mathcal{R}_\ell)$ for solution-layer sampling:

$$\mathrm{CEI}(\tilde{\mathbf{x}}(\mathcal{R}_\ell), \mathbf{x}; \mathcal{R}_\ell) = \mathrm{E}[\max(\mathbb{Y}(\tilde{\mathbf{x}}(\mathcal{R}_\ell)) - \mathbb{Y}(\mathbf{x}), 0) \mid \mathbb{Y}_{\mathscr{N}(\mathcal{R}_\ell)} = \beta \mathbf{1}_{|\mathscr{N}(\mathcal{R}_\ell)|}], \tag{12}$$

where $\tilde{\mathbf{x}}(\mathcal{R}_\ell)$ is the solution in $\mathscr{X}_2(\mathcal{R}_\ell)$ with the smallest sample mean. The expectation in Equation (12) can be computed as in Equation (4) from the conditional distribution in Equation (10) by setting $\mathcal{R}_* = \mathcal{R}_\ell$. We select $\mathbf{x}_{\mathrm{CEI}}(\mathcal{R}_\ell) = \arg\max_{\mathbf{x} \in \mathscr{X}(\mathcal{R}_\ell)} \mathrm{CEI}(\tilde{\mathbf{x}}(\mathcal{R}_\ell), \mathbf{x}; \mathcal{R}_\ell)$ as well as $\tilde{\mathbf{x}}(\mathcal{R}_\ell)$ for sampling.

3 HYPERPARAMETER ESTIMATION FOR PROJECTED GMRF

The prior/posterior distribution of the GMRF depends on the hyperparameters, β and $\boldsymbol{\theta}$. These values are typically estimated after simulating an initial design. Song and Dong [2018] consider **generalized method of moments (GMM)** estimation and Salemi et al. [2019] and Semelhago et al. [2021] adopt the MLEs. However, these algorithms estimate the hyperparameters of the single-layer GMRF, which cannot be done for the scale of the problem considered here.

In this section, we discuss efficient estimation of β and $\boldsymbol{\theta}$ from the region- and solution-layer models. Section 3.1 reviews the MLE problem for single-layer GMRF. In Section 3.2, we discuss how the single-layer MLE problem can be separated into lower-dimensional region- and solution-layer MLE problems by applying our projection scheme.

3.1 MLEs for Single-Layer GMRF

For the single-layer GMRF, Salemi et al. [2019] derive the log-likelihood function given $\mathbb{Y}_2^\epsilon = \mathcal{Y}_2$ to be

$$\mathcal{L}(\beta, \boldsymbol{\theta} \mid \mathbb{Y}_2^\epsilon = \mathcal{Y}_2) \propto \tfrac{1}{2}\log|(\Sigma_{22} + \mathbf{Q}_\epsilon^{-1})^{-1}| - \tfrac{1}{2}(\mathcal{Y}_2 - \beta \mathbf{1}_{|\mathscr{X}_2|})^\top (\Sigma_{22} + \mathbf{Q}_\epsilon^{-1})^{-1}(\mathcal{Y}_2 - \beta \mathbf{1}_{|\mathscr{X}_2|}), \tag{13}$$

where $\Sigma_{22} = (\mathbf{Q}_{22} - \mathbf{Q}_{12}^\top \mathbf{Q}_{11}^{-1} \mathbf{Q}_{12})^{-1}$. From the optimality condition of Equation (13), the MLE of β can be written as a function of $\boldsymbol{\theta}$:

$$\hat{\beta}(\boldsymbol{\theta}) = \left(\mathbf{1}_{|\mathscr{X}_2|}^\top \left(\Sigma_{22} + \mathbf{Q}_\epsilon^{-1}\right)^{-1} \mathbf{1}_{|\mathscr{X}_2|}\right)^{-1} \mathbf{1}_{|\mathscr{X}_2|}^\top \left(\Sigma_{22} + \mathbf{Q}_\epsilon^{-1}\right)^{-1} \mathcal{Y}_2. \tag{14}$$

We define the profile likelihood, $\mathcal{L}(\boldsymbol{\theta} \mid \mathbb{Y}_2^\epsilon = \mathcal{Y}_2)$, by plugging Equation (14) into $\mathcal{L}(\beta, \boldsymbol{\theta} \mid \mathbb{Y}_2^\epsilon = \mathcal{Y}_2)$. Recall from Section 2.1 that the feasibility constraints for $\boldsymbol{\theta}$ are (i) $\theta_0 > 0$, (ii) $\theta_i \geq 0$ for $i = 1, 2, \ldots, d$, and (iii) $\sum_{i=1}^d \theta_i < 0.5$. Thus, the MLE of $\boldsymbol{\theta}$ can be obtained by solving

$$\begin{aligned} \max_{\boldsymbol{\theta}} \quad & \mathcal{L}\left(\boldsymbol{\theta} \mid \mathbb{Y}_2^\epsilon = \mathcal{Y}_2\right) \\ \text{subject to} \quad & \sum\nolimits_{i=1}^d \theta_i < 0.5, \theta_0 > 0 \ \text{ and } \ \theta_i \geq 0, \ i = 1, 2, \ldots, d, \end{aligned} \tag{15}$$

where strict inequality $<$ ($>$) can be replaced with $\leq$ ($\geq$) by subtracting (adding) a small constant to the right-hand side.

For high-dimensional problems, estimating $\boldsymbol{\theta}$ by solving Equation (15) is challenging because Equations (13) and (14) are expensive to compute. Both require Σ_{22} to be computed first, for which the bottleneck is factorizing $\mathbf{Q}_{11}$. Since $|\mathscr{X}_1| \gg |\mathscr{X}_2|$, factorizing $\mathbf{Q}_{11}$ is just as expensive as factorizing $\mathbf{Q}$, which is prohibitively high for the scale of $\mathscr{X}$ considered here.

3.2 Hierarchical MLEs for Region- and Solution-Layer GMRFs

In this section, we reformulate Equation (15) to lower-dimensional region- and solution-layer MLE problems by exploiting Theorem 2.1 and Proposition 2.3. We refer to this estimation procedure as a hierarchical estimation scheme, as the region- and solution-layer MLE problems are solved in sequence. Below, we first introduce the region-layer MLE problem.

Let $\boldsymbol{\tau} = (\tau_0, \tau_1, \ldots, \tau_{d_2})$ be the hyperparameter vector of the approximate region-layer precision matrix, $\mathbf{T}$. Since each element of $\mathbf{T} = \mathbf{PQP}^\top$ is the sum of the elements of a block matrix of $\mathbf{Q}$, we can obtain the functional relationship between $\boldsymbol{\tau}$ and $\boldsymbol{\theta}$. That is, $\tau_0 = K_s\theta_0(1 - 2\sum_{i=1}^{d_1}\theta_i)$, and $\tau_i = K_s\theta_0\theta_{d_1+i}/\tau_0$ for $i = 1, 2, \ldots, d_2$. Or, equivalently,

$$\sum\nolimits_{i=1}^{d_1}\theta_i = 0.5 - \frac{\tau_0}{2K_s\theta_0}, \tag{16}$$

$$\theta_{d_1+i} = \frac{\tau_0\tau_i}{K_s\theta_0}, \quad i = 1, 2, \ldots, d_2. \tag{17}$$

Similar to the single-layer GMRF MLE problem, the region-layer MLEs can be found by maximizing

$$\mathcal{L}\left(\beta, \boldsymbol{\tau} \mid \mathscr{P}, \mathbb{Z}_2^{\xi} = \mathcal{Z}_2\right) \propto \frac{1}{2}\log|\left(\mathbf{K}_{22} + \mathbf{T}_{\xi}^{-1}\right)^{-1}| - \tfrac{1}{2}\left(\mathcal{Z}_2 - \beta\mathbf{1}_{|\mathscr{R}_2|}\right)^\top\left(\mathbf{K}_{22} + \mathbf{T}_{\xi}^{-1}\right)^{-1}\left(\mathcal{Z}_2 - \beta\mathbf{1}_{|\mathscr{R}_2|}\right), \tag{18}$$

where $\mathbf{K}_{22} = (\mathbf{T}_{22} - \mathbf{T}_{12}^\top\mathbf{T}_{11}^{-1}\mathbf{T}_{12})^{-1}$. Recall that $\mathrm{E}[\mathbb{Z}] = \beta\mathbf{1}_{K_r}$. Thus, the MLE of β can be computed at the region layer. As in Equation (14), β that maximizes Equation (18) can be written as a function of $\boldsymbol{\tau}$:

$$\hat{\beta}(\boldsymbol{\tau}) = \left(\mathbf{1}_{|\mathscr{R}_2|}^\top\left(\mathbf{K}_{22} + \mathbf{T}_{\xi}^{-1}\right)^{-1}\mathbf{1}_{|\mathscr{R}_2|}\right)^{-1}\mathbf{1}_{|\mathscr{R}_2|}^\top\left(\mathbf{K}_{22} + \mathbf{T}_{\xi}^{-1}\right)^{-1}\mathcal{Z}_2. \tag{19}$$

Consequently, $\mathcal{L}(\boldsymbol{\tau} \mid \mathbb{Z}_2^{\xi} = \mathcal{Z}_2)$ is defined by plugging Equation (19) into Equation (18). The MLE of $\boldsymbol{\tau}$ can be found by solving

$$\begin{aligned} \max_{\boldsymbol{\tau}} \quad & \mathcal{L}(\boldsymbol{\tau} \mid \mathbb{Z}_2^{\xi} = \mathcal{Z}_2) \\ \text{subject to} \quad & \sum\nolimits_{i=1}^{d_2}\tau_i < 0.5, \tau_0 > 0 \text{ and } \tau_i \geq 0 \text{ for } i = 1, 2, \ldots, d_2. \end{aligned} \tag{20}$$

Once solution $\hat{\boldsymbol{\tau}}$ to Equation (20) is found, $\theta_{d_1+1}, \theta_{d_1+2}, \ldots, \theta_d$ are identified up to a scaling factor from Equation (17). Let $\boldsymbol{\theta}^s \triangleq (\theta_1, \theta_2, \ldots, \theta_{d_1})$ be the vector of remaining elements of $\boldsymbol{\theta}$. From Equation (16), θ_0 can be written as a function of $\boldsymbol{\theta}^s$ and τ_0.

We exploit the conditional distribution in Proposition 2.3 to set up a computationally efficient solution-layer MLE problem for $\boldsymbol{\theta}^s$. The solution-layer log-likelihood function at $\mathcal{R}_*$ computed from Equation (10) is

$$\begin{aligned} &\mathcal{L}\left(\beta, \boldsymbol{\theta}^s \mid \mathbb{Y}_{2,\epsilon}^* = \mathcal{Y}_2^*, \mathcal{N}(\mathcal{R}_*) = \beta\mathbf{1}_{|\mathcal{N}(\mathcal{R}_*)|}\right) \\ &\quad\propto \tfrac{1}{2}\log|\left(\Sigma_{22}^* + \mathbf{Q}_{*,\epsilon}^{-1}\right)^{-1}| - \tfrac{1}{2}\left(\mathcal{Y}_2^* - \hat{\beta}\mathbf{1}_{|\mathscr{X}_2(\mathcal{R}_*)|}\right)^\top\left(\Sigma_{22}^* + \mathbf{Q}_{*,\epsilon}^{-1}\right)^{-1}\left(\mathcal{Y}_2^* - \hat{\beta}\mathbf{1}_{|\mathscr{X}_2(\mathcal{R}_*)|}\right), \end{aligned}$$

where $\Sigma_{22}^* = (\mathbf{Q}_{22}^* - \mathbf{Q}_{12}^{*\top}\mathbf{Q}_{11}^{*-1}\mathbf{Q}_{12}^*)^{-1}$. Recall that θ_0 is a function of $\boldsymbol{\theta}^s$ given $\hat{\tau}_0$ and $\hat{\beta}$ is already computed in the region layer from Equation (19). Therefore, the solution-layer MLE problem can

be constructed as

$$\begin{aligned} \max_{\theta_0, \boldsymbol{\theta}^s} \quad & \mathcal{L}\left(\hat{\beta}, \boldsymbol{\theta}^s \mid \mathbb{Y}^*_{2,\epsilon} = \boldsymbol{\mathcal{Y}}^*_2, \mathcal{N}(\mathcal{R}_*) = \beta \mathbf{1}_{|\mathcal{N}(\mathcal{R}_*)|}\right) \\ \text{subject to} \quad & \frac{1}{\theta_0} \le \frac{K_s}{\hat{\tau}_0}, \ \frac{\hat{\tau}_0}{K_s \theta_0} + 2\sum_{i=1}^{d_1} \theta_i = 1, \ \theta_0 > 0 \text{ and } \theta_i \ge 0 \text{ for } i = 1, 2, \ldots, d_1. \end{aligned} \tag{21}$$

Reparameterizing $\tilde{\theta}_0 = \frac{1}{\theta_0}$, all constraints in Equation (21) become linear. The equality constraint in Equation (21) can be removed by substituting $\tilde{\theta}_0$ as a linear function of $\boldsymbol{\theta}^s$. Since our goal is to find the global optimum, we choose $\mathcal{R}_* = \mathcal{R}_{\min}$ to compute $\boldsymbol{\theta}^s$, where $\mathcal{R}_{\min}$ is the node to which the current best solution is projected so that the resulting solution-layer GMRF best fits $\mathcal{R}_{\min}$ and makes the solution-layer search in $\mathcal{R}_{\min}$ more effective.

The hierarchical scheme has significant computational advantages over the single-layer MLE problem. For Equations (20) and (21), the most expensive operations are factorizing $\mathbf{T}_{11}$ and $\mathbf{Q}^*_{11}$, respectively, which are far smaller than $\mathbf{Q}_{11}$.

We can also ensure that the estimate of $\boldsymbol{\theta}$ obtained from the hierarchical scheme satisfies the feasibility constraint of the single-layer MLE problem. Clearly, $\boldsymbol{\theta}$ is nonnegative. Thus, it remains to show that the diagonal dominance condition holds. Observe that

$$\sum_{i=1}^{d} \theta_i = \sum_{i=1}^{d_1} \theta_i + \sum_{i=d_1+1}^{d} \theta_i = \left(0.5 - \frac{\tau_0}{2K_s\theta_0}\right) + \frac{\tau_0}{K_s\theta_0}\sum_{i=1}^{d_2}\tau_i = 0.5 - \frac{\tau_0}{K_s\theta_0}\left(0.5 - \sum_{i=1}^{d_2}\tau_i\right) < 0.5,$$

where the second equality follows from Equations (16) and (17), and the last inequality holds because $0 < \sum_{i=1}^{d_2}\tau_i < 0.5$ and $0 < \tau_0/(K_s\theta_0) \le 1$ from the first constraint of Equation (21).

Although adopting the same $\hat{\beta}$ for both region and solution layers is statistically consistent with our modeling assumptions, in practice, this tends to make solution-layer GMRFs provide poor inference when $\mathscr{X}$ is large. In particular, when there are only a few simulated solutions in a region-layer node and the sample means of the outputs at these solutions are smaller than $\hat{\beta}$, then the posterior distribution of the posterior means of the solution-layer GMRF at unsimulated solutions tend to be large, i.e., reversion to the mean. This causes the algorithm to exploit the simulated solutions instead of exploring new solutions for several iterations. To avoid this, we make heuristic adjustments in our experiments in Section 5; when $\mathcal{R} \in \mathscr{R}_2$ is selected for sampling, we recompute the MLE $\hat{\beta}(\mathcal{R})$ from Equation (14) from the simulation observations made within $\mathcal{R}$.

4 PROJECTED GAUSSIAN MARKOV IMPROVEMENT ALGORITHM

In this section, we propose our DOvS algorithms based on the projected GMRF model. The pGMIA is introduced in Section 4.1. Then, it is extended to the pGMIA+ in Section 4.2 to address even higher-dimensional problems. A complete table of notation is provided in Appendix **??** for convenience. Both pGMIA and pGMIA+ update $\mathscr{P}$ periodically.

4.1 Algorithmic Details of pGMIA

Algorithm 1 outlines the pGMIA. The pGMIA first randomly selects the initial $\mathscr{P}$ and then solves the region- and solution-layer MLE problems hierarchically. At each iteration, the pGMIA finds new solutions to sample by performing the *hierarchical search* in Algorithm 3. When a *projection update criterion* is satisfied, it updates $\mathscr{P}$ according to a *projection selection criterion* and computes the GMRF hyperparameters accordingly, as described in Algorithm 4.

There are three criteria that require user inputs in Algorithm 1: stopping (Step 5), projection update (Step 7), and projection selection (Step 8) criteria. Although *any* user-defined criteria can be applied within the pGMIA, we discuss the choices adopted in our experiments in Section 5.

ALGORITHM 1: pGMIA: Projected Gaussian Markov Improvement Algorithm

Input : d_1, n_r, n_s, r; stopping, projection update & selection criteria
Randomly select $\mathscr{P}$ for initialization; set $iter = 1$.
Choose n_r initial design nodes and n_s initial design solutions projected to each design node by space filling.
At each design solution $\mathbf{x}$, simulate r replications and compute $\bar{Y}(\mathbf{x})$ and $S^2(\mathbf{x})$.
Compute two-layer GMRF hyperparameters → **Algorithm 4**
while *stopping criterion not satisfied* **do**
    Perform *hierarchical search* → **Algorithm 3**
    if *projection update criterion is satisfied* **then**
        Choose $\mathscr{P}$ according to the *projection selection criterion* (e.g., **Algorithm 2**).
        Update the projection and compute two-layer GMRF hyperparameters → **Algorithm 4**
    else
        Update $\bar{Z}(\mathcal{R})$ and $V(\mathcal{R})$ according to Equations (6) and (8), respectively, for all $\mathcal{R} \in \mathscr{R}_2$ with $|\mathscr{X}_2(\mathcal{R})| \geq 2$.
        Update $\tilde{\mathbf{x}} = \arg\min_{\mathbf{x} \in \mathscr{X}_2} \bar{Y}(\mathbf{x})$ and $\mathcal{R}_{\min} = \{\mathcal{R} \in \mathscr{R} : \tilde{\mathbf{x}} \in \mathscr{X}(\mathcal{R})\}$.
    end
    Set $iter = iter + 1$.
end
Output : $\tilde{\mathbf{x}}$.

For the stopping criterion, we update $\mathscr{P}$ every p iterations for some fixed p. Since the search progress tends to slow down under the same projection as iterations proceed, large p is not desirable. As the MLE of the hyperparameters are recomputed each time $\mathscr{P}$ is updated, small p tends to increase the per-iteration computational cost. In Section 5.1.1, we empirically show that the performance of the pGMIA is robust for a wide range of p.

For projection selection, we test two approaches in our empirical study in Section 5. The first is to choose $\mathscr{P}$ at random among $\binom{d}{d_1}$ candidates with equal probabilities. The second is to apply a global sensitivity measure to select d_2 "most important" dimensions. Székely et al. [2007] introduce the distance correlation, $R(X_1, X_2)$, to measure dependence between two random vectors X_1 and X_2: $R(X_1, X_2) \triangleq \left(\frac{\mathcal{V}^2(X_1, X_2)}{\sqrt{\mathcal{V}^2(X_1, X_1)\mathcal{V}^2(X_2, X_2)}}\right)^{1/2}$, where

$$\mathcal{V}^2(X_1, X_2) = \mathrm{E}[\|X_1 - X_1'\|_2 \|X_2 - X_2'\|_2] + \mathrm{E}[\|X_1 - X_1'\|_2]\mathrm{E}[\|X_2 - X_2'\|_2] - 2\mathrm{E}\left[\mathrm{E}[\|X_1 - X_1'\|_2 |X_1] - \mathrm{E}[\|X_2 - X_2'\|_2 |X_2]\right],$$

X_1 and X_1' (X_2 and X_2') are independent random vectors following the same marginal distribution. Note that $R(X_1, X_2)$ takes a value between 0 and 1, and $R(X_1, X_2) = 0$, if and only if X_1 and X_2 are independent. A benefit of $R(X_1, X_2)$ over the product–moment correlation is that X_1 and X_2 need not have the same dimension. Da Veiga [2015] applies $R(X_1, X_2)$ in global sensitivity analysis to measure the sensitivity of GP prediction (X_2) to a subset of input dimensions (X_1). In our notation, they estimate the distance correlation between $\mathbf{M}(\mathbf{x})$ and x_i by sampling n_g space-filling design points in $\mathscr{X}$, $\mathscr{X}_g = \{\mathbf{x}_a = (x_{1,a}, x_{2,a}, \ldots, x_{d,a}), 1 \leq a \leq n_g\}$, then computing

$$\hat{R}(x_i, \mathbf{M}(\mathbf{x})) = \left(\frac{\hat{\mathcal{V}}^2(x_i, \mathbf{M}(\mathbf{x}))}{\sqrt{\hat{\mathcal{V}}^2(x_i, x_i)\hat{\mathcal{V}}^2(\mathbf{M}(\mathbf{x}), \mathbf{M}(\mathbf{x}))}}\right)^{1/2}, \tag{22}$$

where

ALGORITHM 2: Distance Correlation Projection Selection Criterion

Input : $\mathscr{P}, n_g, \hat{\beta}, \hat{\tau}, \hat{\boldsymbol{\theta}^s}, \bar{Y}(\mathscr{X}_2), S^2(\mathscr{X}_2), \bar{Z}(\mathscr{R}_2)$ and $V(\mathscr{R}_2)$,
Select a set of n_g design solutions $\mathscr{X}_g$ via Latin hypercube sampling.
for $\mathbf{x} \in \mathscr{X}_g$ **do**
  Find $\mathcal{R} = \{\mathcal{R} \in \mathscr{R} : \mathbf{x} \in \mathscr{X}(\mathcal{R})\}$
  if $\mathcal{R} \in \mathscr{R}_2$ **then**
    Compute the solution-layer posterior mean $\mathbf{M}(\mathbf{x})$ within $\mathcal{R}$ from (10) at $\mathbf{x}$.
  else
    Compute the region-layer posterior mean $\mathbf{M}(\mathcal{R})$ from Equation (9), and set $\mathbf{M}(\mathbf{x}) = \mathbf{M}(\mathcal{R})$.
  end
end
Calculate $\hat{R}(x_i, \mathbf{M}(\mathbf{x}))$ from Equation (22), for all $i = 1, \ldots, d$.
Select d_2 region-layer dimensions with d_2 largest $\hat{R}(x_i, \mathbf{M}(\mathbf{x}))$.
Output: $\mathscr{P}$.

$$\hat{\mathcal{V}}^2(x_i, \mathbf{M}(\mathbf{x})) = \frac{1}{n_g^2}\sum_{a=1}^{n_g}\sum_{b=1}^{n_g}\|x_{i,a} - x_{i,b}\|_2\|\mathbf{M}(\mathbf{x}_a) - \mathbf{M}(\mathbf{x}_b)\|_2 + \left(\frac{1}{n_g^2}\sum_{a=1}^{n_g}\sum_{b=1}^{n_g}\|x_{i,a} - x_{i,b}\|_2\right)\left(\frac{1}{n_g^2}\sum_{a=1}^{n_g}\sum_{b=1}^{n_g}\|\mathbf{M}(\mathbf{x}_a) - \mathbf{M}(\mathbf{x}_b)\|_2\right)$$
$$- \frac{2}{n_g}\sum_{a=1}^{n_g}\left[\left(\frac{1}{n_g}\sum_{b=1}^{n_g}\|x_{i,a} - x_{i,b}\|_2\right)\left(\frac{1}{n_g}\sum_{b=1}^{n_g}\|\mathbf{M}(\mathbf{x}_a) - \mathbf{M}(\mathbf{x}_b)\|_2\right)\right],$$
$$\hat{\mathcal{V}}^2(\mathbf{M}(\mathbf{x}), \mathbf{M}(\mathbf{x})) = \frac{1}{n_g^2}\sum_{a=1}^{n_g}\sum_{b=1}^{n_g}\|\mathbf{M}(\mathbf{x}_a) - \mathbf{M}(\mathbf{x}_b)\|_2^2 + \left(\frac{1}{n_g^2}\sum_{a=1}^{n_g}\sum_{b=1}^{n_g}\|\mathbf{M}(\mathbf{x}_a) - \mathbf{M}(\mathbf{x}_b)\|_2\right)^2,$$

and $\hat{\mathcal{V}}^2(x_i, x_i)$ is computed similarly as $\hat{\mathcal{V}}^2(\mathbf{M}(\mathbf{x}), \mathbf{M}(\mathbf{x}))$, replacing $\mathbf{M}(\mathbf{x}_a)$ and $\mathbf{M}(\mathbf{x}_b)$ with $x_{i,a}$ and $x_{i,b}$, respectively. In general, x_i can be replaced with any subvector of $\mathbf{x}$ to estimate the distance correlation between a set of dimensions and $\mathbf{M}(\mathbf{x})$. In this case, $x_{i,a}$ is replaced with the corresponding subvector of $\mathbf{x}_a$.

Extending this idea, we apply the distance correlation in Algorithm 2 to decide the d_2 dimensions to include in the region layer. Ideally, we would like to choose d_2 dimensions that jointly have the largest distance correlation with $\mathbf{M}(\mathbf{x})$. However, this requires estimating $\binom{d}{d_2}$ distance correlations, which can be expensive when d is large. Instead, Algorithm 2 greedily selects d_2 dimensions with the highest marginal distance correlations.

Another challenge in computing Equation (22) is that the pGMIA does not compute the single-layer GMRF's posterior mean, which is prohibitively expensive to compute for our problem scale. Instead, we use the solution-layer GMRF's posterior mean at $\mathbf{x}$ in Equation (10) for $\mathbf{M}(\mathbf{x})$ in Equation (22) if $\mathbf{x}$ belongs to the region-layer node, $\mathcal{R}$, that has been simulated before. Otherwise, we use the region-layer posterior mean of $\mathcal{R}$, $\mathbf{M}(\mathcal{R})$, in place of $\mathbf{M}(\mathbf{x})$.

Algorithm 3 presents how the pGMIA selects the next group of solutions to simulate. From the region-layer posterior distribution, the pGMIA finds three region-layer nodes to explore: (i) the node that the current best solution is projected to ($\mathcal{R}_{\min}$); (ii) the current region-layer optimum ($\tilde{\mathcal{R}} = \arg\min_{\mathcal{R}\in\mathscr{R}_2}\bar{Z}(\mathcal{R})$); and (iii) the node with the largest region-layer CEI value ($\mathcal{R}_{\text{CEI}}$). Sampling $\tilde{\mathcal{R}}$ and $\mathcal{R}_{\text{CEI}}$ facilitates pGMIA's global convergence guarantee in Theorem 4.1 stated at the end of this section. We borrow the idea of sampling $\mathcal{R}_{\min}$ from MR-GMIA in Salemi et al. [2019]; although sampling $\mathcal{R}_{\min}$ does not affect the global convergence guarantee, we observed that doing so can improve the finite-sample performance by sampling more aggressively near the current best solution. Note that $\mathcal{R}_{\min}$ may coincide with $\mathcal{R}_{\text{CEI}}$ or $\tilde{\mathcal{R}}$. In such cases, we sample only two region-layer nodes in that iteration.

ALGORITHM 3: Hierarchical search

Input : $n_s, r, \mathscr{P}, \hat{\beta}, \hat{\boldsymbol{\tau}}, \hat{\boldsymbol{\theta}^s}, \bar{Y}(\mathscr{X}_2), S^2(\mathscr{X}_2), \bar{Z}(\mathscr{R}_2)$ and $V(\mathscr{R}_2)$

1 Find $\tilde{\mathcal{R}} = \arg\min_{\mathcal{R}\in\mathscr{R}_2} \bar{Z}(\mathcal{R})$.
2 Compute $\mathbf{M}(\mathcal{R})$, $\mathbf{V}(\mathcal{R})$, and $\mathbf{C}(\tilde{\mathcal{R}}, \mathcal{R})$ from Equation (9), $\forall \mathcal{R} \in \mathscr{R}$.
3 Compute $\text{CEI}(\tilde{\mathcal{R}}, \mathcal{R})$ from Equation (11), $\forall \mathcal{R} \in \mathscr{R}$.
4 Find $\mathcal{R}_{\text{CEI}} = \arg\max_{\mathcal{R}\in\mathscr{R}} \text{CEI}(\tilde{\mathcal{R}}, \mathcal{R})$.
5 **for** $\mathcal{R} \in \{\mathcal{R}_{\min}, \tilde{\mathcal{R}}, \mathcal{R}_{\text{CEI}}\}$ **do**
6 **if** $|\mathscr{X}_2(\mathcal{R})| < n_s$ **then**
7 Select $n_s - |\mathscr{X}_2(\mathcal{R})|$ new solutions via space filling.
8 Run r replications and compute $\bar{Y}(\mathbf{x})$ and $S^2(\mathbf{x})$ at each new solution.
9 **end**
10 Find $\tilde{\mathbf{x}}(\mathcal{R}) = \arg\min_{\mathbf{x}\in\mathscr{X}_2(\mathcal{R})} \bar{Y}(\mathbf{x})$.
11 Compute $\mathbf{M}(\mathbf{x})$, $\mathbf{V}(\mathbf{x})$, and $\mathbf{C}(\tilde{\mathbf{x}}(\mathcal{R}), \mathbf{x})$ from Equation (10) for all $\mathbf{x} \in \mathscr{X}(\mathcal{R})$.
12 Compute $\text{CEI}(\tilde{\mathbf{x}}(\mathcal{R}), \mathbf{x}; \mathcal{R})$ from Equation (12), $\forall \mathbf{x} \in \mathscr{X}(\mathcal{R})$.
13 Find $\mathbf{x}_{\text{CEI}}(\mathcal{R}) = \arg\max_{\mathbf{x}\in\mathscr{X}(\mathcal{R})} \text{CEI}(\tilde{\mathbf{x}}(\mathcal{R}), \mathbf{x}; \mathcal{R})$.
14 Simulate r replications at $\tilde{\mathbf{x}}(\mathcal{R})$ and $\mathbf{x}_{\text{CEI}}(\mathcal{R})$; update $\bar{Y}(\tilde{\mathbf{x}}(\mathcal{R}))$, $S^2(\tilde{\mathbf{x}}(\mathcal{R}))$, $\bar{Y}(\mathbf{x}_{\text{CEI}}(\mathcal{R}))$ and $S^2(\mathbf{x}_{\text{CEI}}(\mathcal{R}))$.
15 Update $\bar{Z}(\mathcal{R})$ and $V(\mathcal{R})$ from Equations (6) and (8), respectively.
16 **end**

Output: $\bar{Y}(\mathscr{X}_2), S^2(\mathscr{X}_2), \bar{Z}(\mathscr{R}_2)$ and $V(\mathscr{R}_2)$.

For each of the selected region-layer nodes, the algorithm updates the solution-layer posterior distribution and finds the current best solution and the solution with the largest CEI to simulate within the region. These operations can be parallelized, as no information is exchanged among the nodes. Any of the selected region-layer nodes may be undersampled, in which case the inference at the solution-layer GMRF may be poor. To prevent this, Steps 6 to 9 first ensure that we sample at least n_s design solutions within each region. If $|\mathscr{X}_2(\mathcal{R})| > 0$, i.e., $\mathcal{R}$ has been sampled before, then we apply the approach proposed in Wang [2003] to obtain $n_s - |\mathscr{X}_2(\mathcal{R})|$ new solutions. Given n_s, their algorithm first defines the Latin Hypercube grid in the domain and shrinks the grid by leaving out the rows and columns that already contain existing design solutions. Then, it generates a Latin Hypercube sample of size $n_s - |\mathscr{X}_2(\mathcal{R})|$ on the shrunk grid and maps it back to the original domain. We modify their algorithm to sample design solutions on the integer grid; further details can be found in Appendix **??**.

In Steps 2 and 11 of Algorithm 3, in lieu of computing the full posteriors, we only compute the elements of the posterior mean and covariance matrix necessary for computing CEIs in the region and solution layers, respectively.

When $\mathscr{P}$ is updated, the region-layer hyperparameters, $\boldsymbol{\tau}$, must be updated according to the new $\mathscr{P}$. The new $\boldsymbol{\tau}$ can be computed from the old $\boldsymbol{\tau}$ and $\boldsymbol{\theta}^s$ estimated under the previous projection, say, $\mathcal{P}'$, by first computing $\theta_{d_1+1}, \theta_{d_1+2}, \ldots, \theta_d$ from Equation (17), then solving $\mathbf{T} = \mathbf{PQP}^\top$ for each element of $\boldsymbol{\tau}$. However, we empirically observed that updating the hyperparameters allows pGMIA to make a faster search progress, particularly in earlier iterations.

Algorithm 4 details the projection and hyperparameter update in the pGMIA. Updating $\mathscr{P}$ tends to increase $|\mathscr{R}_2|$ because the solutions projected to the same region-layer node under the previous projection $\mathscr{P}'$ have the same coordinates in the dimensions included in the region layer under $\mathscr{P}'$. Thus, when the region-layer dimensions change under $\mathscr{P}$, those solutions are allocated to different region-layer nodes, possibly generating more sampled nodes. In earlier iterations, this may result in several nodes with only one sampled solution. For such nodes $\mathcal{R}$, $V(\mathcal{R})$ cannot be computed. Thus, we do not include those nodes in the set of sampled nodes, $\mathscr{R}_2$, in the region-layer

ALGORITHM 4: Projection and Hyperparameters Update

Input : $n_r, r, \mathscr{P}, \bar{Y}(\mathscr{X}_2), S^2(\mathscr{X}_2)$
1 **while** $|\{\mathcal{R} \in \mathscr{R}_2 : |\mathscr{X}_2(\mathcal{R})| \geq 2\}| < n_r$ **do**
2 **if** $\exists \mathcal{R}$ *such that* $|\mathscr{X}_2(\mathcal{R})| < 2$ **then**
3 Select $2 - |\mathscr{X}_2(\mathcal{R})|$ new solutions via space filling.
4 **else**
5 Randomly select $\mathcal{R} \in \{\mathcal{R} \in \mathscr{R}_1 : |\mathscr{X}_2(\mathcal{R})| = 1\}$.
6 Select one additional solution not in $\mathscr{X}_2(\mathcal{R})$.
7 **end**
8 Run r replications and compute $\bar{Y}(\mathbf{x})$ and $S^2(\mathbf{x})$ at each.
9 **end**
10 Compute $\bar{Z}(\mathcal{R})$ and $V(\mathcal{R})$ from Equations (6) and (8), respectively, for all $\mathcal{R} \in \mathscr{R}_2$ with $|\mathscr{X}_2(\mathcal{R})| \geq 2$.
11 Solve the region-layer MLE problem in Equation (20) to find $\hat{\boldsymbol{\tau}}$ and compute $\hat{\beta}$ from Equation (19)
12 Find $\tilde{\mathbf{x}} = \arg\min_{\mathbf{x} \in \mathscr{X}_2} \bar{Y}(\mathbf{x})$ and $\mathcal{R}_{\min} = \{\mathcal{R} \in \mathscr{R} : \tilde{\mathbf{x}} \in \mathscr{X}(\mathcal{R})\}$.
13 Solve the solution-layer MLE problem in Equation (21) to find $\hat{\boldsymbol{\theta}}^s$ within $\mathcal{R}_{\min}$.
Output: $\hat{\beta}, \hat{\boldsymbol{\tau}}, \hat{\boldsymbol{\theta}}^s, \bar{Y}(\mathscr{X}_2), S^2(\mathscr{X}_2), \bar{Z}(\mathscr{R}_2)$ and $V(\mathscr{R}_2)$.

MLE calculation. Step 1 of Algorithm 4 checks whether there are at least n_r nodes under $\mathscr{P}$ that contain two or more simulated solutions. If not, we randomly select region-layer nodes among those with only one simulated solution and sample an extra solution for each region-layer node to ensure that $V(\cdot)$ can be computed.

We establish global convergence of pGMIA under Assumption 1 stated below.

Assumption 1. *For all* $\mathbf{x} \in \mathscr{X}$, $y(\mathbf{x}) > -\infty$ *and* $0 < \mathrm{Var}[Y(\mathbf{x})] < +\infty$. *Moreover, the MLEs of the GMRF hyperparameters are updated only finitely many times for each* $\mathscr{P}$ *throughout the run.*

Note that $Y(\mathbf{x})$ need not be normally distributed as long as its variance is finite. The last condition ensures that the hyperparameters are fixed after a finite number of iterations. Theorem 4.1 guarantees that pGMIA converges to the global optimum almost surely regardless of the projection selection criterion. See Appendix **??** for its proof.

Theorem 4.1. *Under Assumption 1, the pGMIA run without stopping converges to the global optimum almost surely.*

4.2 pGMIA+: Multi-layer Extension of pGMIA

As the problem dimension increases, the two-layer model adopted in the pGMIA eventually runs into the computational limitation as the number of solutions/nodes in each layer becomes too large. In this section, we propose a multi-layer extension of pGMIA, pGMIA+, which partitions the dimensions of the solution space into $m > 2$ batches to achieve computational efficiency for even-higher-dimensional problems.

Let the number of dimensions included in the jth group of the partition be d_j so that $\sum_{j=1}^{m} d_j = d$. Without loss of generality, let us label the dimensions in the jth group as $\sum_{i=1}^{j-1} d_i + 1, \sum_{i=1}^{j-1} d_i + 2, \ldots, \sum_{i=1}^{j} d_i$ for $j = 1, 2, \ldots, m$. Each layer of the multi-layer GMRF model is constructed by hierarchically projecting the dimensions of the solution space onto a lower-dimensional integer lattice. The mth-layer graph is the top-layer graph where all dimensions are projected to the last d_m dimensions. Each mth-layer node has unique d_m-dimensional coordinates. For $2 \leq j \leq m$, each jth-layer node can be mapped to the $(j-1)$th-layer graph consisting of the $(j-1)$th-layer nodes projected to that jth-layer node.

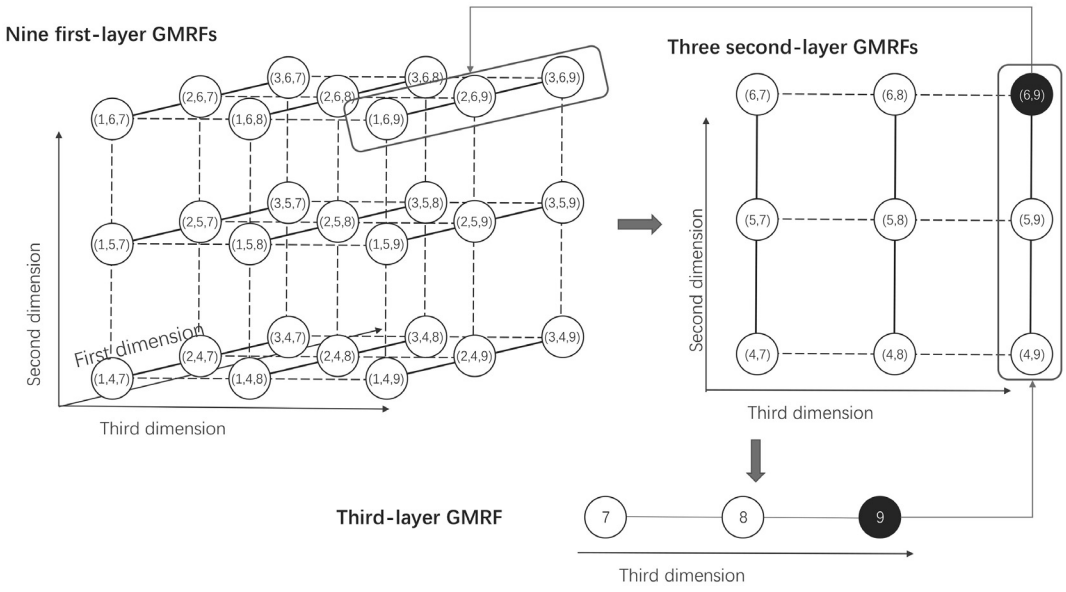


Fig. 2. Illustration of a three-layer GMRF; hierarchical projection (⇒) and search (←).

We illustrate the m-layer projection scheme in Figure 2 using a three-dimensional solution space consisting of 27 solutions, where the first dimension takes values from 1 to 3, the second from 4 to 6, and the third from 7 to 9. Suppose that the dimensions are partitioned into three batches: $\{1\}, \{2\}$, and $\{3\}$. Therefore, the top (third) layer consists of the third dimension only, resulting in a one-dimensional graph with three nodes. Note that the third-layer node with coordinate 9 is mapped to the second-layer graph with three nodes, $(4, 9)$, $(5, 9)$, and $(6, 9)$, whose third coordinates are 9. Similarly, other nodes in the third layer also have corresponding second-layer graphs. Thus, three second-layer graphs are defined in total. The figure also shows that the second-layer node $(6, 9)$ is mapped to the first-layer graph consisting of $(1, 6, 9)$, $(2, 6, 9)$, and $(3, 6, 9)$. There are total 9 first-layer graphs.

The pGMIA+ defines the response at each jth-layer node as the average of the responses at all nodes in the corresponding $(j-1)$th-layer graph. For $1 \leq j \leq m$, the jth-layer GMRF is defined for each jth-layer graph. As in the pGMIA, the pGMIA+ selects a node to sample starting at the mth layer and continues its way down to the first layer by selecting a node in the lower-layer graph. For instance, in Figure 2, suppose that 9 is selected for sampling in the third layer. Then, the pGMIA+ continues to make a sampling decision on the second layer by selecting a node among $(4, 9)$, $(5, 9)$, and $(6, 9)$. This procedure is repeated in the first layer, until a solution among $(1, 6, 9)$, $(2, 6, 9)$, and $(3, 6, 9)$ is selected. Hence, while there are 27 feasible solutions in the single-layer GMRF, the pGMIA+ only needs to make inference at three nodes on each layer, which is why the computational cost is significantly reduced in the pGMIA+.

Similar to the two-layer case, the MLEs of GMRF hyperparameters can be computed hierarchically from the m-th layer to the first layer, exploiting Theorem 2.1 and Proposition 2.3. The details of multi-layer GMRF distributions, their MLE problems, and the pGMIA+ are presented in Appendix **??**. Corollary 4.2 states the global convergence of pGMIA+; its proof can be found in Appendix **??**.

Corollary 4.2. *Under Assumption 1, the pGMIA+ run without stopping converges to the global optimum almost surely.*

In practice, making exact inference on all layers may be too expensive when d is large. One way to reduce the computational overhead is to make exact inferences on the top $m-1$ layers while randomly sampling a solution to simulate in the first layer. The computational saving can be made substantial by setting d_1 large.

5 EMPIRICAL ANALYSIS

In this section, we provide extensive empirical results to demonstrate the performances of the pGMIA and pGMIA+. Both algorithms are compared with MR-GMIA in Salemi et al. [2019] and four state-of-the-art high-dimensional BO algorithms: **Random EMbedding Bayesian Optimization (REMBO)** in Wang et al. [2016], **Sparse Axis-Aligned Subspace Bayesian Optimization (SAASBO)** in Eriksson and Jankowiak [2021], and **High-Dimensional Bayesian Optimization (HDBO)** and **High-Dimensional Batch Bayesian Optimization (HDBBO)** in Wang et al. [2017]. REMBO and SAASBO are projection-based BO algorithms. While SAASBO only considers an axis-aligned projection, both adopt the **Expected Improvement (EI)** as the sampling criterion. HDBO and HDBBO are batching approaches. Both apply the **Upper Confidence Bound (UCB)** as the sampling criterion. However, HDBBO has parameter B, which controls the number of solutions sampled in each iteration; it selects B solutions with the largest UCB values. Although their inference is still valid, REMBO, SAASBO, HDBO, and HDBBO are not designed specifically for the integer solution space. When a fractional solution is chosen from these algorithms, we round it to the nearest integer solution for simulation.

5.1 Performance of pGMIA

We test two choices of projection selection criteria with the pGMIA, random selection and distance correlation. To differentiate the two criteria, we denote them as pGMIA-R and pGMIA-DC, respectively. We demonstrate the performance of pGMIA-R and pGMIA-DC using three popular optimization test functions (https://www.sfu.ca/~ssurjano/optimization.html): Zakharov (Section 5.1.1), Branin (Section 5.1.2), and Styblinski-Tang (Section 5.1.3) functions. To convert them to DOvS problems, we add stochastic noise to the function values. All dimensions of Zakharov function are active and it is non-decomposable. The Branin function has a two-dimensional active subspace; thus, it is favorable to projection-based methods such as REMBO and SAASBO. The Styblinski-Tang function is decomposable in each coordinate direction; thus, it is favorable to batching methods such as HDBO and HDBBO. We also include a slightly modified Styblinski-Tang function to induce interactions between the dimensions.

All three test problems are 10-dimensional and there are 5 feasible values in each dimension. We set $d_1 = d_2 = 5$ for pGMIA-R and pGMIA-DC. We choose the last 5 dimensions to be included in the region layer for MR-GMIA and set the dimension of the active subspace of REMBO to be 5. Since Wang et al. [2017] do not give specific guidance on the batch size, we choose $B = 6$ for HDBBO as pGMIA-R and pGMIA-DC typically sample 6 solutions at each iteration.

To estimate the hyperparameters, all algorithms initially sample 100 design solutions via space filling for the Zakharov and Styblinski-Tang functions. In pGMIA-R, pGMIA-DC and MR-GMIA, $n_r = 10$ region-layer nodes and $n_s = 10$ solutions per node are selected via space filling. For the Branin function, all algorithms initially sample 9 design solutions (n_r=3 and $n_s = 3$ for pGMIA-R, pGMIA-DC, and MR-GMIA). Note that the Branin function has two active dimensions. Therefore, we choose a small initial design to avoid finding the optimal solution right after sampling the initial design. All algorithms run 10 replications each time they sample a solution.

In pGMIA-R and pGMIA-DC, we compute the MLEs from the outputs of 50 most-sampled region-layer nodes to avoid sampling too many new solutions when the projection scheme is updated and to save the computational cost.

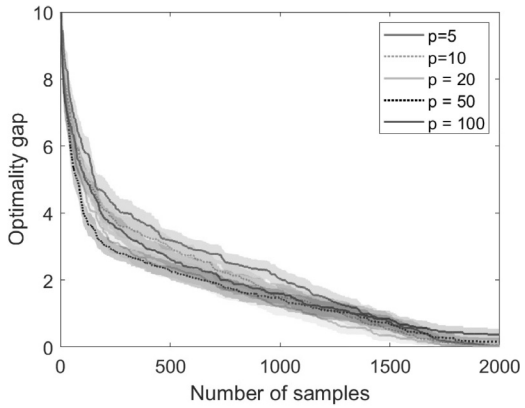


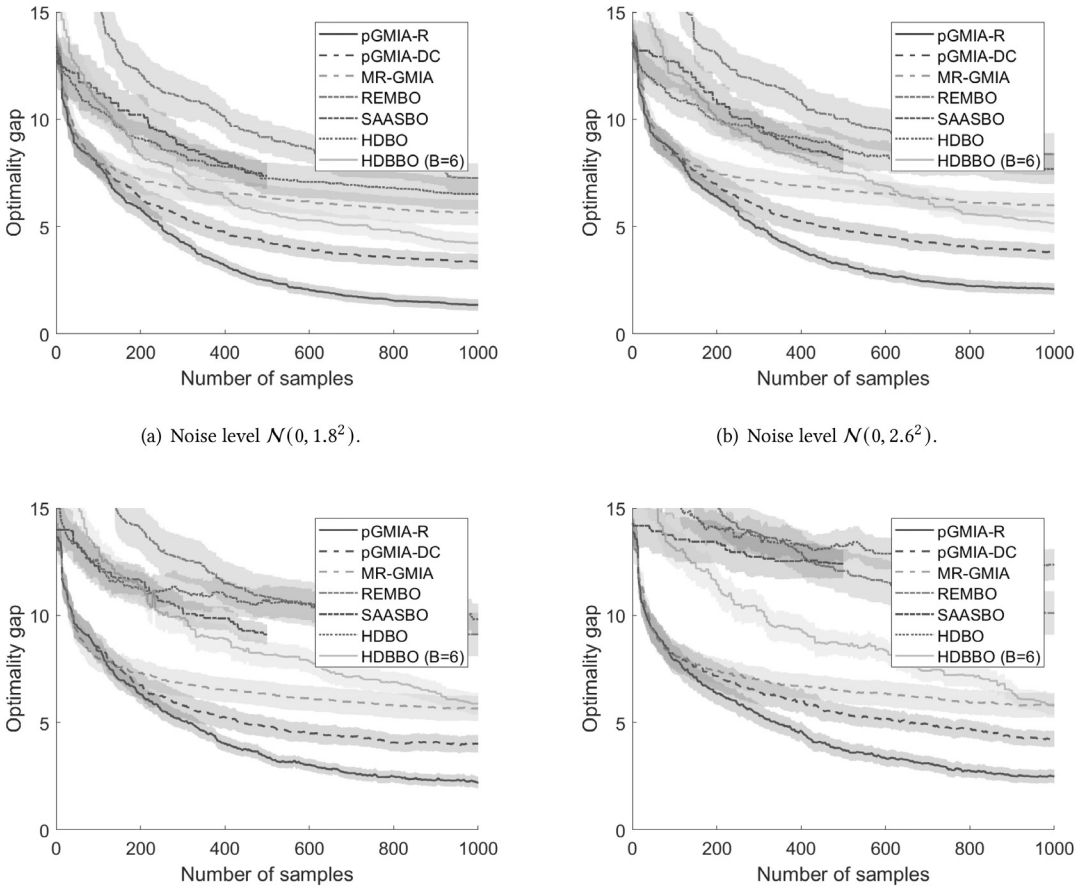
Fig. 3. Sensitivity analysis for pGMIA-R on the choice of p; 10-dimensional Zakharov function.

5.1.1 Zakharov Function. Given $\mathbf{x} = (x_1, x_2, \ldots, x_{10})^\top$, the 10-dimensional Zakharov function is $f(\mathbf{x}) = \sum_{i=1}^{10} x_i^2 + \left(\sum_{i=1}^{10} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{10} 0.5ix_i\right)^4$. The Zakharov function is neither additive nor has a lower-dimensional subspace; all ten dimensions are active and there is an interaction term between any two dimensions. These features make all four benchmarking algorithms not particularly advantageous for the Zakharov function. We choose $\mathscr{X} = \{-2, -1, 0, 1, 2\}^{10}$ as the feasible solution space, which makes $f(\mathbf{x}) \in [0, 9.15 \times 10^6]$. The global optimum of the Zakharov function is at $\mathbf{x} = \mathbf{0}_{10}$ with response 0, where the difference between the responses at the global minimum and the second best is 1.3. To make the output stochastic, we add normal noise that follows $\mathcal{N}(0, 1.8^2)$ to the response function.

We first examine the sensitivity of the pGMIA to the choice of p before comparing it with the benchmarking algorithms. Figure 3 displays the trajectories of optimality gap against the number of samples pGMIA-R evaluates averaged over 100 macro-runs; recall that each sample consists of 10 simulation replications. The shaded area around each curve shows point-wise ±2 standard error of the average performance computed from 100 macro-runs. Although $p = 5$ and $p = 100$ show slightly less desirable behaviors in the beginning and toward the end, respectively, the choice of p does not appear to affect the pGMIA's performance significantly. As more solutions are sampled, for all five choices of p, pGMIA-R eventually converges to the global optimum. The same sensitivity analysis was conducted for pGMIA-DC and showed similar conclusions.

In the remainder of Section 5, we adopt $p = 20$ for pGMIA-R and pGMIA-DC. To match the pGMIA, we also set REMBO to update its parameters every 20 iterations. Additionally, we let SAASBO, HDBO, and HDBBO update the projection/decomposition and parameters every 20 iterations. The MR-GMIA only computes the parameters once at the beginning of the algorithm.

Figure 4(a) compares the performance of pGMIA-R and pGMIA-DC with all other algorithms. Notably, pGMIA-R converges faster than pGMIA-DC, indicating that randomly selecting the projection is effective. The performance difference between pGMIA-DC and pGMIA-R can be attributed to the frequency at which each algorithm updates the projection. For the experiments reported in Figure 4(a), pGMIA-R and pGMIA-DC evaluate the projection update criterion 19 times by the 1,000th sample. While pGMIA-R updates the projection 18.8 times on average across 100 macro-runs (standard error 0.037), pGMIA-DC updates only 14.2 times (0.397), indicating that the same projection may be chosen multiple times by the distance correlation criterion. Early on in the algorithm, there is a benefit of changing the projection more frequently as it induces more exploration in all dimensions. Moreover, pGMIA-DC estimates the distance correlations based on the posterior means obtained from the GMRF



(a) Noise level $\mathcal{N}(0, 1.8^2)$.

(b) Noise level $\mathcal{N}(0, 2.6^2)$.

(c) Noise level $\mathcal{N}(0, 3.9^2)$.

(d) Noise level $\mathcal{N}(0, 5.2^2)$.

Fig. 4. Comparison of the trajectories of optimality gap of the Zakharov function with different stochastic noises for pGMIA-R, pGMIA-DC, MR-GMIA, REMBO, SAASBO, HDBO and HDBBO ($B = 6$) averaged over 100 macro-runs.

model as discussed in Section 4.1, which tends to have large prediction errors in the earlier iterations.

As expected, REMBO, SAASBO, HDBO, and HDBBO perform poorly on the Zakharov function, which is not additive and all dimensions are active. We stopped running the SAASBO around 500 samples because it takes more than 2 hours at each iteration at that point, and the trajectory already demonstrates the efficiency of pGMIA over SAASBO. Additionally, we observe that the MR-GMIA, which uses a fixed projection scheme, progresses slowly after 200 samples. This shows that updating the projection scheme is indeed effective in narrowing the optimality gap.

Next, we examine robustness of the pGMIA and benchmarking algorithms to different levels of stochastic noise. In Figures 4(b) to 4(d), we increase the stochastic noise variance from 1.8^2 to $2.6^2, 3.9^2$, and 5.2^2, respectively. Observe that the trajectories of pGMIA-R, pGMIA-DC, and MR-GMIA are similar across all four cases, suggesting that these algorithms perform consistently under different noise conditions. On the contrary, REMBO, SAASBO, HDBO and HDBBO show noticeable performance degradation with increasing noise levels. In particular, the HDBBO starts outperforming the MR-GMIA around the 300-sample mark when the stochastic noise

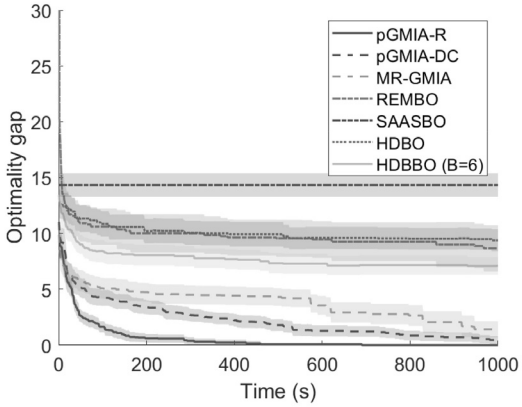


Fig. 5. Comparison of the trajectories of optimality gap of the Zakharov function versus wall-clock time for pGMIA-R, pGMIA-DC, MR-GMIA, REMBO, SAASBO, HDBO and HDBBO ($B = 6$) averaged over 30 macro-runs.

variance is 1.8^2, but the cross-over point increases to 900 samples when stochastic noise variance is 5.2^2.

To examine the computational overhead of the algorithms, Figure 5 plots the average trajectories of the optimality gap against the wall-clock time computed from 30 macro-runs, where the stochastic noise variance is 1.8^2. All algorithms are run on a single thread of Intel Core i5-9600K CPU (3.70 GHz) with 16.0 GB RAM to measure the wall-clock time. Observe that pGMIA-R shows the fastest progress and clearly dominates the others. The performance difference between pGMIA-R and pGMIA-DC can be attributed to the cost of computing the distance correlations in pGMIA-DC. All three variations of the GMIA outperform the others in wall-clock time, reflecting the computational benefit of GMRF-based algorithms over the GP-based algorithms applied to the integer solution space. SAASBO shows no improvement over the 1,000-second period due to its heavy computational overhead.

To summarize, the experiment results in this section demonstrate the robustness of the pGMIA to different levels of stochastic noise and highlight the superior sample and computational efficiency of the pGMIA on a non-additive objective function for which all dimensions are active compared with other batching or projection-based BO algorithms.

5.1.2 Branin Function. For $\mathbf{x} = (x_1, x_2, \ldots, x_{10})^\top$, the 10-dimensional Branin function is

$$f(\mathbf{x}) = \left(\bar{x}_2 - \frac{5.1}{4\pi^2}\bar{x}_1^2 + \frac{5}{\pi}\bar{x}_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\cos(\bar{x}_1)\right), \tag{23}$$

where $\bar{x}_1 = 10x_1 - 5$, $\bar{x}_2 = 10x_2$. Observe that only the first two dimensions of $\mathbf{x}$ are active in Equation (23), which benefits REMBO and SAASBO. We choose $\mathscr{X} = \{0, 0.25, 0.5, 0.75, 1.0\}^{10}$ as the feasible solution space, which makes $f(\mathbf{x}) \in [2.42, 308.13]$. The global optimum of the Branin function is $(x_1, x_2) = (0.75, 0.25)$ with response 2.4153. The responses at the global minimum and the second best differ by 0.5. We add $\mathcal{N}(0, 0.7^2)$ stochastic noise to the response function.

Figure 6 shows the trajectories of optimality gap averaged over 100 macro-runs when the algorithms are terminated after 500 samples. Although both pGMIA-R and pGMIA-DC are slower in the beginning, they outperform all other algorithms after 120 samples. REMBO makes reasonably good progress at the beginning since the Branin function has a two-dimensional active subspace. However, it significantly slows down after about 100 samples. On the other

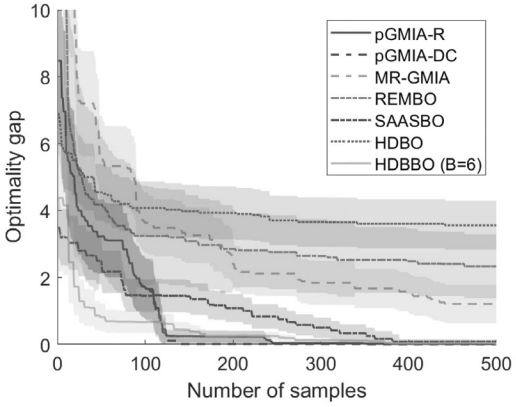


Fig. 6. Comparison of the trajectories of optimality gap of the Branin function versus the number of samples for pGMIA-R, pGMIA-DC, MR-GMIA, REMBO, SAASBO, HDBO, and HDBBO ($B = 6$) averaged over 100 macro-runs.

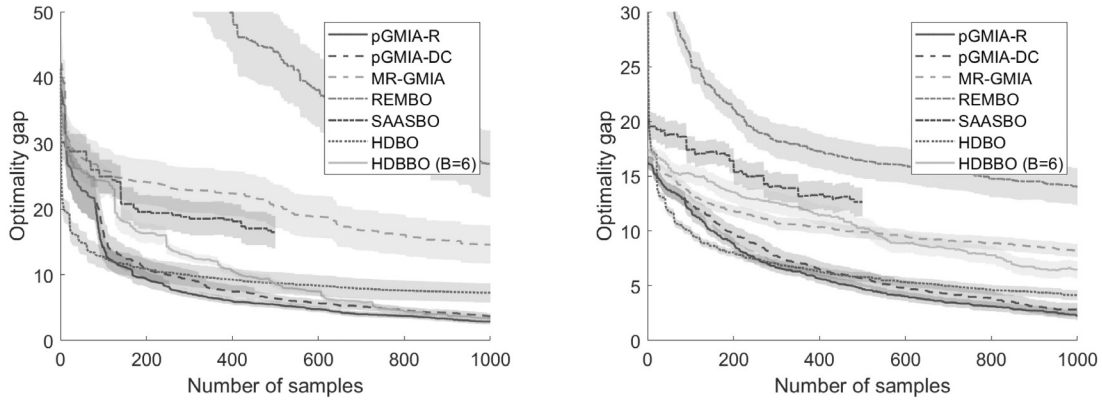
hand, SAASBO converges to the global optimal after 400 samples. For MR-GMIA, the first five dimensions in the region layer include the two active dimensions of the Branin function. Therefore, all solutions within a region in MR-GMIA have identical responses, which makes the solution-layer search meaningless. HDBO shows good initial performance but slows down after 100 samples, exhibiting the worst performance among all. On the other hand, HDBBO is almost as competitive as pGMIA-R and pGMIA-DC. We conjecture that the batch sampling scheme of HDBBO increases its chance to explore more along the active dimensions.

We close this section by pointing out that even if the true objective function has a lower-dimensional active subspace, the pGMIA outperforms the benchmarked active subspace-based BO algorithms.

5.1.3 Styblinski-Tang Function. Given $\mathbf{x} = (x_1, x_2, \ldots, x_{10})^\top$, the 10-dimensional (scaled) Styblinski-Tang function, $f(\mathbf{x}) = \frac{1}{20}\sum_{i=1}^{10}(x_i^4 - 16x_i^2 + 5x_i)$, is separable in each dimension. The feasible solution space is set to be $\mathscr{X} = \{-6, -3, 0, 3, 6\}^{10}$, which makes $f(\mathbf{x}) \in [-39, 375]$ and $\mathbf{x} = (-3)\mathbf{1}_{10}$ the global optimum with response -39. The difference between the responses at the global minimum and the second best is 2. We add $\mathcal{N}(0, 3^2)$ stochastic noise to the response function.

In the following, we compare the performance of pGMIA-R and pGMIA-DC tested on the Styblinski-Tang function against MR-GMIA, REMBO, SAASBO, HDBO, and HDBBO. Figure 7(a) shows the trajectories of the optimality gap against the number of samples averaged over 100 macro-runs. Note that HDBO exhibits the best performance at the beginning of the algorithm. The pGMIA-R and pGMIA-DC outperform HDBO after the 300-sample mark—this is when the projection and MLEs are updated for the first time. Until the first projection update, pGMIA-R and pGMIA-DC are identical, but thereafter, pGMIA-DC shows slower progress than pGMIA-R. HDBBO is not competitive at the beginning of the algorithm, but catches up the pGMIA-R and pGMIA-DC in the long run. REMBO's and SAASBO's poor performances are not surprising, as all dimensions are active in the Styblinski-Tang function. MR-GMIA performs better than REMBO but not as well as the other four algorithms. In addition, both MR-GMIA and REMBO do not update the initial projection scheme, which explains their slower convergence.

To examine the effect of interactions across dimensions on the algorithms' performances, we modify the Styblinski-Tang function to $f(\mathbf{Ax})$, where $\mathbf{A} = [a_{i,j}]$ is a 10×10 matrix whose elements are zeroes except that $a_{i,i} = a_{i,i+1} = 0.5$ for $1 \leq i \leq 9$ and $a_{10,1} = a_{10,10} = 0.5$. Given the



(a) Styblinski-Tang function.

(b) Modified Styblinski-Tang function.

Fig. 7. Comparison of the trajectories of optimality gap against the number of samples averaged across 100 macro-runs on the original Styblinski-Tang and modified Styblinski-Tang functions for pGMIA-R, pGMIA-DC, MR-GMIA, REMBO, HDBO, and HDBBO ($B = 6$).

same $\mathscr{X}$, the global optimum is at $\mathbf{x} = (-3)\mathbf{1}_{10}$ and its response remains the same. Figure 7(b) shows the results for the modified Styblinski-Tang function. All algorithms except for HDBBO show smaller optimality gap at a given number of samples compared with the original function while pGMIA-R and pGMIA-DC still exhibit the best performances among all. The relatively poor performance of the HDBBO can be attributed to the fact that the function is no longer additive. Its sample performance is affected more than the HDBO's, as it selects a batch of solutions assuming the additive structure. Not surprisingly, the two active-subspace-based algorithms, REMBO and SAASBO, still show poor performances.

In both Figures 7(a) and 7(b), there are no significant differences between the performances of pGMIA-DC and pGMIA-R. This can be attributed to the fact that all dimensions in the Styblinski-Tang function are equally important. As a result, the marginal distance correlations of all dimensions are identical.

5.2 Performance of pGMIA+

In this section, we test the 100-dimensional Branin (Section 5.2.1) and Zakharov (5.2.2) functions, and a 30-dimensional Assemble-to-Order DOvS problem (5.2.3) to demonstrate the performance of pGMIA+. We only test pGMIA+ with random selection as the projection selection criterion because (i) it shows better performance when combined with pGMIA in Section 5.1 and (ii) to avoid computing the distance correlations for all dimensions. The MR-GMIA is dropped here as the problem size is too large for it to be computationally feasible.

In Sections 5.2.1 and 5.2.2, we adopt a three-layer GMRF model for the pGMIA+, where $d_3 = 3$, $d_2 = 3$, and $d_1 = 94$. With this setting, the pGMIA+ samples 8 to 19 solutions at each iteration, which depends on whether the CEI-maximizing or sample-best node at each layer coincides with the node that the sample-best solution is projected to. See Appendix **??** for the details. We set the active subspace dimension of REMBO to be $d_2 + d_3 = 6$ and adopt $B = 18$ for HDBBO, as pGMIA+ typically samples 18 solutions at each iteration. All algorithms are initialized with the same number of design solutions selected via Latin hypercube sampling, and simulate $r = 10$ replications per sampling. As in Section 5.1, we select (at most) 50 most-sampled nodes for MLE estimation.

In Section 5.2.3, we change the setting for the pGMIA+ to $d_3 = 3$, $d_2 = 3$, and $d_1 = 24$.

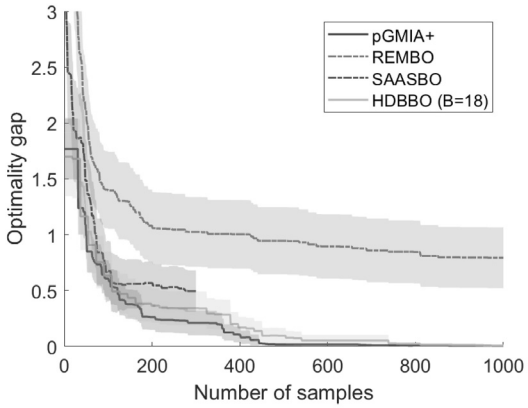


Fig. 8. Comparison of the trajectories of optimality gap of the 100-dimensional Branin function versus the number of samples for pGMIA+, REMBO, SAASBO, and HDBBO ($B = 18$) averaged over 100 macro-runs.

5.2.1 Branin Function. The 100-dimensional Branin function is the same as Equation (23) except that $\mathbf{x} = (x_1, x_2, \ldots, x_{100})^\top$. We choose $\mathscr{X} = \{0, 0.1, 0.2, \ldots, 1\}^{100}$, which makes $f(\mathbf{x}) \in [0.64, 308.13]$. Any solution with $(x_1, x_2) = (0.8, 0.2)$ is a global optimum of the 100-dimensional Branin function with response 0.6445. The difference between the responses at the global minimum and the second best is 0.2. We add $\mathcal{N}(0, 0.6^2)$ stochastic noise to the response function. All algorithms initially sample 32 design solutions ($n_3 = 4$, $n_2 = 4$, and $n_1 = 2$ for pGMIA+) for hyperparameter estimation.

Figure 8 shows the trajectories of the optimality gap averaged over 100 macro-runs. Note that pGMIA+ outperforms REMBO and SAASBO. Again, the observation for SAASBO is cut short due to its prohibitively large computational cost. The pGMIA+ converges to the global optimum after 500 samples while HDBBO converges after 800 samples. Unlike in the 10-dimensional Branin function, REMBO performs significantly worse than pGMIA+, HDBBO, and SAASBO. This is because the random projection that REMBO adopts tends to project the solution selected in the 6-dimensional active subspace to outside of the feasible region of the 100-dimensional solution space. A similar observation has been reported by Letham et al. [2020].

5.2.2 Zakharov Function. The 100-dimensional Zakharov function is $f(\mathbf{x}) = \sum_{i=1}^{100} x_i^2 + \left(\sum_{i=1}^{100} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{100} 0.5 i x_i\right)^4$, where $\mathbf{x} = (x_1, x_2, \ldots, x_{100})^\top$. All dimensions are active in this test function. We choose $\mathscr{X} = \{-5, -4, \ldots, 5\}^{100}$, which makes $f(\mathbf{x}) \in [0, 2.54 \times 10^{16}]$. The global optimum of the Zakharov function is at $\mathbf{x} = \mathbf{0}_{100}$ with response 0, where the difference between the responses at the global minimum and the second optimal solution is 1.3. We add $\mathcal{N}(0, 1.8^2)$ stochastic noise to the response function. All algorithms initially sample 200 design solutions ($n_3 = 10$, $n_2 = 10$ and $n_1 = 2$ for pGMIA+) for hyperparameter estimation.

Figure 9 shows the trajectories of true optimality gap averaged over 100 macro-runs when the algorithms are terminated after 1,000 samples. In the first 400 samples, it is difficult to distinguish the four algorithms due to large run-to-run variation. We stopped running SAASBO after 300 samples because its iteration takes more than one hour at this point. After 400 iterations, we can observe that pGMIA+ shows better performance than the other two algorithms. Although it appears that the pGMIA+'s progress slows down as the algorithm proceeds, we note that the remaining optimality gap (approximately $1,000$) is negligible compared with the range of the function, 2.54×10^{16}.

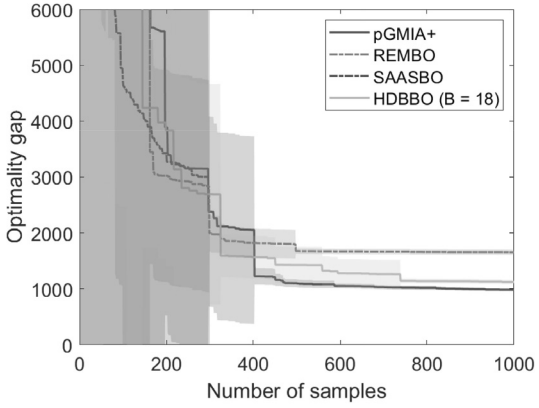


Fig. 9. Comparison of the trajectories of optimality gap of the 100-dimensional Zakharov function versus the number of samples for pGMIA+, REMBO, SAASBO, and HDBBO ($B = 18$) averaged over 100 macro-runs.

Table 1. Parameters for the kth Product Group for the ATO System

Component	h_{ik}	μ_{ik}	σ_{ik}	C_{ik}
$I1$	$2/k$	0.15	0.0225	20
$I2$	$2/k$	0.40	0.06	20
$I3$	$2/k$	0.25	0.0375	20
$I4$	$2/k$	0.15	0.0225	20
$I5$	$2/k$	0.25	0.0375	20
$I6$	$2/k$	0.08	0.0120	20

Product	λ_{jk}	$p_{jk}(\$)$	I_{1k}	I_{2k}	I_{3k}	I_{4k}	I_{5k}	I_{6k}
J_{1k}	3.6	$16/k$	0	1	0	1	0	1
J_{2k}	3.0	$17/k$	0	1	1	1	0	1
J_{3k}	2.4	$18/k$	1	0	0	1	0	1
J_{4k}	1.8	$19/k$	1	0	1	0	1	1
J_{5k}	1.2	$20/k$	1	1	1	1	1	1

5.2.3 Assemble-to-order (ATO) System. The **assemble-to-order (ATO)** system [Hong and Nelson 2006] is an inventory management example, where the final product is assembled from the components made to stock whenever an order is received. The components required for each product vary. We modify the ATO problem to have 5 product groups, $1 \le k \le 5$, where each has 5 product types, $J_{1k}, J_{2k}, \ldots, J_{5k}$, assembled from 6 components, $I_{1k}, I_{2k}, \ldots, I_{6k}$. The objective is to determine the expected profit-maximizing inventory levels for $I_{1k}, I_{2k}, \ldots, I_{6k}, 1 \le k \le 5$, which is a 30-dimensional DOvS problem.

The system applies a continuous-review base-stock policy, that is, whenever there is a demand for a component, the system automatically starts producing the component to fill up to the base-stock level. For the ith component of the kth product group, the production time follows a truncated normal distribution in $(0, +\infty)$ with mean μ_{ik} and variance σ_{ik}^2. For each k, there are only 2 machines producing the components on the first-in-first-out basis. Each component has the inventory capacity, C_{ik}, and the holding cost per period, h_{ik}.

The arrival process of orders of the jth product of the kth product group follows a Poisson distribution with rate λ_{jk}, for $j = 1, 2, \ldots, 5$ and $k = 1, 2, \ldots, 5$. When a product order is received, if any of its required components are out of stock, the order is canceled with a fixed penalty $r = \$3/k$. Otherwise, the order is assembled immediately and generates profit p_{jk}. Table 1 displays the parameters for the ATO system for the kth product group.

In each replication, 20 warm-up periods are adopted, and the per-period average profit is calculated from the following 50 periods. We set $\mathscr{X} = \{0, 2, 4, \ldots, 20\}^{30}$, which results in 11^{30} feasible solutions. Because the true objective function values are unknown, they are estimated from 40,000 replications at each solution. With this sample size, the largest relative error (standard error divided

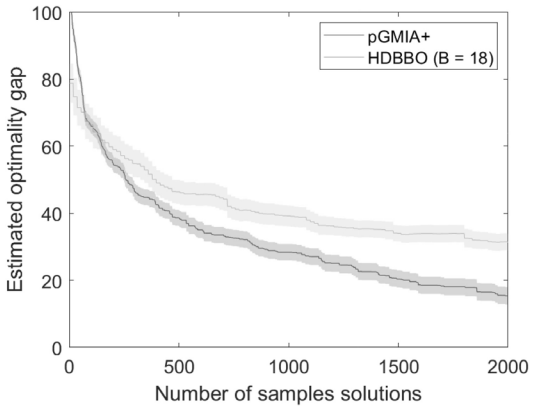


Fig. 10. Comparison of the trajectories of estimated optimality gap of the ATO function at the current best for pGMIA+ and HDBBO ($B = 18$).

by the mean estimate) is 2.87×10^{-4}. The estimated optimum is $(\mathbf{x}_{ik})_{i=1}^{6} = (3, 5, 3, 5, 2, 5)^\top$ for $1 \leq k \leq 5$, with the per-period expected averaged total profit of \$276.56. Additionally, the estimated expected profit of solutions ranges from \$276.56 to −\$237.47, where the negative value indicates loss.

Unlike other examples, the ATO problem has heteroscedastic stochastic noise across the solutions. Thus, we drop REMBO from comparison in this section as it only allows homoscedastic noise. We also drop SAASBO due to its heavy computational cost. Figure 10 shows the trajectories of the estimated optimality gap at the current best solution against the number of samples averaged over 100 macro-runs. Clearly, the pGMIA+ outperforms the HDBBO and the optimality gap widens as the algorithms continue. This indicates effectiveness of the pGMIA+ when applied to a more realistic simulation optimization problem.

6 CONCLUSIONS

The pGMIA reduces computational complexity of a high-dimensional DOvS problem by batching the dimensions into region and solution layers and projecting the solution space onto the region-layer dimensions. The pGMIA hierarchically optimizes each layer; it first selects a region-layer node to simulate, then selects a solution to simulate within the solution-layer graph projected to the region-layer node. Since the dimensions at both region and solution layers are lower than the solution space, the computational overhead of the algorithm is greatly reduced. However, the region-layer GMRF's precision matrix becomes a dense matrix after the projection, which negates the computational benefit of the GMRF in solving a DOvS problem. We resolve this issue by proposing the approximate sparse precision matrix introduced in Theorem 2.1, which becomes increasingly accurate as the number of values in each dimension increases. Exploiting the approximate precision matrix, we also propose novel approaches to estimate the hyperparameters of the region- and solution-layer GMRFs. The pGMIA can be extended to the pGMIA+ to incorporate more layers to even further exploit the computational benefit of projection.

Across all numerical examples, we have observed that the pGMIA with the random projection selection criterion outperforms other benchmarks. We attribute its performance to the fact that the pGMIA simply uses batching and projection as means to reduce the computational burden instead of assuming that the objective function is additive or has a lower-dimensional active subspace. It periodically updates the batches of dimensions by randomization, which allows the interaction effects among dimensions previously included in different batches to be learned. Moreover, the pGMIA does not regard the projected dimensions to be inactive. Once a region-layer sampling decision is made based on the projected GMRF model, it applies a sampling criterion to the

solution-layer model consisting of the projected dimensions in the region layer to complete the sampling decision.

We have also observed that the pGMIA tends to perform better than the GMIA, even for a lower-dimensional problem whose computational overhead is cheap enough for the GMIA. The GMIA fits a single-layer GMRF for the entire solution space and it is harder to obtain an accurate global fit if the solution space is large. The projected GMRF, on the other hand, models the solution-layer graph with a solution-layer GMRF that provides better local inference. Meanwhile, the region-layer GMRF provides a low-resolution inference to quickly find promising regions of the solution space. Consequently, the pGMIA learns local performance of the objective function more effectively at regions with good average performances.

Although the pGMIA+ scales well with the dimension when the number of feasible values in each dimension is relatively small, if the number is large, then it is limited by the size of the precision matrix that can be factorized by a computer. In the latter case, the pGMIA+ may only contain a few dimensions in each batch. This can potentially slow down the algorithm's progress as interaction effects among the dimensions in separate batches may not be captured as effectively until the dimensions are rebatched. Aggregating feasible values in each dimension to obtain a coarser model may help in this case. Similar ideas have been explored by [Mes et al. 2011].

Exploiting parallel computing to improve efficiency of the pGMIA is another important future research topic. As mentioned in Section 4, there is no exchange of information among the solutions projected to different nodes in pGMIA. Thus, the solution-layer search can be run in parallel if multiple regions are selected for sampling. This will require the sampling criterion to be extended to evaluate the benefit of jointly sampling a set of regions as well as a stopping criterion to terminate the parallel search when a region-layer node is deemed not worth exploring anymore.

ACKNOWLEDGMENTS

The authors thank Andreas Wächter for helpful discussions on linear algebraic techniques.

REFERENCES

Ricardo Baptista and Matthias Poloczek. 2018. Bayesian optimization of combinatorial structures. In *Proceedings of the 35th International Conference on Machine Learning*.

Mickaël Binois, David Ginsbourger, and Olivier Roustant. 2020. On the choice of the low-dimensional domain for global optimization via random embeddings. *Journal of Global Optimization* 76, 1 (2020), 69–90.

Mickaël Binois and Nathan Wycoff. 2022. A survey on high-dimensional Gaussian process modeling with application to Bayesian optimization. *ACM Transactions on Evolutionary Learning and Optimization* 2, 2 (2022), 1–26.

Mucahit Cevik, Mehmet Ali Ergun, Natasha K. Stout, Amy Trentham-Dietz, Mark Craven, and Oguzhan Alagoz. 2016. Using active learning for speeding up calibration in simulation models. *Medical Decision Making* 36, 5 (2016), 581–593.

William Gemmell Cochran. 1977. *Sampling Techniques*. New York: Wiley.

Sebastien Da Veiga. 2015. Global sensitivity analysis with dependence measures. *Journal of Statistical Computation and Simulation* 85, 7 (2015), 1283–1305.

Josip Djolonga, Andreas Krause, and Volkan Cevher. 2013. High-dimensional Gaussian process bandits. In *Advances in Neural Information Processing Systems*. 1025–1033.

David Eriksson and Martin Jankowiak. 2021. High-dimensional Bayesian optimization with sparse axis-aligned subspaces. In *Uncertainty in Artificial Intelligence*. PMLR, 493–503.

Eduardo C. Garrido-Merchán and Daniel Hernández-Lobato. 2020. Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes. *Neurocomput.* 380, C (2020), 20–35.

James Hensman, Alexander G. Matthews, and Zoubin Ghahramani. 2015. Scalable variational Gaussian process classification. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, Vol. 38. 351–360.

Michael Hoffman, Eunhye Song, Michael Brundage, and Soundar Kumara. 2018. Condition-based maintenance policy optimization using genetic algorithms and Gaussian Markov improvement algorithm. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society*.

L. Jeff Hong and Barry Nelson. 2006. Discrete optimization via simulation using COMPASS. *Operations Research* 54 (2006), 115–129.

Donald R. Jones, Matthias Schonlau, and William J. Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization* 13, 4 (1998), 455–492.

Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. 2015. High dimensional Bayesian optimisation and bandits via additive models. In *International Conference on Machine Learning*. 295–304.

Benjamin Letham, Roberto Calandra, Akshara Rai, and Eytan Bakshy. 2020. Re-examining linear embeddings for high-dimensional Bayesian optimization. In *Advances in Neural Information Processing Systems*, Vol. 33. 1546–1558.

Xinru Li and Eunhye Song. 2020. Smart linear algebraic operations for efficient Gaussian Markov improvement algorithm. In *Proceedings of the 2020 Winter Simulation Conference*. 2887–2898.

Xiaoyu Lu, Javier Gonzalez, Zhenwen Dai, and Neil Lawrence. 2018. Structured variationally auto-encoded optimization. In *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80. 3267–3275.

Logan Mathesen, Kaushik Keezhnagar Chandrasekar, Xinsheng Li, Giulia Pedrielli, and K. Selçuk Candan. 2019. Subspace communication driven search for high dimensional optimization. In *Proceedings of the 2019 Winter Simulation Conference*. 3528–3539.

Martijn R. K. Mes, Warren B. Powell, and Peter I. Frazier. 2011. Hierarchical knowledge gradient for sequential sampling. *Journal of Machine Learning Research* 12, 90 (2011), 2931–2974.

Riccardo Moriconi, Marc P. Deisenroth, and K. S. Sesh Kumar. 2020. High-dimensional Bayesian optimization using low-dimensional feature spaces. *Machine Learning* 109 (2020), 1925–1943.

Mojmir Mutny and Andreas Krause. 2018. Efficient high dimensional Bayesian optimization with additivity and quadrature Fourier features. In *Advances in Neural Information Processing Systems 31*. 9005–9016.

Changyong Oh, Jakub M. Tomczak, Efstratios Gavves, and Max Welling. 2019. Combinatorial Bayesian optimization using the graph Cartesian product. In *Proceedings of 33rd Conference on Neural Information Processing Systems*.

Paul Rolland, Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. 2018. High-dimensional Bayesian optimization via additive models with overlapping groups. In *International Conference on Artificial Intelligence and Statistics*. 298–307.

Olivier Roustant, Espéran Padonou, Yves Deville, Aloïs Clément, Guillaume Perrin, Jean Giorla, and Henry Wynn. 2020. Group kernels for Gaussian process metamodels with categorical inputs. *SIAM/ASA Journal on Uncertainty Quantification* 8, 2 (2020), 775–806.

Havard Rue and Leonhard Held. 2005. *Gaussian Markov Random Fields: Theory and Applications*. New York: Chapman and Hall/CRC.

Peter Salemi, Eunhye Song, Barry L. Nelson, and Jeremy Staum. 2019. Gaussian Markov random fields for discrete optimization via simulation: Framework and algorithms. *Operations Research* 67, 1 (2019), 250–266.

Mark Semelhago, Barry L. Nelson, Eunhye Song, and Andreas Wächter. 2021. Rapid discrete optimization via simulation with Gaussian Markov random fields. *INFORMS Journal on Computing* 33, 3 (2021), 915–930.

Mark Semelhago, Barry L. Nelson, Andreas Wächter, and Eunhye Song. 2017. Computational methods for optimization via simulation using Gaussian Markov random fields. In *Proceedings of 2017 Winter Simulation Conference*. 2080–2091.

Eunhye Song and Yi Dong. 2018. Generalized method of moments approach to hyperparameter estimation for Gaussian Markov random fields. In *Proceedings of 2018 Winter Simulation Conference*. 1790–1801.

Lihua Sun, L. Jeff Hong, and Zhaolin Hu. 2014. Balancing exploitation and exploration in discrete optimization via simulation through a Gaussian process-based search. *Operations Research* 62, 6 (2014), 1416–1438.

Gábor J. Székely, Maria L. Rizzo, and Nail K. Bakirov. 2007. Measuring and testing dependence by correlation of distances. *The Annals of Statistics* 35, 6 (2007), 2769–2794.

G. Gary Wang. 2003. Adaptive response surface method using inherited Latin hypercube design points. *J. Mech. Des.* 125, 2 (2003), 210–220.

Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. 2018. Batched large-scale Bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics*. 745–754.

Zi Wang, Chengtao Li, Stefanie Jegelka, and Pushmeet Kohli. 2017. Batched high-dimensional Bayesian optimization via structural kernel learning. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70. 3656–3664.

Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando De Freitas. 2013. Bayesian optimization in high dimensions via random embeddings. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*. 1778–1784.

Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando de Freitas. 2016. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, Vol. 55. 361–387.

Jing Xie, Peter I. Frazier, and Stephen E. Chick. 2016. Bayesian optimization via simulation with pairwise sampling and correlated prior beliefs. *Operations Research* 64, 2 (2016), 542–559.

Received 9 August 2022; revised 3 January 2024; accepted 4 February 2024