

INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
<http://pubsonline.informs.org>

Rapid Discrete Optimization via Simulation with Gaussian Markov Random Fields

Mark Semelhago, Barry L. Nelson, Eunhye Song, Andreas Wächter

To cite this article:

Mark Semelhago, Barry L. Nelson, Eunhye Song, Andreas Wächter (2020) Rapid Discrete Optimization via Simulation with Gaussian Markov Random Fields. INFORMS Journal on Computing

Published online in Articles in Advance 14 Oct 2020

. <https://doi.org/10.1287/ijoc.2020.0971>

Full terms and conditions of use: <https://pubsonline.informs.org/Publications/Librarians-Portal/PubsOnLine-Terms-and-Conditions>

This article may be used only for the purposes of research, teaching, and/or private study. Commercial use or systematic downloading (by robots or other automatic processes) is prohibited without explicit Publisher approval, unless otherwise noted. For more information, contact permissions@informs.org.

The Publisher does not warrant or guarantee the article's accuracy, completeness, merchantability, fitness for a particular purpose, or non-infringement. Descriptions of, or references to, products or publications, or inclusion of an advertisement in this article, neither constitutes nor implies a guarantee, endorsement, or support of claims made of that product, publication, or service.

Copyright © 2020, INFORMS

Please scroll down for article—it is on subsequent pages






With 12,500 members from nearly 90 countries, INFORMS is the largest international association of operations research (O.R.) and analytics professionals and students. INFORMS provides unique networking and learning opportunities for individual professionals, and organizations of all types and sizes, to better understand and use O.R. and analytics tools and methods to transform strategic visions and achieve better outcomes.

For more information on INFORMS, its publications, membership, or meetings visit <http://www.informs.org>

Rapid Discrete Optimization via Simulation with Gaussian Markov Random Fields

 Mark Semelhago,^a Barry L. Nelson,^a Eunhye Song,^b Andreas Wächter^a
^aDepartment of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois 60208;

^bDepartment of Industrial and Manufacturing Engineering, The Pennsylvania State University, University Park, Pennsylvania 16802

Contact: mark.semelhago@u.northwestern.edu (MS); nelsonb@northwestern.edu,  <https://orcid.org/0000-0002-1325-2624> (BLN); eus358@psu.edu,  <https://orcid.org/0000-0002-5171-0614> (ES); andreas.waechter@northwestern.edu,  <https://orcid.org/0000-0002-3278-5637> (AW)

Received: July 20, 2019

Revised: November 26, 2019; February 6, 2020

Accepted: February 12, 2020

Published Online in Articles in Advance: October 14, 2020

<https://doi.org/10.1287/ijoc.2020.0971>
Copyright: © 2020 INFORMS

Abstract. Inference-based optimization via simulation, which substitutes Gaussian process (GP) learning for the structural properties exploited in mathematical programming, is a powerful paradigm that has been shown to be remarkably effective in problems of modest feasible-region size and decision-variable dimension. The limitation to “modest” problems is a result of the computational overhead and numerical challenges encountered in computing the GP conditional (posterior) distribution on each iteration. In this paper, we substantially expand the size of discrete-decision-variable optimization-via-simulation problems that can be attacked in this way by exploiting a particular GP—discrete Gaussian Markov random fields—and carefully tailored computational methods. The result is the rapid Gaussian Markov Improvement Algorithm (rGMIA), an algorithm that delivers both a global convergence guarantee and finite-sample optimality-gap inference for significantly larger problems. Between infrequent evaluations of the global conditional distribution, rGMIA applies the full power of GP learning to rapidly search smaller sets of promising feasible solutions that need not be spatially close. We carefully document the computational savings via complexity analysis and an extensive empirical study.

Summary of Contribution: The broad topic of the paper is optimization via simulation, which means optimizing some performance measure of a system that may only be estimated by executing a stochastic, discrete-event simulation. Stochastic simulation is a core topic and method of operations research. The focus of this paper is on significantly speeding-up the computations underlying an existing method that is based on Gaussian process learning, where the underlying Gaussian process is a discrete Gaussian Markov Random Field. This speed-up is accomplished by employing smart computational linear algebra, state-of-the-art algorithms, and a careful divide-and-conquer evaluation strategy. Problems of significantly greater size than any other existing algorithm with similar guarantees can solve are solved as illustrations.

History: Accepted by Bruno Tuffin, Area Editor for Simulation.

Funding: This work was supported by the National Science Foundation [Grant DMS-1854562].

Supplemental Material: The online supplement is available at <https://doi.org/10.1287/ijoc.2020.0971>.

Keywords: design of experiments • efficiency • statistical analysis

1. Introduction

Stochastic simulation is a standard tool for designing complex systems that are subject to uncertainty, where a natural goal is to optimize system performance with respect to controllable decision variables. The focus of this paper is minimizing the expected value of a stochastic simulation output of interest, which is often referred to as optimization via simulation (OvS). Within OvS, algorithms have been created that provide various theoretical or practical guarantees. The algorithm we present in this paper has a global convergence guarantee as well as finite-time optimality-gap inference for OvS problems whose decision variables assume integer-ordered values. Such discrete OvS (DOvS) problems appear frequently in operations research when whole units

of a resource (e.g., machines on an assembly line, beds in a hospital, or agents in a call center) need to be allocated.

We are specifically interested in problems whose feasible solutions are defined on a finite subset of the integer lattice, and the number of feasible solutions, combined with the execution time of the simulation, implies that only a small fraction of the feasible solutions can be simulated. Nevertheless, we desire strong finite-time global inference, such as that provided by ranking and selection (R&S)—which simulates all feasible solutions—and a global convergence guarantee in the limit, such as that provided by adaptive random search.

What we refer to as *inference-based optimization* represents the unknown objective function surface as a realization of a random (typically Gaussian)

process, sequentially updates the conditional (posterior) distribution of the objective function as the search progresses, and uses the conditional distribution to guide the search and indicate when it is safe to stop with some statistical guarantee on the optimality gap, which is the difference between the mean of the chosen solution and the optimal solution. This remarkably effective approach is usually credited to Jones et al. (1998); in their setting, the computer simulation was deterministic but so computationally expensive that only a small number of simulation runs could be completed and therefore each one needed to be deployed as productively as possible. Inference-based optimization strategies are a staple of the Bayesian optimization literature.

Inference-based optimization employs a more sophisticated and computationally expensive search step than adaptive random search: updating the conditional distribution. The computational overhead needed to provide this inference has sometimes been ignored because the simulations were so computationally expensive that the time saved by not simulating poor solutions overwhelmed the inference overhead. *In our setting, the output is stochastic, and the number of feasible solutions is huge, but individual replications of a solution may be relatively cheap compared with a deterministic computer experiment. In combination, the computational overhead for inference is no longer negligible compared with the simulation cost.*

An example of the class of problems we consider is condition-based maintenance-policy optimization, as studied in Hoffman et al. (2018): The objective is to minimize the expected cost of operation by assigning a condition number to each machine in a preventative maintenance (PM) queue to avoid more expensive corrective action if it fails. Each machine has a degrading health index of L (perfect health), $L - 1, \dots, 0$ (complete failure). The PM condition is assigned based on the health index, and thus there are $L - 1$ feasible conditions for each machine excluding 0 and L . For a system with d machines in total, the size of the feasible solution space is $(L - 1)^d$, which explodes as the number of machines d increases. A single simulation replication of this problem is relatively cheap (a few seconds) but has large stochastic error variance, which makes it computationally impossible to apply R&S. The computational cost of inference-based optimization also increases with d .

Obviously the effectiveness of inference-based optimization depends critically on how well the chosen Gaussian process (GP) provides insight into the unknown objective function. A GP is defined by its mean function and most critically its covariance function (Santner et al. 2003). Salemi et al. (2019) showed that the continuous-decision-variable covariance functions that are often employed in Bayesian optimization may fail spectacularly when applied to discrete-decision-variable problems, particularly when used for optimality-gap

inference. A discrete Gaussian Markov random field (GMRF), on the other hand, provided excellent search guidance and stopping inference. *Our primary contribution is to greatly extend the reach of GMRF-based optimization by dramatically reducing the computational cost of inference.*

We achieve our speed-up without resorting to any approximations and therefore obtain the full benefits of this powerful inference-based approach. Our rapid Gaussian Markov Improvement Algorithm (rGMIA) combines infrequent evaluations of the full conditional distribution for global inference, with rapid learning on a smaller, adaptive subset of promising solutions. The fact that these small subsets need not be spatially close is key to rGMIA making per-iteration search progress that is nearly the same as would be obtained by computing the full conditional distribution on each iteration.

The remainder of the paper is structured as follows. In Section 2, we review the use of GPs in DOvS algorithms. Section 3 provides the necessary background on GMRFs and complete expected improvement, a functional of the conditional distribution of the GP that guides the search. Section 4 restates GMIA as presented in Salemi et al. (2019). In Section 5, we introduce rGMIA and delve into its computational details in Section 6. In particular, we analyze the computational complexity of rGMIA relative to GMIA and prove its global convergence. Section 7 shows numerical results, evaluating rGMIA against GMIA on carefully selected test problems, and Section 8 contains concluding remarks.

2. Gaussian Processes in DOvS

GPs are stochastic processes with the property that any finite collection of the constituent random variables are jointly normal. GPs are in common use in the design and analysis of computer experiments to model an unknown response surface (Santner et al. 2003). Of interest to us is their use in search algorithms where they play the role of known mathematical properties of the objective function surface. As feasible solutions are evaluated (deterministic computer model) or simulated (stochastic simulation), the conditional distribution of the GP is updated and employed to guide the search for improved solutions. Choosing the covariance function of a GP is important as it implies certain properties of the objective function surface it models, and this has consequences both on the validity of the statistical learning and on the computations. Calculating the conditional distribution usually requires inverting a large, dense, and sometimes ill-conditioned covariance matrix, and this is the essential bottleneck for applying GP optimization to large-scale problems.

The use of GPs in OvS problems, with both continuous and discrete decision variables, often results in algorithms that choose a solution to simulate \mathbf{x}_t at iteration t where the selection criterion is prescribed

by the acquisition function $a(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$. We use $(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ to represent the posterior mean and variance, respectively, of the GP $\mathbb{Y}(\cdot)$ that represents the unknown surface $y(\cdot)$ at iteration t . This notation will be defined more precisely later. In the following, we review GP methods devised for solving DOvS problems.

Frazier et al. (2009) consider a Bayesian R&S problem with independent normal responses and use a GP model with correlation among alternatives as a prior on the mean values of the response. They then treat the problem of finding the alternative with the smallest mean as a dynamic programming problem to optimally allocate computer effort. Because this problem is intractable, they myopically approximate an optimal allocation by simulating the alternative that maximizes the benefit received as if each iteration were the last iteration of the dynamic program. They term this acquisition function the knowledge gradient (KG). Xie et al. (2016) address the same setting where a multivariate normal prior is used to represent the means of a finite number of alternatives. They extend the acquisition function found in Frazier et al. (2009) by considering pairwise sampling using common random numbers (CRN). Our GMRF-based approach can be considered a form of Bayesian R&S where there is a prior distribution exhibiting strong correlation among solutions, as in Xie et al. (2016). Therefore, not all solutions need to be simulated to make optimality-gap inference.

Employing a very different approach, Sun et al. (2014) model the simulation output at a solution, \mathbf{x} , as $G(\mathbf{x}) = M(\mathbf{x}) + \epsilon(\mathbf{x})$, where $M(\mathbf{x})$ is a stationary, mean-zero GP and $\epsilon(\mathbf{x})$ is an error term that models the stochastic noise in the simulation output. The “stochastic kriging” model, G , is updated as the algorithm proceeds and used to construct a distribution from which the next solution to simulate will be sampled. The use of a sampling distribution as the acquisition function to guide the search distinguishes this method from the others discussed above. None of the prior work cited above considers problems on the scale that we address here in terms of the number of feasible solutions in a discrete space.

3. Optimization Using GMRFs

Consider the global DOvS problem: $\min_{\mathbf{x} \in \mathcal{X}} y(\mathbf{x}) = \mathbb{E}[Y(\mathbf{x})]$, where the feasible region \mathcal{X} is a finite subset of the d -dimensional integer lattice \mathbb{Z}^d ; let $n = |\mathcal{X}|$ be the number of feasible solutions. In particular, we assume \mathcal{X} is a d -dimensional hyperrectangle. At each feasible solution \mathbf{x} , the objective function $y(\mathbf{x})$ is the unknown mean of the simulation output, $Y(\mathbf{x})$, which can be estimated via simulation. For any feasible solution \mathbf{x} , we observe the output $Y_j(\mathbf{x}) = y(\mathbf{x}) + \epsilon_j(\mathbf{x})$ on replication $j = 1, 2, \dots$, where $\{\epsilon_j(\mathbf{x})\}$ are assumed i.i.d. normal with mean 0 and finite (unknown) variance $\sigma^2(\mathbf{x})$ that may depend on \mathbf{x} . In this section, we present the

underlying stochastic process for our inference-based optimization procedure to solve the DOvS problem.

3.1. Gaussian Markov Random Fields

A GP-based optimization method for a finite feasible-solution space starts by modeling the unknown objective function values $\mathbf{y} = [y(\mathbf{x}_1), y(\mathbf{x}_2), \dots, y(\mathbf{x}_n)]^\top$ as a multivariate normal random vector $\mathbb{Y} = [\mathbb{Y}_1, \mathbb{Y}_2, \dots, \mathbb{Y}_n]^\top$ with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. A GMRF, a special case of GP, is a nondegenerate $n \times 1$ Gaussian random vector \mathbb{Y} that is associated with an undirected and labeled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where \mathcal{V} denotes the set of nodes and \mathcal{E} denotes the set of edges; see Rue and Held (2005). Each node in \mathcal{V} is associated with a unique element of \mathbb{Y} . Two nodes in the graph are called neighbors if they are connected by an edge. As described below, the graph \mathcal{G} determines the structure of the precision matrix, \mathbf{Q} , which is the inverse of the covariance matrix $\boldsymbol{\Sigma}$ of \mathbb{Y} .

In general, the diagonal entries Q_{ii} of a precision matrix are such that $\text{Var}(\mathbb{Y}_i | \mathbb{Y}_{\mathcal{V} \setminus \{i\}}) = 1/Q_{ii}$, where $\mathbb{Y}_{\mathcal{V} \setminus \{i\}}$ is the vector of values of the GMRF observed at the nodes in $\mathcal{V} \setminus \{i\}$. Thus, they are the reciprocals of the conditional variances. The off-diagonal elements are proportional to the conditional correlations; specifically, $\text{Corr}(\mathbb{Y}_i, \mathbb{Y}_j | \mathbb{Y}_{\mathcal{V} \setminus \{i,j\}}) = -Q_{ij} / \sqrt{Q_{ii}Q_{jj}}$, where $\mathbb{Y}_{\mathcal{V} \setminus \{i,j\}}$ is the vector of values of the GMRF observed only at the nodes in $\mathcal{V} \setminus \{i,j\}$.

The graph \mathcal{G} determines the nonzero pattern of the precision matrix \mathbf{Q} , and vice versa, because for a GMRF $Q_{ij} \neq 0$ if and only if $\{i, j\} \in \mathcal{E}$. Thus, the precision matrix is sparse if the set of edges is small. GMRFs are “Markov” because they possess the local Markov property: $\mathbb{Y}_i \perp \mathbb{Y}_{\mathcal{V} \setminus \{i, \mathcal{N}(i)\}} | \mathbb{Y}_{\mathcal{N}(i)}$ for every $i \in \mathcal{V}$, where $\mathcal{N}(i) = \{j: \{i, j\} \in \mathcal{E}\}$. This local Markovian property encapsulates the prior belief that if all of the neighbors of a feasible solution have been observed, then there is little additional information about that solution remaining in non-neighboring solutions; this regularity is often appropriate for DOvS problems that tend to feature locally well-behaved objective functions. By contrast, the Gaussian covariance function favored in Bayesian optimization implies an objective function that is infinitely continuously differentiable, a much stronger condition.

3.2. Optimization

In a DOvS problem with integer-ordered decision variables, the natural graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ defines the nodes \mathcal{V} to be \mathcal{X} . Construction of \mathcal{E} requires a neighborhood. Salemi et al. (2019) show that a particularly effective choice is based on the ℓ_2 distance, $\mathcal{N}(\mathbf{x}) = \{\mathbf{x}' \in \mathcal{X}: \|\mathbf{x} - \mathbf{x}'\|_2 = 1\}$, which implies that the fraction of nonzero entries in the precision matrix \mathbf{Q} is bounded above by $(2d + 1)/n$ for hyperrectangular \mathcal{X} , which makes \mathbf{Q} very sparse for large n . This allows

faster computations than when a dense precision matrix is used.

We parameterize the entries of \mathbf{Q} by $\boldsymbol{\theta} = [\theta_0, \theta_1, \dots, \theta_d]^\top$. For the neighborhood $\mathcal{N}(\mathbf{x})$, we let $Q_{ij} = \theta_0$, if $\mathbf{x}_i = \mathbf{x}_j$, and $Q_{ij} = -\theta_0\theta_j$, if $|\mathbf{x}_i - \mathbf{x}_j| = \mathbf{e}_j$, where $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, \mathbf{e}_j is the j th standard basis vector and $|\cdot|$ is the component-wise absolute value. In all other cases, $Q_{ij} = 0$. Thus, θ_0 is the conditional precision of each solution, and θ_j is the conditional correlation between solutions that differ by 1 in the j th coordinate direction, given their neighbors. Under this parametrization $\mathbf{Q} = \mathbf{Q}(\boldsymbol{\theta})$, but we omit $\boldsymbol{\theta}$ for notational simplicity. Solutions on the boundaries of the feasible region, or without neighbors in all coordinate directions, would require adjusted parameters for the GMRF to be stationary. We have chosen to ignore this, as the impact seems negligible and, therefore, treat our GMRF as nonstationary.

Because the conditional precisions must be positive, it follows that $\theta_0 > 0$. We also want neighbors to have nonnegative conditional correlations, so $\theta_1, \theta_2, \dots, \theta_d$ are chosen to be nonnegative. Additionally, \mathbf{Q} should be positive definite. With these conditions, \mathbf{Q} is a nonsingular M -matrix so its inverse is nonnegative (Johnson 1982). In other words, there are no negative (unconditional) correlations among nodes in the GMRF, a property that makes sense in many DOvS problems as the objective-function values of neighboring solutions should be similar to one another. Notice that even though we construct \mathbf{Q} to be sparse, its covariance matrix, $\Sigma = \mathbf{Q}^{-1}$, is typically dense, as it should be.

Based on our GMRF model, the prior joint distribution of \mathbb{Y} is $N(\boldsymbol{\mu}, \mathbf{Q}^{-1})$. We adopt noninformative constant prior mean $\boldsymbol{\mu} = \mu \mathbf{1}_{n \times 1}$, where $\mathbf{1}_{n \times 1}$ is an $n \times 1$ vector of 1s. In total, we have $d + 2$ parameters to specify a GMRF for a d -dimensional decision variable \mathbf{x} .

Suppose that we simulate a subset of solutions in \mathcal{X} . Let $\tilde{\mathbf{Y}}$ be an $n \times 1$ vector such that each element is either the sample mean of the associated feasible solution, if it has been simulated, or μ if it has not. Consistent with the output model, we represent $\tilde{\mathbf{Y}}$ as a realization of the GMRF $\mathbb{Y}^\epsilon = \mathbb{Y} + \boldsymbol{\epsilon}$, where the entries of $\boldsymbol{\epsilon}$ are jointly normally distributed, if the corresponding solutions have been simulated, and 0s, otherwise. The composite prior distribution of \mathbb{Y}^ϵ is $N(\boldsymbol{\mu}, (\mathbf{Q} + \mathbf{Q}_\epsilon)^{-1})$. We choose to simulate all solutions independently (no CRN), which makes \mathbf{Q}_ϵ a diagonal matrix so that the sparsity pattern of \mathbf{Q} is preserved for $\mathbf{Q} + \mathbf{Q}_\epsilon$. If solution \mathbf{x} has been simulated, the corresponding diagonal element of \mathbf{Q}_ϵ is estimated by $r(\mathbf{x})/S^2(\mathbf{x})$, where $r(\mathbf{x})$ is the number of replications that have been obtained and $S^2(\mathbf{x})$ is the sample variance estimate of $\sigma^2(\mathbf{x})$; otherwise, the corresponding element in \mathbf{Q}_ϵ is set to 0.

Salemi et al. (2019) prove that the conditional distribution of $\mathbb{Y}|\mathbb{Y}^\epsilon = \tilde{\mathbf{Y}}$ is

$$N\left(\boldsymbol{\mu} + \bar{\mathbf{Q}}^{-1}\mathbf{Q}_\epsilon(\tilde{\mathbf{Y}} - \boldsymbol{\mu}), \bar{\mathbf{Q}}^{-1}\right), \quad (1)$$

where $\bar{\mathbf{Q}} = \mathbf{Q} + \mathbf{Q}_\epsilon$ is the conditional precision matrix. Notice that computing the conditional mean and variance requires $\bar{\mathbf{Q}}^{-1}$, and $\bar{\mathbf{Q}}$ changes as we simulate additional feasible solutions. *Efficiently calculating quantities that depend on (1) for a large number of feasible solutions is the principal topic of this paper.* In practice, parameters such as $\boldsymbol{\theta}$ and $\boldsymbol{\mu}$ are unknown but are estimated via maximum likelihood after simulating an initial set of feasible solutions. The intrinsic precision matrix, \mathbf{Q}_ϵ , on the other hand, is often directly estimated from simulation output by using the sample variances at simulated solutions, as described above.

Both the GMIA algorithm of Salemi et al. (2019) and our rGMIA guide their search and (possibly) termination using *complete expected improvement* (CEI), which is defined in Salemi et al. (2019). At any iteration, the estimated optimal solution is $\tilde{\mathbf{x}} = \arg \min_{\{\mathbf{x} \in \mathcal{X}: r(\mathbf{x}) > 0\}} \tilde{\mathbf{Y}}(\mathbf{x})$, where $\tilde{\mathbf{Y}}(\mathbf{x})$ is the component of $\tilde{\mathbf{Y}}$ associated with solution \mathbf{x} . The CEI of each candidate solution, $\mathbf{x} \in \mathcal{X} \setminus \tilde{\mathbf{x}}$, is the expected improvement in the objective function offered by solution \mathbf{x} compared with $\tilde{\mathbf{x}}$, where the expectation is with respect to the current conditional distribution of the GMRF. Thus, the CEI of a candidate solution \mathbf{x} relative to $\tilde{\mathbf{x}}$ is $\text{CEI}(\tilde{\mathbf{x}}, \mathbf{x}) = \mathbb{E}[\max(\mathbb{Y}(\tilde{\mathbf{x}}) - \mathbb{Y}(\mathbf{x}), 0) | \mathbb{Y}^\epsilon = \tilde{\mathbf{Y}}]$, where the expectation is conditional on $\mathbb{Y}^\epsilon = \tilde{\mathbf{Y}}$, the simulation output that has been collected. CEI is an extension of the EI acquisition function (Jones et al. 1998) tailored for stochastic simulation (Salemi et al. 2019). The joint conditional distribution of $\mathbb{Y}(\tilde{\mathbf{x}})$ and $\mathbb{Y}(\mathbf{x})$, $\tilde{\mathbf{x}} \neq \mathbf{x}$ is bivariate normal with parameters taken from the mean and the covariance matrix of (1) corresponding to $\tilde{\mathbf{x}}$ and \mathbf{x} . We denote the conditional mean and conditional variance at \mathbf{x} as $M(\mathbf{x})$ and $V(\mathbf{x})$, respectively, and the conditional covariance between $\tilde{\mathbf{x}}$ and \mathbf{x} as $C(\tilde{\mathbf{x}}, \mathbf{x})$. For a given solution, \mathbf{x} , the variance of the difference of $\mathbb{Y}(\tilde{\mathbf{x}}) - \mathbb{Y}(\mathbf{x})$ is $V(\tilde{\mathbf{x}}, \mathbf{x}) \equiv V(\tilde{\mathbf{x}}) + V(\mathbf{x}) - 2C(\tilde{\mathbf{x}}, \mathbf{x})$.

Salemi et al. (2019) show that the CEI of \mathbf{x} can be expressed as

$$\begin{aligned} \text{CEI}(\tilde{\mathbf{x}}, \mathbf{x}) &= (M(\tilde{\mathbf{x}}) - M(\mathbf{x}))\Phi\left(\frac{M(\tilde{\mathbf{x}}) - M(\mathbf{x})}{\sqrt{V(\tilde{\mathbf{x}}, \mathbf{x})}}\right) \\ &\quad + \sqrt{V(\tilde{\mathbf{x}}, \mathbf{x})}\phi\left(\frac{M(\tilde{\mathbf{x}}) - M(\mathbf{x})}{\sqrt{V(\tilde{\mathbf{x}}, \mathbf{x})}}\right), \end{aligned} \quad (2)$$

where ϕ and Φ are the density and cumulative distribution functions, respectively, of a standard normal random variable. Both GMIA and rGMIA use CEI for search guidance (simulate next the solution with the largest CEI) and as a stopping criterion (stop when $\max_{\mathbf{x} \in \mathcal{X} \setminus \tilde{\mathbf{x}}} \text{CEI}(\tilde{\mathbf{x}}, \mathbf{x}) \leq \delta$, where δ is user-specified

acceptable optimality gap). CEI has been shown to have desirable properties. For instance, Chen and Ryzhov (2019) prove that under simplified conditions (R&S with independent and normally distributed simulation output with known variances), CEI satisfies the conditions found in Glynn and Juneja (2004) that ensure that the probability of incorrect selection converges to zero at the fastest possible exponential rate as the total simulation budget increases to infinity. Such asymptotic properties, along with the impressive empirical performance shown in Salemi et al. (2019), argue that CEI is a good acquisition function for inference-based optimization.

Let $\mathbf{M}(\mathbf{x}_{\mathcal{X}}) = [M(\mathbf{x}_1), M(\mathbf{x}_2), \dots, M(\mathbf{x}_n)]^T$, $\mathbf{V}(\mathbf{x}_{\mathcal{X}}) = [V(\mathbf{x}_1), V(\mathbf{x}_2), \dots, V(\mathbf{x}_n)]^T$, and $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{X}}) = [C(\tilde{\mathbf{x}}, \mathbf{x}_1), C(\tilde{\mathbf{x}}, \mathbf{x}_2), \dots, C(\tilde{\mathbf{x}}, \mathbf{x}_n)]^T$. From a computational point of view, to obtain $V(\tilde{\mathbf{x}}, \mathbf{x})$, $\forall \mathbf{x} \in \mathcal{X}$, we need to compute the diagonal of $\bar{\mathbf{Q}}^{-1}$ to obtain $\mathbf{V}(\mathbf{x}_{\mathcal{X}})$ and the column of $\bar{\mathbf{Q}}^{-1}$ corresponding to $\tilde{\mathbf{x}}$ for $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{X}})$. The latter operation requires solving the linear system $\bar{\mathbf{Q}}\mathbf{z} = \mathbf{e}_{\tilde{\mathbf{x}}}$ for \mathbf{z} , where $\mathbf{e}_{\tilde{\mathbf{x}}}$ is an n -dimensional basis vector consisting of zeroes, except for a 1 in the position corresponding to $\tilde{\mathbf{x}}$. The former is more expensive to compute; a naive approach is to compute the full inverse $\bar{\mathbf{Q}}^{-1}$ and extract its diagonal. Both operations require factorizing $\bar{\mathbf{Q}}$ at every iteration. Although sparsity of $\bar{\mathbf{Q}}$ helps, it is increasingly expensive for large n . Such computational challenges serve as our motivation to substantially extend GMIA's reach to larger numbers of feasible solutions in higher dimensions.

Salemi et al. (2019) introduced a multiresolution framework in which the feasible solution space is divided into nonoverlapping regions. Each region is represented by a solution-level GMRF, and the average objective function values of the regions are represented by a region-level GMRF. Their approach provides global and local search guidance as well as stopping inference while reducing the size of the solution-level GMRFs. Of course, any such multiresolution approach will eventually be limited by the largest solution-level GMRF it can handle. Thus, we concentrate on extending the solution-level algorithm in this paper.

Semelhago et al. (2017) propose an efficient way to compute the diagonal elements of $\bar{\mathbf{Q}}^{-1}$ without full inversion when $\bar{\mathbf{Q}}$ is sparse. PARDISO (Schenk and Gärtner 2018), a linear solver specialized for parallel computation using state-of-the-art algorithms, was employed to perform this calculation. However, the Semelhago et al. (2017) algorithm still requires factorizing $\bar{\mathbf{Q}}$ on every iteration. *Our approach not only avoids fully updating $\bar{\mathbf{Q}}^{-1}$, but also factorizing $\bar{\mathbf{Q}}$ on every iteration, and it employs exact, rapidly computed CEIs on all iterations.*

4. Gaussian Markov Improvement Algorithm

In this section, we provide a quick review of GMIA. As presented in Algorithm 1, GMIA begins by simulating a

small number, n_0 , of well-placed initial design points (feasible solutions) and uses the outputs to compute the maximum-likelihood estimators (MLEs) of $\boldsymbol{\mu}$ and $\boldsymbol{\theta}$. Then, it updates the conditional distribution in (1) given the simulation outputs from the initial design and computes the CEIs of all solutions in \mathcal{X} . Although the stopping criterion is not satisfied, GMIA simulates the current sample-best solution, $\tilde{\mathbf{x}}$, and the solution with the largest CEI, \mathbf{x}^{CEI} , at each iteration. If $Y(\mathbf{x})$ is discrete-valued, then $\arg \min_{\{\mathbf{x} \in \mathcal{X}: r(\mathbf{x}) > 0\}} \bar{Y}(\mathbf{x})$ and $\arg \max_{\mathbf{x} \in \mathcal{X} \setminus \tilde{\mathbf{x}}} \text{CEI}(\tilde{\mathbf{x}}, \mathbf{x})$ may be sets of size greater than 1. When this occurs, we randomly select a single solution in the set to be $\tilde{\mathbf{x}}$ and \mathbf{x}^{CEI} , respectively.

There are two stopping paradigms in OvS: fixed-precision and fixed-budget (Hunter and Nelson 2017). For the former, the algorithm terminates when the inferred optimality gap of the current best solution falls below a user-defined δ . Using CEI to terminate, as discussed in Section 3.2, is an example of a fixed-precision approach. In this paradigm, the performance of an algorithm is evaluated by whether it actually achieves the inferred optimality gap at termination, as well as the computational effort required to terminate. On the other hand, for a fixed-budget paradigm an algorithm terminates when a predefined computational budget is expended and the performance of the algorithm is evaluated by how small the achieved optimality gap is at termination. Typically for an R&S procedure, the computational budget is specified as the allowable number of simulation replications, because other computational overhead is negligible when the number of feasible solutions is small. For large-scale, inferential optimization, however, the budget should encompass both simulation time and nonsimulation time.

Algorithm 1 (GMIA)

- 1 Choose $n_0 \ll n$ initial design points. Simulate r replications for each design point and use the simulation output to compute MLEs for the GMRF parameters $(\boldsymbol{\mu}, \boldsymbol{\theta})$. Construct $\bar{\mathbf{Q}} = \mathbf{Q} + \mathbf{Q}_\epsilon$ and $\bar{\mathbf{Y}}$;
- 2 **while** Stopping criterion not reached **do**
- 3 Find $\tilde{\mathbf{x}} = \arg \min_{\{\mathbf{x} \in \mathcal{X}: r(\mathbf{x}) > 0\}} \bar{Y}(\mathbf{x})$;
- 4 Compute Cholesky factor of $\bar{\mathbf{Q}}$: $\mathbf{L}_{\bar{\mathbf{Q}}}$;
- 5 Compute $\mathbf{V}(\mathbf{x}_{\mathcal{X}}) = \text{diag}(\bar{\mathbf{Q}}^{-1})$, using $\mathbf{L}_{\bar{\mathbf{Q}}}$;
- 6 Compute $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{X}}) = \bar{\mathbf{Q}}^{-1} \mathbf{e}_{\tilde{\mathbf{x}}}$, using $\mathbf{L}_{\bar{\mathbf{Q}}}$;
- 7 Compute $\mathbf{M}(\mathbf{x}_{\mathcal{X}}) = \boldsymbol{\mu} + \bar{\mathbf{Q}}^{-1} \mathbf{Q}_\epsilon (\bar{\mathbf{Y}} - \boldsymbol{\mu})$, using $\mathbf{L}_{\bar{\mathbf{Q}}}$;
- 8 Calculate $\text{CEI}(\tilde{\mathbf{x}}, \mathbf{x})$, $\forall \mathbf{x} \in \mathcal{X}$;
- 9 Find $\mathbf{x}^{\text{CEI}} = \arg \max_{\mathbf{x} \in \mathcal{X} \setminus \tilde{\mathbf{x}}} \text{CEI}(\tilde{\mathbf{x}}, \mathbf{x})$;
- 10 Simulate at $\tilde{\mathbf{x}}$ and \mathbf{x}^{CEI} . Update $\bar{\mathbf{Y}}$, \mathbf{Q}_ϵ , and $\bar{\mathbf{Q}}$ by incorporating the new simulation outputs;
- 11 **end**
- 12 Return $\tilde{\mathbf{x}} = \arg \min_{\{\mathbf{x} \in \mathcal{X}: r(\mathbf{x}) > 0\}} \bar{Y}(\mathbf{x})$ as the estimated optimal solution;

In Algorithm 1, Steps 4 and 5 are the most expensive in terms of nonsimulation overhead. As mentioned in

the previous section, Semelhago et al. (2017) propose extracting the diagonal elements of $\bar{\mathbf{Q}}^{-1}$ without computing the inverse entirely. Although this approach greatly reduces the cost of Step 5, Step 4 remains a bottleneck. Because of the sparsity of $\bar{\mathbf{Q}}$, the cost of the Cholesky factorization is much cheaper than it is for a dense matrix. Nonetheless, it still becomes costly when the problem size is large, limiting the scope of GMIA. In GP-based optimization algorithms, a common trick is to update the conditional distribution efficiently using the Sherman–Morrison–Woodbury (SMW) formula to avoid factorizing $\bar{\mathbf{Q}}$ every iteration. In Appendix C of the online supplement, we show that this approach results in greater computational burden than our rGMIA.

5. Overview of rGMIA

Computing the CEIs for *all* feasible solutions enables GMIA to exploit global optimality-gap inference, but it comes at a computational cost. Moreover, when \mathcal{X} is large, most solutions' CEIs are largely unaffected by the new simulation outputs at $\tilde{\mathbf{x}}$ and \mathbf{x}^{CEI} . If we knew that a much smaller subset of solutions would contain those with the largest CEIs over the next, say, $p - 1$ iterations, then we could update the CEIs for only those solutions in the subset. Of course, we do not know such a subset, but this insight motivates restricting CEI computation to a small subset of *promising* solutions for several iterations. Because we only require the diagonal elements of $\bar{\mathbf{Q}}^{-1}$ corresponding to those solutions in the subset, this strategy will greatly reduce the computational overhead in Step 5 of Algorithm 1. Furthermore, as shown in the following sections, this scheme avoids an expensive factorization in Step 4 by replacing it with much cheaper, lower-dimensional linear algebra. *Accomplishing this in a way that significantly reduces computation without hampering search progress is our key contribution.*

Algorithm 2 illustrates the steps of rGMIA including the necessary computation required at each step. We defer discussion of the derivation of these results to Section 6 and provide a high-level description here. There are three stages to rGMIA: initialization, rapid search, and global search. In the initialization stage, rGMIA estimates the GMRF parameters and updates its conditional distribution. Then, it proceeds to Step 27 of global search.

rGMIA alternates between many *rapid-search* iterations and a single *global-search* iteration, as long as the global-search termination criterion is not met. For a fixed-budget setting, this would be the constraint on the algorithm run-time. For a fixed-precision setting, the CEI stopping criterion, $\max_{\mathbf{x} \in \mathcal{X} \setminus \tilde{\mathbf{x}}} \text{CEI}(\tilde{\mathbf{x}}, \mathbf{x}) \leq \delta$, is used. At each global-search iteration (Steps 14–31), rGMIA partitions the feasible region into a *search set*

$\mathcal{S} \subset \mathcal{X}$ and a *fixed set* $\mathcal{F} \equiv \mathcal{X} \setminus \mathcal{S}$. The former contains the best simulated solution, $\tilde{\mathbf{x}} = \arg \min_{\{\mathbf{x} \in \mathcal{X}: r(\mathbf{x}) > 0\}} \tilde{Y}(\mathbf{x})$, and promising candidate solutions that need not be spatially close. The intermediate matrices, \mathbf{A} and \mathbf{B} , and vector, \mathbf{a} , required for fast linear algebra during the rapid-search iterations are also computed. Then, rGMIA proceeds to rapid search (Steps 7–12), checking the rapid-search termination criterion along the way, which allows the algorithm to escape from simulating the solutions in \mathcal{S} and return to a global-search iteration when the benefit from additional rapid search is marginal. We discuss candidates for the rapid-search termination criterion in Section 6.1. During rapid-search iterations, rGMIA computes the CEIs of solutions in \mathcal{S} exactly and selects the next solution to simulate within \mathcal{S} . In the following global-search iteration, \mathcal{S} and \mathcal{F} are updated reflecting cumulative simulation results.

We let $\mathbf{M}(\mathbf{x}_{\mathcal{S}})$, $\mathbf{V}(\mathbf{x}_{\mathcal{S}})$, and $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{S}})$ represent the vectors of conditional means, conditional variances, and conditional covariances with respect to $\tilde{\mathbf{x}}$, respectively, of solutions in \mathcal{S} ; $\mathbf{M}(\mathbf{x}_{\mathcal{F}})$, $\mathbf{V}(\mathbf{x}_{\mathcal{F}})$, and $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{F}})$ are defined similarly for \mathcal{F} . During rapid-search iterations, we choose $\tilde{\mathbf{x}}$ to be the best simulated solution within \mathcal{S} —i.e., $\tilde{\mathbf{x}} = \arg \min_{\{\mathbf{x} \in \mathcal{S}: r(\mathbf{x}) > 0\}} \tilde{Y}(\mathbf{x})$. This ensures that we only need to update the conditional distribution of solutions in \mathcal{S} during the rapid-search iterations. Because CEI is relative to the current sample-best solution, if we allowed $\tilde{\mathbf{x}}$ to be in \mathcal{F} , then we would need a full conditional-distribution update to compute the exact CEIs. We do a full update only on a global-search iteration.

Computational savings per iteration for rGMIA come largely from $|\mathcal{S}| \ll |\mathcal{F}| \approx |\mathcal{X}|$. That is, the relatively small cardinality of \mathcal{S} is the key factor. However, effective search, which is per-iteration progress toward the optimal solution, depends on the content of \mathcal{S} . Our proposal is to select solutions with the largest CEIs with respect to $\tilde{\mathbf{x}}$ at each global-search iteration. This is based on the premise that the CEIs of solutions change incrementally in subsequent iterations unless they are very close to a solution chosen for simulation. Other choices are possible. There is no computational advantage for the solutions in \mathcal{S} to be close to each other in \mathcal{X} , which allows the rapid search to remain global even though only considering a subset of solutions. We have observed that the resulting \mathcal{S} includes solutions near $\tilde{\mathbf{x}}$, other solutions with favorable sample means, as well as solutions in unexplored regions of \mathcal{X} . However, savings in the form of per-iteration computational overhead do not depend on this choice of \mathcal{S} .

The idea of restricting inference to a smaller subset to reduce computational cost appears in other work as well. For instance, for their GP-based search,

Xie et al. (2016) propose forming a smaller set of candidate solutions in some randomized fashion or applying a local gradient search on the KG surface by relaxing the integrality condition. Unlike our approach, these subsets or local search perimeters are altered and the GP conditional distribution is updated for a different set of solutions at every iteration. By contrast, concentrating on the same \mathcal{S} for several rapid-search iterations allows rGMIA to exploit the savings from cheap computational linear algebra to a greater extent.

6. Properties of rGMIA

In this section, we provide computational complexity analysis of rGMIA. We analyze the computational costs of rapid search and global search in Sections 6.1 and 6.2, respectively. Section 6.3 then compares rGMIA to GMIA and proves global convergence.

Partitioning $\bar{\mathbf{Q}}$ into block matrices corresponding to \mathcal{F} and \mathcal{S} as

$$\bar{\mathbf{Q}} = \begin{bmatrix} \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}} & \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}} \\ \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}^\top & \bar{\mathbf{Q}}_{\mathcal{S}\mathcal{S}} \end{bmatrix},$$

Algorithm 2 (rGMIA)

<ol style="list-style-type: none"> 1 Choose $n_0 \ll n$ initial solutions. Simulate at each solution and compute MLEs for the GMRF parameters $(\boldsymbol{\mu}, \boldsymbol{\theta})$. Construct $\bar{\mathbf{Q}} = \mathbf{Q} + \mathbf{Q}_\epsilon$; 2 Find $\tilde{\mathbf{x}} = \arg \min_{\{\mathbf{x} \in \mathcal{X}: r(\mathbf{x}) > 0\}} \bar{Y}(\mathbf{x})$; 3 Compute Cholesky factor of $\bar{\mathbf{Q}}$: $\mathbf{L}_{\bar{\mathbf{Q}}}$; 4 Compute $\mathbf{V}(\mathbf{x}_{\mathcal{Q}}) = \text{diag}(\bar{\mathbf{Q}}^{-1})$, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{Q}}) = \bar{\mathbf{Q}}^{-1} \mathbf{e}_{\tilde{\mathbf{x}}}$, $\mathbf{M}(\mathbf{x}_{\mathcal{Q}}) = \boldsymbol{\mu} + \bar{\mathbf{Q}}^{-1} \mathbf{Q}_\epsilon (\bar{Y} - \boldsymbol{\mu})$, using $\mathbf{L}_{\bar{\mathbf{Q}}}$. Go to Step 27; 5 while <i>global-search termination criterion not reached</i> do 6 while <i>rapid-search termination criterion not reached</i> do 7 Simulate at $\tilde{\mathbf{x}}$, \mathbf{x}^{CEI}. Update simulation information by updating $\bar{Y}(\tilde{\mathbf{x}})$, $\bar{Y}(\mathbf{x}^{\text{CEI}})$, \mathbf{Q}_ϵ, $\bar{\mathbf{Q}}$, $\bar{\mathbf{Q}}_{\mathcal{S}\mathcal{S}}$; 8 Find $\tilde{\mathbf{x}} = \arg \min_{\{\mathbf{x} \in \mathcal{X}: r(\mathbf{x}) > 0\}} \bar{Y}(\mathbf{x})$; 9 Compute $\mathbf{V}(\mathbf{x}_{\mathcal{S}})$, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{S}})$ by computing $\Sigma_{\mathcal{S}\mathcal{S}} = (\bar{\mathbf{Q}}_{\mathcal{S}\mathcal{S}} - \mathbf{B})^{-1}$; 10 Compute $\mathbf{M}(\mathbf{x}_{\mathcal{S}}) = \boldsymbol{\mu}_{\mathcal{S}} + \Sigma_{\mathcal{S}\mathcal{S}} ([\mathbf{Q}_\epsilon]_{\mathcal{S}\mathcal{S}} (\bar{Y}(\mathbf{x}_{\mathcal{S}}) - \boldsymbol{\mu}_{\mathcal{S}}) - \mathbf{a})$; 11 Calculate CEI($\tilde{\mathbf{x}}, \mathbf{x}$), $\forall \mathbf{x} \in \mathcal{S}$; 12 Find $\mathbf{x}^{\text{CEI}} = \arg \max_{\mathbf{x} \in \mathcal{S} \setminus \tilde{\mathbf{x}}} \text{CEI}(\tilde{\mathbf{x}}, \mathbf{x})$; 13 end 14 Simulate at $\tilde{\mathbf{x}}$, \mathbf{x}^{CEI}. Update simulation information by updating $\bar{Y}(\tilde{\mathbf{x}})$, $\bar{Y}(\mathbf{x}^{\text{CEI}})$, \mathbf{Q}_ϵ, $\bar{\mathbf{Q}}$, $\bar{\mathbf{Q}}_{\mathcal{S}\mathcal{S}}$; 15 Find $\tilde{\mathbf{x}} = \arg \min_{\{\mathbf{x} \in \mathcal{X}: r(\mathbf{x}) > 0\}} \bar{Y}(\mathbf{x})$; 16 Compute $\mathbf{V}(\mathbf{x}_{\mathcal{S}})$ from $\Sigma_{\mathcal{S}\mathcal{S}} = (\bar{\mathbf{Q}}_{\mathcal{S}\mathcal{S}} - \mathbf{B})^{-1}$; 17 Compute $\mathbf{M}(\mathbf{x}_{\mathcal{S}}) = \boldsymbol{\mu}_{\mathcal{S}} + \Sigma_{\mathcal{S}\mathcal{S}} ([\mathbf{Q}_\epsilon]_{\mathcal{S}\mathcal{S}} (\bar{Y}(\mathbf{x}_{\mathcal{S}}) - \boldsymbol{\mu}_{\mathcal{S}}) - \mathbf{a})$; 18 Compute $\mathbf{V}(\mathbf{x}_{\mathcal{F}}) = \text{diag}(\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1}) + \text{diag}(\mathbf{A} \Sigma_{\mathcal{S}\mathcal{S}} \mathbf{A}^\top)$, using $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}}$; 19 Compute $\mathbf{M}(\mathbf{x}_{\mathcal{F}}) = \boldsymbol{\mu}_{\mathcal{F}} + \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} [\mathbf{Q}_\epsilon]_{\mathcal{F}\mathcal{F}} (\bar{Y}(\mathbf{x}_{\mathcal{F}}) - \boldsymbol{\mu}_{\mathcal{F}}) - \mathbf{A}(\mathbf{M}(\mathbf{x}_{\mathcal{S}}) - \boldsymbol{\mu}_{\mathcal{S}})$, using $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}}$; 20 if $\tilde{\mathbf{x}} \in \mathcal{S}$ then 21 Compute $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{S}}) = [\Sigma_{\mathcal{S}\mathcal{S}}]_{\tilde{\mathbf{x}}}$; 22 Compute $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{F}}) = -\mathbf{A} [\Sigma_{\mathcal{S}\mathcal{S}}]_{\tilde{\mathbf{x}}}$; 23 else 24 Compute $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{S}}) = -\Sigma_{\mathcal{S}\mathcal{S}} [\mathbf{A}^\top]_{\tilde{\mathbf{x}}}$; 25 Compute $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{F}}) = \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} \mathbf{e}_{\tilde{\mathbf{x}}} + \mathbf{A} \Sigma_{\mathcal{S}\mathcal{S}} [\mathbf{A}^\top]_{\tilde{\mathbf{x}}}$, using $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}}$; 26 end 27 Calculate CEI($\tilde{\mathbf{x}}, \mathbf{x}$), $\forall \mathbf{x} \in \mathcal{X}$; 28 Find $\mathbf{x}^{\text{CEI}} = \arg \max_{\mathbf{x} \in \mathcal{X} \setminus \tilde{\mathbf{x}}} \text{CEI}(\tilde{\mathbf{x}}, \mathbf{x})$; 29 Construct $\{\mathcal{F}, \mathcal{S}\}$ partition of $\bar{\mathbf{Q}}$ into $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}$, $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}$, $\bar{\mathbf{Q}}_{\mathcal{S}\mathcal{S}}$; 30 Compute Cholesky factor of $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}$: $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}}$; 31 Compute $\mathbf{A} = \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}$, using $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}}$, $\mathbf{B} = \bar{\mathbf{Q}}_{\mathcal{S}\mathcal{S}}^{-1} \mathbf{A}$, $\mathbf{a} = \mathbf{A}^\top ([\mathbf{Q}_\epsilon]_{\mathcal{S}\mathcal{S}} (\bar{Y}(\mathbf{x}_{\mathcal{S}}) - \boldsymbol{\mu}_{\mathcal{S}}))$; 32 end 33 Return $\tilde{\mathbf{x}}$ as the estimated optimal solution; 	<div style="display: flex; flex-direction: column; align-items: center; justify-content: center;"> <div style="margin-bottom: 20px;">}</div> <div style="margin-bottom: 20px;">}</div> <div style="margin-bottom: 20px;">}</div> <div style="margin-bottom: 20px;">}</div> </div>	<p>Initialization</p> <p>Rapid search</p> <p>Global search</p>
---	---	--

we obtain the following expression for Σ via standard block-matrix inversion:

$$\begin{aligned} \Sigma &= \begin{bmatrix} \sum_{\mathcal{F}\mathcal{F}} & \sum_{\mathcal{F}\mathcal{S}} \\ \sum_{\mathcal{F}\mathcal{F}}^\top & \sum_{\mathcal{F}\mathcal{S}} \end{bmatrix} \\ &= \begin{bmatrix} \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} + \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}} \sum_{\mathcal{F}\mathcal{F}} \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}^\top & -\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}} \sum_{\mathcal{F}\mathcal{F}} \\ -\sum_{\mathcal{F}\mathcal{F}} \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}^\top \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} & \sum_{\mathcal{F}\mathcal{S}} \end{bmatrix}, \end{aligned} \quad (3)$$

where $\Sigma_{\mathcal{F}\mathcal{F}} = (\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}} - \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^\top \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}^{-1} \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}})^{-1}$ is the covariance matrix of the search set. Our focus is on $\Sigma_{\mathcal{F}\mathcal{F}}$ during the rapid-search iterations. Recall that before beginning rapid search, rGMIA computes intermediate matrices \mathbf{A} and \mathbf{B} . These contain information to compute $\mathbf{V}(\mathbf{x}_{\mathcal{F}})$ and $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{F}})$ during rapid-search iterations without updating $\mathbf{V}(\mathbf{x}_{\mathcal{F}})$ and $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{F}})$. Because only solutions in \mathcal{S} are simulated, $\mathbf{B} = \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}^\top \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}^{-1} \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}$ remains unchanged during rapid search and needs to be computed only once at the end of the previous global-search iteration. In addition, we retain the Cholesky factor of $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}$ (that is, $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}}$ such that $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}} = \mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}} \mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}}^\top$), as well as $\mathbf{A} = \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}$ because they are needed to update the exact conditional means and variances of the solutions in \mathcal{F} efficiently in the next global iteration.

Like the conditional covariance matrix, we partition the conditional mean vector $\mathbf{M}(\mathbf{x}_{\mathcal{X}})$:

$$\begin{aligned} \mathbf{M}(\mathbf{x}_{\mathcal{X}}) &= \begin{bmatrix} \boldsymbol{\mu}_{\mathcal{F}} \\ \boldsymbol{\mu}_{\mathcal{S}} \end{bmatrix} + \begin{bmatrix} \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}} & \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}} \\ \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}^\top & \bar{\mathbf{Q}}_{\mathcal{S}\mathcal{S}} \end{bmatrix}^{-1} \begin{bmatrix} [\mathbf{Q}_\epsilon]_{\mathcal{F}\mathcal{F}} & \mathbf{0}_{n_{\mathcal{F}} \times n_{\mathcal{S}}} \\ \mathbf{0}_{n_{\mathcal{S}} \times n_{\mathcal{F}}} & [\mathbf{Q}_\epsilon]_{\mathcal{S}\mathcal{S}} \end{bmatrix} \\ &\quad \times \left(\begin{bmatrix} \bar{\mathbf{Y}}(\mathbf{x}_{\mathcal{F}}) \\ \bar{\mathbf{Y}}(\mathbf{x}_{\mathcal{S}}) \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}_{\mathcal{F}} \\ \boldsymbol{\mu}_{\mathcal{S}} \end{bmatrix} \right), \end{aligned} \quad (4)$$

where $n_{\mathcal{S}} = |\mathcal{S}|$, $n_{\mathcal{F}} = |\mathcal{F}|$, $[\mathbf{Q}_\epsilon]_{\mathcal{F}\mathcal{F}}$ and $[\mathbf{Q}_\epsilon]_{\mathcal{S}\mathcal{S}}$ are block matrices of \mathbf{Q}_ϵ corresponding to \mathcal{F} and \mathcal{S} , and $\{\bar{\mathbf{Y}}(\mathbf{x}_{\mathcal{F}}), \bar{\mathbf{Y}}(\mathbf{x}_{\mathcal{S}})\}$ and $\{\boldsymbol{\mu}_{\mathcal{F}}, \boldsymbol{\mu}_{\mathcal{S}}\}$ are subvectors of $\bar{\mathbf{Y}}$ and $\boldsymbol{\mu}$, respectively. Thus,

$$\begin{aligned} \mathbf{M}(\mathbf{x}_{\mathcal{S}}) &= \boldsymbol{\mu}_{\mathcal{S}} + \sum_{\mathcal{F}\mathcal{S}} ([\mathbf{Q}_\epsilon]_{\mathcal{S}\mathcal{S}} (\bar{\mathbf{Y}}(\mathbf{x}_{\mathcal{S}}) - \boldsymbol{\mu}_{\mathcal{S}}) \\ &\quad - \mathbf{A}^\top [\mathbf{Q}_\epsilon]_{\mathcal{F}\mathcal{F}} (\bar{\mathbf{Y}}(\mathbf{x}_{\mathcal{F}}) - \boldsymbol{\mu}_{\mathcal{F}})). \end{aligned} \quad (5)$$

During the rapid search, only $[\mathbf{Q}_\epsilon]_{\mathcal{S}\mathcal{S}}$ and $\bar{\mathbf{Y}}(\mathbf{x}_{\mathcal{S}})$ change, whereas $\mathbf{a} = \mathbf{A}^\top [\mathbf{Q}_\epsilon]_{\mathcal{F}\mathcal{F}} (\bar{\mathbf{Y}}(\mathbf{x}_{\mathcal{F}}) - \boldsymbol{\mu}_{\mathcal{F}})$ remains unchanged; \mathbf{A} , \mathbf{B} , $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}}$, and \mathbf{a} are intermediate matrices that we store in memory at the end of each global-search iteration. In the following sections, we discuss the computational details of rapid-search and global-search iterations.

6.1. Rapid Search

During the rapid-search iterations, we replace sparse-matrix inversions of very large $\bar{\mathbf{Q}}$ with dense inversions of very small $\Sigma_{\mathcal{F}\mathcal{F}}$. From (3), $\Sigma_{\mathcal{F}\mathcal{F}} = (\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}} - \mathbf{B})^{-1}$, which is performed in Step 9 of Algorithm 2. By

construction, $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}$ is a sparse matrix, but \mathbf{B} may be dense. Hence, the floating point operation (flop) count of computing $\Sigma_{\mathcal{F}\mathcal{F}}$ is $\mathcal{O}(n_{\mathcal{F}}^3)$. Following directly from (5), Step 10 computes $\mathbf{M}(\mathbf{x}_{\mathcal{S}})$ by multiplying the dense $n_{\mathcal{F}} \times n_{\mathcal{S}}$ matrix $\Sigma_{\mathcal{F}\mathcal{F}}$ by a vector, which costs $\mathcal{O}(n_{\mathcal{F}}^2)$. Thus, the overall cost of a single rapid-search iteration is $\mathcal{O}(n_{\mathcal{F}}^3)$. Compared with a single iteration of GMIA, this can be made much cheaper by choosing the size of the search set $n_{\mathcal{S}} \ll n$. Later we consider $n_{\mathcal{S}}$ ranging from 50–200.

Rapid-search iterations continue until the termination criterion is reached in Step 6. We propose two candidate termination criteria and evaluate their performance empirically in Section 7. The first is to employ a fixed $p - 1$ iterations of rapid search, implying that global search is repeated every p iterations. There is a trade-off between large versus small p . The former brings greater computational savings for inference by restricting the search to be within \mathcal{S} longer; however, effectiveness of the search will diminish if p is so large that there is not much information left to gain from this set. Determining the best value of p is difficult without complete knowledge of the response surface of the problem as well as the stochastic error variance at the solutions. Also, the best p may be different late in the search as opposed to earlier. We show later that $p = n_{\mathcal{S}}$ is often a reasonable choice.

The second criterion is to stop simulating within the current search set \mathcal{S} based on optimality-gap inference. Consider the following thought experiment: If we also knew the CEIs of solutions in the fixed set \mathcal{F} at every rapid-search iteration, then we would escape from \mathcal{S} when all of the CEIs of solutions $\mathbf{x}_{\mathcal{S}}$ fall below the maximum CEI in $\mathbf{x}_{\mathcal{F}}$. As an approximation of this ideal choice, we instead escape \mathcal{S} when $\max_{\mathbf{x}_{\mathcal{S}} \in \mathcal{S} \setminus \tilde{\mathbf{x}}} \text{CEI}(\tilde{\mathbf{x}}, \mathbf{x}) < \gamma$, where γ is a small positive number. In words, we stop searching within \mathcal{S} when the CEIs of solutions within \mathcal{S} fall below a threshold, γ , as it implies that only marginal reduction in the optimality gap is expected by further exploring \mathcal{S} . We refer to this criterion as the *adaptive scheme*. A sensible choice for γ is the maximum CEI of the solutions in \mathcal{F} at the last global-search iteration. Other choices of γ are possible, but our results (Lemma 1 and Theorem 1) were developed with this choice in mind. Clearly, this is not the same as the true maximum CEI of the solutions in \mathcal{F} , as it does not reflect the new simulation results obtained during the rapid-search iterations, and it is calculated with respect to the best solution at the time of the last global iteration, which may have changed. Nevertheless, this threshold is a strong indicator that greater improvement might be obtained by exploring solutions in \mathcal{F} .

6.2. Global Search

When the rapid-search termination criterion is met, rGMIA switches to global search, first selecting $\tilde{\mathbf{x}}$

among *all* simulated solutions in \mathcal{X} in Step 15, then proceeding to compute the CEIs for *all* solutions. Although one might be tempted to compute CEIs of all solutions as in Steps 3 and 4 in the initialization phase of rGMIA, this involves factorizing $\bar{\mathbf{Q}}$ and computing $\text{diag}(\bar{\mathbf{Q}}^{-1})$. Then, after choosing \mathcal{S} and \mathcal{F} , we would once again need to factorize $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}$ and $\text{diag}(\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1})$ to set up the rapid-search iterations. To avoid doing these expensive computations twice, rGMIA computes the CEIs of all solutions without factorizing $\bar{\mathbf{Q}}$, but using the matrices computed in the previous global-search iteration and the last rapid-search iteration. In the following, we explain this scheme in detail.

Steps 16 and 17 compute $\mathbf{V}(\mathbf{x}_{\mathcal{G}})$ and $\mathbf{M}(\mathbf{x}_{\mathcal{G}})$ in the same way as in Steps 9 and 10 of rapid search. Steps 18 and 19 compute $\mathbf{V}(\mathbf{x}_{\mathcal{F}})$ and $\mathbf{M}(\mathbf{x}_{\mathcal{F}})$, respectively. From (3),

$$\begin{aligned} \sum_{\mathcal{F}\mathcal{F}} &= \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} + \mathbf{A}(\bar{\mathbf{Q}}_{\mathcal{G}\mathcal{G}} - \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{G}}^{\top} \mathbf{A})^{-1} \mathbf{A}^{\top} \\ &= \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} + \mathbf{A} \sum_{\mathcal{G}\mathcal{G}} \mathbf{A}^{\top}. \end{aligned} \quad (6)$$

Because $\mathbf{V}(\mathbf{x}_{\mathcal{F}}) = \text{diag}(\Sigma_{\mathcal{F}\mathcal{F}})$, we have $\mathbf{V}(\mathbf{x}_{\mathcal{F}}) = \text{diag}(\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1}) + \text{diag}(\mathbf{A} \sum_{\mathcal{G}\mathcal{G}} \mathbf{A}^{\top})$. Recall that $\mathbf{A} = \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{G}}$ is computed and saved from the previous global-search iteration. Further, $\text{diag}(\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1})$ can be computed by performing a selected inverse, as discussed in Semelhago et al. (2017), using the Cholesky factor of $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}$ saved from the previous iteration. Moreover, $\text{diag}(\mathbf{A} \sum_{\mathcal{G}\mathcal{G}} \mathbf{A}^{\top})$ can be obtained efficiently without computing the entire matrix by exploiting that the i th diagonal element of $\mathbf{A} \sum_{\mathcal{G}\mathcal{G}} \mathbf{A}^{\top}$ is equal to the sum of squared elements of the i th column vector of $\mathbf{A} \mathbf{L}_{\Sigma_{\mathcal{G}\mathcal{G}}}$, where $\mathbf{L}_{\Sigma_{\mathcal{G}\mathcal{G}}}$ is the lower Cholesky factor of $\Sigma_{\mathcal{G}\mathcal{G}}$. This operation costs $\mathcal{O}(n_{\mathcal{G}}^3)$ flops, whereas fully computing $\mathbf{A} \sum_{\mathcal{G}\mathcal{G}} \mathbf{A}^{\top}$ requires $\mathcal{O}(n_{\mathcal{G}}^2 n)$. From (4),

$$\begin{aligned} \mathbf{M}(\mathbf{x}_{\mathcal{F}}) &= \boldsymbol{\mu}_{\mathcal{F}} + \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} [\mathbf{Q}_{\epsilon}]_{\mathcal{F}\mathcal{F}} (\bar{\mathbf{Y}}(\mathbf{x}_{\mathcal{F}}) - \boldsymbol{\mu}_{\mathcal{F}}) \\ &\quad - \mathbf{A}(\mathbf{M}(\mathbf{x}_{\mathcal{G}}) - \boldsymbol{\mu}_{\mathcal{G}}). \end{aligned} \quad (7)$$

Notice that $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} [\mathbf{Q}_{\epsilon}]_{\mathcal{F}\mathcal{F}} (\bar{\mathbf{Y}}(\mathbf{x}_{\mathcal{F}}) - \boldsymbol{\mu}_{\mathcal{F}})$ can be computed efficiently by solving $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}} \mathbf{z} = [\mathbf{Q}_{\epsilon}]_{\mathcal{F}\mathcal{F}} (\bar{\mathbf{Y}}(\mathbf{x}_{\mathcal{F}}) - \boldsymbol{\mu}_{\mathcal{F}})$ for \mathbf{z} using the Cholesky factor of $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}$. Thus, the only remaining pieces needed for CEI computation are the covariance vectors.

Because $\tilde{\mathbf{x}}$ is selected globally in the global-search iteration, $\tilde{\mathbf{x}}$ can be in either \mathcal{S} or \mathcal{F} . This does not affect the way conditional variances and conditional means are calculated; however, it does affect the way the covariance vectors, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{G}})$ and $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{F}})$, are computed. When $\tilde{\mathbf{x}} \in \mathcal{S}$, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{G}})$ is simply $[\Sigma_{\mathcal{G}\mathcal{G}}]_{\tilde{\mathbf{x}}}$, the column of $\Sigma_{\mathcal{G}\mathcal{G}}$ corresponding to $\tilde{\mathbf{x}}$. Also, from (3),

$$\begin{aligned} \sum_{\mathcal{F}\mathcal{G}} &= -\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{G}} (\bar{\mathbf{Q}}_{\mathcal{G}\mathcal{G}} - \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{G}}^{\top} \mathbf{A})^{-1} = -\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{G}} \sum_{\mathcal{G}\mathcal{G}} \\ &= -\mathbf{A} \sum_{\mathcal{G}\mathcal{G}}. \end{aligned}$$

Therefore, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{G}}) = -\mathbf{A}[\Sigma_{\mathcal{G}\mathcal{G}}]_{\tilde{\mathbf{x}}}$. These are computed in Steps 21 and 22.

When $\tilde{\mathbf{x}} \in \mathcal{F}$, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{G}})$ is a column of $\Sigma_{\mathcal{G}\mathcal{F}}$ corresponding to $\tilde{\mathbf{x}}$. Because $\Sigma_{\mathcal{G}\mathcal{F}} = -\Sigma_{\mathcal{G}\mathcal{G}} \mathbf{A}^{\top}$, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{G}}) = -\Sigma_{\mathcal{G}\mathcal{G}} [\mathbf{A}^{\top}]_{\tilde{\mathbf{x}}}$. Similarly, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{F}})$ is a column of $\Sigma_{\mathcal{F}\mathcal{F}}$ corresponding to $\tilde{\mathbf{x}}$. From (6), $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{F}}) = \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} \mathbf{e}_{\tilde{\mathbf{x}}} + \mathbf{A} \sum_{\mathcal{G}\mathcal{G}} [\mathbf{A}^{\top}]_{\tilde{\mathbf{x}}}$. Again, $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} \mathbf{e}_{\tilde{\mathbf{x}}}$ can be computed efficiently by solving $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}} \mathbf{z} = \mathbf{e}_{\tilde{\mathbf{x}}}$ for \mathbf{z} . Steps 24 and 25 perform these computations.

Combining these pieces, rGMIA computes the CEIs for all solutions in \mathcal{X} and constructs a new $\{\mathcal{F}, \mathcal{S}\}$ partition in Steps 28 and 29. Finally, the intermediate matrices are recomputed according to the new partition and stored for the next global-search iteration.

The most expensive calculations during a global-search iteration are the Cholesky factorization of $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}$, performing a selected inverse to compute $\text{diag}(\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1})$ and solving a linear system of equations with $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}$. We use the PARDISO software (Schenk and Gärtner 2018) to perform these calculations, which improves their efficiency by preprocessing large matrices such as $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}$. Unfortunately, this makes it difficult to characterize the flops required by these calculations. Therefore, we conducted timing experiments to estimate how the computation times scale as the number of feasible solutions and problem dimension grow; see Appendix B of the online supplement for the results.

Despite the lack of explicit flop counts for PARDISO calculations, we can still characterize the computational savings attained by rGMIA compared with GMIA by parameterizing the flop counts for computing $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}}$, the Cholesky factor of $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}$, for performing a selected inverse to obtain $\text{diag}(\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1})$ given $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}}$, and for solving a single-column right-hand-side linear system involving $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}$ given $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}}$; we denote these by $C_F = C_F(\mathcal{G}_{\mathcal{F}})$, $C_I = C_I(\mathcal{G}_{\mathcal{F}})$, and $C_L = C_L(\mathcal{G}_{\mathcal{F}})$, respectively. Note that $\mathcal{G}_{\mathcal{F}}$ is the induced graph of solutions in \mathcal{F} associated with the GMRF that uniquely specifies the sparsity pattern of $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}$ and thus determines the cost of performing these matrix operations.

As previously characterized for rapid-search iterations, computing $\mathbf{V}(\mathbf{x}_{\mathcal{G}})$ and $\mathbf{M}(\mathbf{x}_{\mathcal{G}})$ costs $\mathcal{O}(n_{\mathcal{G}}^3)$ flops. To compute $\mathbf{V}(\mathbf{x}_{\mathcal{F}})$, it costs C_I for $\text{diag}(\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1})$ and $\mathcal{O}(n_{\mathcal{G}}^3)$ flops for $\text{diag}(\mathbf{A} \sum_{\mathcal{G}\mathcal{G}} \mathbf{A}^{\top})$. For $\mathbf{M}(\mathbf{x}_{\mathcal{F}})$, it costs C_L for solving a system of linear equations and $\mathcal{O}(n_{\mathcal{G}} n)$ for the matrix-vector multiplication in (7). The cost for covariance vector computation depends on whether $\tilde{\mathbf{x}}$ is selected in \mathcal{S} or \mathcal{F} ; the latter case is the most expensive, costing $C_L + \mathcal{O}(n_{\mathcal{G}} n)$ flops. In our numerical experiments, we observed that $\tilde{\mathbf{x}}$ tends to remain in \mathcal{S} in later iterations. Finally, computing the intermediate matrices requires $C_F + n_{\mathcal{G}} C_L + \mathcal{O}(n_{\mathcal{G}}^2 n)$.

To summarize, a single global-search iteration incurs a cost of $C_F + C_I + (n_{\mathcal{G}} + 2)C_L + \mathcal{O}(n_{\mathcal{G}}^2 n)$ flops. See Appendix B of the online supplement for a more detailed analysis.

6.3. rGMIA vs. GMIA

To illustrate the computational savings of rGMIA, we analyze how the number of flops grows for both GMIA and rGMIA as n increases. Recall that GMIA factorizes $\bar{\mathbf{Q}}$ at every iteration to compute $\text{diag}(\bar{\mathbf{Q}}^{-1})$ and $\mathbf{M}(\mathbf{x}_{\mathcal{G}})$. Thus, per-iteration cost of GMIA is $\mathcal{O}(C_F(\mathcal{G}) + C_I(\mathcal{G}) + C_L(\mathcal{G}) + n)$, where $\mathcal{O}(n)$ comes from computing $\mathbf{Q}_\epsilon(\bar{\mathbf{Y}} - \boldsymbol{\mu})$ in Step 7 of Algorithm 1. Although $C_F(\mathcal{G}_{\mathcal{F}}) \neq C_F(\mathcal{G})$, their difference is negligible as \mathcal{F} includes most of the solutions in \mathcal{X} .

In rGMIA, for a cycle of $p - 1$ rapid-search iterations and one global-search iteration, the per-iteration cost grows as $\mathcal{O}(n_{\mathcal{G}}^2 + (C_F + C_I + n_{\mathcal{G}}C_L + n_{\mathcal{G}}^2n)/p)$. Recall that $n_{\mathcal{G}}$ is small by construction of \mathcal{S} and C_F, C_I, C_L , and n are relatively large. In fact, C_F, C_I , and C_L grow at a rate at least as fast as, and often faster than, n (see Appendix B of the online supplement for evidence), suggesting that p should be chosen large to mitigate the per-iteration cost. Immediately, we see that performing $p - 1$ rapid-search iterations amortizes the cost of performing the expensive operations during the global-search iteration. As the problem size grows, if we allow p to grow as quickly as C_F, C_I , and C_L grow, then we can control the cost of expensive matrix operations in global-search iterations by performing many rapid-search iterations cheaply. No such control is available in GMIA, and the number of flops simply grows without bound. From a computational standpoint, this explains the power of rGMIA.

To give a sense of the relative time cost of rapid-search versus global-search computations, consider $\bar{\mathbf{Q}}$ associated with a two-dimensional DOvS problem having a 1000×1000 feasible region and a randomly selected search set \mathcal{S} with $n_{\mathcal{G}} = 100$. The global calculations of matrix factorization, selected inverse to obtain the diagonal elements, and solving a single-column right-hand-side linear system, performed by PARDISO over 100 trials, took on average 31.17 seconds (0.16 seconds), 44.55 seconds (0.27 seconds), and 1.09 seconds (0.02 seconds), respectively, with standard errors in parentheses. Compare this to the rapid-search operation of computing the inverse of a dense $n_{\mathcal{G}} \times n_{\mathcal{G}}$ matrix. Using MATLAB, with $n_{\mathcal{G}} = 100$, such an operation took on average 0.2203 seconds (0.0087 seconds) over 100 trials. Clearly, global-search operations are the bottleneck, and they become even more significant as problem size and dimension increases. More results demonstrating this are found in Appendix B of the online supplement.

Salemi et al. (2019) show that GMIA without a stopping criterion simulates each solution $\mathbf{x} \in \mathcal{X}$ infinitely often with probability 1 as the number of iterations goes to infinity. This establishes global convergence via the strong law of large numbers. Here, we show that with

far superior computational efficiency—demonstrated empirically in Section 7—rGMIA still achieves global convergence for either the fixed- p or adaptive schemes; see Appendix A of the online supplement for the proofs. To begin, we introduce the following lemma:

Lemma 1. *At any iteration of GMIA or global-search iteration of rGMIA, $\text{CEI}(\tilde{\mathbf{x}}, \mathbf{x}) > 0, \forall \mathbf{x} \in \mathcal{X} \setminus \tilde{\mathbf{x}}$ with probability 1.*

This lemma guarantees that, in the adaptive scheme, our choice of $\gamma = \max_{\mathbf{x} \in \mathcal{F}} \text{CEI}_t(\tilde{\mathbf{x}}, \mathbf{x})$ will be positive with probability 1 after any finite number of iterations of rGMIA. With the aid of Lemma 1, we establish global convergence of rGMIA using only the assumptions presented in Salemi et al. (2019) to prove convergence of GMIA as stated below.

Theorem 1. *Assume (i) $y(\mathbf{x}) > -\infty, \forall \mathbf{x} \in \mathcal{X}$, (ii) $0 < \text{Var}[Y(\mathbf{x})] < \infty, \forall \mathbf{x} \in \mathcal{X}$, and (iii) the initially estimated $\mathbf{Q}(\hat{\boldsymbol{\theta}})$ is positive definite and not updated, where $\hat{\boldsymbol{\theta}}$ are parameter estimates. Given assumptions (i)–(iii), rGMIA, implemented with either the adaptive or fixed- $p < \infty$ scheme and without a stopping condition, simulates each solution $\mathbf{x} \in \mathcal{X}$ infinitely often with probability 1 as the number of iterations goes to infinity.*

7. Empirical Evaluation

We use three test problems to evaluate different aspects of the performance of rGMIA. The first is an (s, S) inventory optimization problem from Koenig and Law (1985), which has characteristics of a practical DOvS problem and has already been used to test the behavior of GMRF-based optimization algorithms in Salemi et al. (2019). The objective function is the expected average cost per period of the inventory system over 30 periods. To obtain a rectangular feasible region, we choose the decision variables to be s and $S - s$. We test two different sized feasible regions: **inventory_100** covering solutions $s \times (S - s) = [1, 2, \dots, 100] \times [1, 2, \dots, 100]$, and **inventory_150** covering solutions $s \times (S - s) = [1, 2, \dots, 150] \times [1, 2, \dots, 150]$. The optimal solution in both cases is $s = 17$ and $S - s = 36$ with an estimated expected average cost per period of \$106.14 based on 500,000 replications at each feasible solution.

The second problem is based on a modified Griewank function; see Bingham and Surjanovic (2017) for a description. The Griewank function is a popular test problem because of its many local minima. We slightly modified the parameters of this function to make the range larger and the global minimum more distinguishable. We chose the domain of the Griewank function to be $[-5, 5] \times [-5, 5]$ in which it has 5 local minima with the global minimum at $(0, 0)$. The range of the function is $[0, 2.5490]$. The four local minima have response values of 0.6828, compared with 0 for the global minimum. To create DOvS

problems based on this surface we project it onto lattices of varying resolution, resulting in four problems with feasible regions of increasing size: **griewank_101** ($101 \times 101 = 10,201$ solutions), **griewank_201** ($201 \times 201 = 40,401$ solutions), **griewank_301** ($301 \times 301 = 90,601$ solutions), and **griewank_401** ($401 \times 401 = 160,801$ solutions). To make it stochastic, we added independent $N(0, 10^{-4})$ simulation noise to the response function, mimicking the behavior of a DOVS problem. Much of the variability in this problem is driven by the nature of the surface rather than that of the stochastic simulation noise.

The third problem is “restaurant seating” modified from a problem available in the SimOpt.org library (Pasupathy and Henderson 2006): Suppose a restaurant has the objective of maximizing profit (or minimizing negative profit). There are d different sizes of tables, $s_i, i = 1, 2, \dots, d$, and we are to decide how many of each size of table to make available, x_{s_i} . Customers arrive in groups that range in size from 1 to s_d and are seated instantly at the smallest available table that can seat the entire group. Successfully seating a group results in revenue r , in \$1,000s, per person. Groups that find no available table upon arrival leave without waiting. Keeping a size- s_i table costs $c_{s_i} \times \$1,000/\text{hour}$. The restaurant runs continuously for T hours. We consider three different problems, **restaurant_125**, **restaurant_25**, and **restaurant_5**, each having 15,625 feasible solutions, but of different dimensions: $d = 2, 3$, and 6 , respectively. Table 5 in Appendix D of the online supplement outlines the parameters used for each problem.

For all experiments, 10 replications were obtained at each simulated solution on first visit, and 2 additional replications on subsequent visits. MLEs of the GMRF parameters were estimated using a Latin hypercube sample of $10d$ feasible solutions, where d is the problem dimension. Experiments were run using a high-performance computing cluster (HPCC) consisting of three compute nodes, each with 40 cores and 256 GB of RAM, and a head node that has 20 cores and 256 GB of RAM. For each experiment, we ran 30 macro-replications, setting different random number streams for each run and assigning a single core for each macro-replication with sufficient memory to successfully perform the experiment.

7.1. Comparing rGMIA to GMIA

We compare the performance of rGMIA to GMIA considering both fixed-precision and fixed-budget paradigms. The version of GMIA used for comparison adopts the smart sparse linear algebra techniques discussed in Semelhago et al. (2017). We use the inventory and restaurant problems in the former setting, where we evaluate the time until termination and the resulting achieved optimality gap of the estimated

optimal solution given desired gaps of $\delta = 0.1, 0.05, 0.01$. We use the Griewank problem in the fixed-budget setting with a time budget of one hour, comparing the achieved optimality gap after the budget has been exhausted for problems of increasing size. To simplify the comparisons, we ran rGMIA for a fixed search set size $n_g = 50$ with $p = 10, 25, 50, 100, 200$ rapid-search iterations per global-search iteration, and the adaptive scheme. Results in Tables 1–3 indicate that $p = 50$ performs especially well. For (favorable) comparisons of the GMIA approach with other Bayesian optimization algorithms, see Salemi et al. (2019). The focus of this paper is providing a computationally superior way to achieve the same search progress and inference.

Table 1 contains the results of fixed-precision GMIA and rGMIA applied to the inventory problem. In each sub-table, we record the mean and maximum run times, mean and maximum achieved optimality gaps, and mean number of iterations until stopping across 30 macro-replications. “Optimality gap” here refers to the difference between the true response at the estimated optimal solution and the true minimum of the response surface. Each column specifies an algorithm and the desired acceptable optimality gap, δ . The inventory problems are low-dimensional and have smaller numbers of solutions compared with other test problems. However, even in this setting with relatively cheap computational overhead, Table 1 shows that GMIA’s mean run time is almost an order of magnitude greater than rGMIA across every choice of p or the adaptive scheme. Such differences in mean run time become larger as the problem size increases (see **inventory_100** versus **inventory_150**). Consider the scenario where a user wishes to solve the **inventory_150** problem to fixed precision given $\delta = 0.01$ and must purchase processor time on an HPCC at an hourly rate. The user of GMIA would potentially be required to purchase almost 3 hours of run time, corresponding to the maximum observed run time in our experiment (10,068.06 seconds). Whereas, for rGMIA with $p = 50$, the maximum observed run time is under 11 minutes; 16 times faster than GMIA. An outlier macro-replication was removed from the **inventory_150** results. The design points placed in this run resulted in MLEs that mischaracterized the surface, highlighting a challenge in initial parameter estimation for both GMIA and rGMIA; they completed only a single iteration before attaining a maximum CEI < 0.05 , terminating with an achieved optimality gap of 8.51.

Table 2 highlights the advantage rGMIA has in a fixed-budget setting using the Griewank problems. For each problem and algorithm, we examine the achieved optimality gap at termination and number of iterations that are performed across 30 macro-replications

Table 1. Fixed-Precision Results Averaged from 30 Macroreplications of rGMIA and GMIA Applied to the inventory_100 and inventory_150 Problems

δ	inventory_100																				
	0.1				0.05				0.01												
	$p = 10$	$p = 25$	$p = 50$	$p = 100$	Adaptive	GMIA	$p = 10$	$p = 25$	$p = 50$	$p = 100$	Adaptive	GMIA	$p = 10$	$p = 25$	$p = 50$	$p = 100$	Adaptive	GMIA			
Problem Algorithm																					
Mean time until termination (s)	247 (11)	121 (4)	102 (4)	124 (4)	173 (6)	140 (5)	1186 (42)	277 (11)	135 (4)	112 (4)	134 (4)	186 (6)	154 (5)	1333 (41)	382 (15)	181 (5)	140 (4)	155 (4)	213 (6)	182 (47)	
Maximum time until termination (s)	410	167	144	174	232	191	1618	454	180	154	183	242	204	1744	590	251	178	200	260	223	2305
Mean optimality gap	0.09 (0.02)	0.11 (0.03)	0.08 (0.02)	0.07 (0.01)	0.04 (0.01)	0.24 (0.09)	0.13 (0.03)	0.06 (0.01)	0.07 (0.02)	0.04 (0.01)	0.09 (0.02)	0.03 (0.01)	0.08 (0.02)	0.16 (0.04)	0.03 (0.01)	0.04 (0.01)	0.03 (0.01)	0.05 (0.01)	0.02 (0.00)	0.06 (0.02)	0.05 (0.01)
Maximum optimality gap	0.33	0.75	0.64	0.14	0.13	2.08	0.77	0.22	0.52	0.23	0.49	0.13	0.53	0.93	0.13	0.23	0.12	0.23	0.11	0.53	0.30
Mean number of iterations	6684 (236)	6698 (236)	7146 (230)	14034 (452)	27988 (904)	6709 (236)	6722 (238)	7512 (242)	7486 (233)	7868 (229)	15238 (442)	30168 (872)	7530 (237)	7592 (236)	10370 (302)	10244 (267)	10233 (247)	17818 (394)	34561 (799)	9994 (261)	10498 (279)

δ	inventory_150																				
	0.1				0.05				0.01												
	$p = 10$	$p = 25$	$p = 50$	$p = 100$	Adaptive	GMIA	$p = 10$	$p = 25$	$p = 50$	$p = 100$	Adaptive	GMIA	$p = 10$	$p = 25$	$p = 50$	$p = 100$	Adaptive	GMIA			
Problem Algorithm																					
Mean time until termination (s)	1207 (67)	603 (36)	352 (20)	412 (25)	540 (34)	403 (26)	5326 (335)	1318 (66)	664 (36)	386 (20)	450 (24)	588 (34)	436 (26)	5902 (326)	1592 (65)	821 (35)	471 (19)	522 (24)	674 (32)	495 (26)	7375 (319)
Maximum time until termination (s)	1805	963	535	628	899	591	9560	1898	1028	568	652	922	619	9560	2107	1159	633	696	964	676	10068
Mean optimality gap	0.04 (0.01)	0.12 (0.03)	0.05 (0.01)	0.05 (0.01)	0.03 (0.01)	0.12 (0.03)	0.12 (0.03)	0.07 (0.02)	0.08 (0.02)	0.08 (0.03)	0.06 (0.02)	0.03 (0.01)	0.05 (0.01)	0.09 (0.02)	0.04 (0.01)	0.04 (0.01)	0.03 (0.01)	0.02 (0.01)	0.01 (0.00)	0.03 (0.01)	0.06 (0.02)
Maximum optimality gap	0.25	0.80	0.14	0.23	0.14	0.64	0.62	0.30	0.30	0.74	0.49	0.14	0.13	0.57	0.19	0.14	0.25	0.13	0.07	0.19	0.64
Mean number of iterations	14002 (810)	14069 (805)	14770 (803)	29235 (1624)	58477 (3215)	14122 (804)	14031 (816)	15500 (792)	15525 (789)	16199 (777)	31784 (1543)	63470 (3079)	15546 (802)	15572 (791)	19357 (767)	19354 (713)	19775 (692)	36718 (1469)	72367 (2832)	19313 (745)	19522 (760)

Notes. Standard errors of mean values are provided in parentheses. Reported results for **inventory_150** averaged across 29 macroreplication with outlier macroreplication removed. This is further explained in Section 7.1.

Table 2. Fixed-Budget Results Averaged from 30 Macroreplications of rGMIA ($|S| = 50$) and GMIA Applied to the **griewank_101**, **griewank_201**, **griewank_301**, and **griewank_401** Problems, Given a 1-Hour Time Budget

Algorithm	$p = 10$	$p = 25$	$p = 50$	$p = 100$	$p = 200$	Adaptive	GMIA	$p = 10$	$p = 25$	$p = 50$	$p = 100$	$p = 200$	Adaptive	GMIA
Problem	griewank_101													
Mean optimality gap	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	3.17E-4 (7.70E-5)	3.60E-4 (8.30E-5)	0 (0)	0 (0)	0 (0)	0 (0)	5.64E-4 (9.07E-5)
Maximum optimality gap	0	0	0	0	0	0	0	1.31E-3	1.31E-3	0	0	0	0	2.00E-3
Mean number of iterations	96026 (838)	254433 (2388)	482821 (3369)	802334 (20208)	1279154 (8494)	40727 (352)	21696 (315)	17226 (261)	39209 (650)	115576 (2542)	238308 (3165)	402881 (5291)	220697 (1966)	4947 (95)
Problem	griewank_301													
Mean optimality gap	3.69E-4 (6.00E-5)	3.78E-4 (5.06E-5)	1.71E-4 (3.41E-5)	2.49E-4 (4.92E-5)	2.50E-4 (4.27E-5)	3.00E-4 (4.90E-5)	6.86E-2 (3.80E-2)	4.79E-2 (3.16E-2)	2.96E-4 (3.64E-5)	2.57E-4 (3.67E-5)	2.02E-4 (2.67E-5)	1.62E-4 (2.92E-5)	2.07E-4 (3.04E-5)	9.70E-2 (4.28E-2)
Maximum optimality gap	1.22E-3	8.89E-4	5.83E-4	8.89E-4	8.89E-4	8.89E-4	6.83E-1	6.83E-1	6.87E-4	6.87E-4	5.00E-4	5.00E-4	5.00E-4	6.83E-1
Mean number of iterations	7853 (180)	18598 (540)	34386 (838)	82451 (2042)	166888 (1445)	30675 (870)	2137 (32)	4346 (251)	10356 (546)	16698 (637)	31538 (631)	58188 (1599)	20009 (858)	928 (25)

Note. Standard errors of mean values are provided in parentheses.

after the one-hour time budget has been exhausted. Keeping dimension fixed ($d = 2$), as the number of solutions increases, it becomes more difficult to find the optimum, because (1) more simulations are required as there are more feasible solutions and (2) computational overhead for inference at each iteration increases. However, the latter affects GMIA far more than rGMIA. For example, the mean number of iterations GMIA performs within 1 hour in **griewank_401** is 1/23 of that in **griewank_101**. The impact is far milder for rGMIA; for example, the mean number of iterations of rGMIA with $p = 200$ decreases by 1/2 comparing **griewank_401** and **griewank_101**. Performing more iterations given a time budget means more simulations are made, which ultimately manifests in the optimality gap of the solution returned at termination. Even though **griewank_401** was a difficult problem to solve for all algorithms tested, we note that GMIA had a mean optimality gap that was two orders of magnitude larger than that of most settings of rGMIA.

To test the effect of increasing dimensions, we ran GMIA and rGMIA on the restaurant problems under the fixed-precision setting. Table 3 contains three subtables corresponding to **restaurant_125**, **restaurant_25**, and **restaurant_5** problems. Recall that all three problems have 15,625 solutions but have dimensions $d = 2, 3, 6$, respectively. This affects both simulation time as well as computational overhead. To ensure that optimal solutions are located in the interior of the feasible region, arrival rates were chosen to be different for each problem; see Table 5 in Appendix D of the online supplement for details. As a result, the simulation time per replication generally increases as the problem dimension decreases. On the other hand, the computational overhead increases as the precision matrix becomes denser in higher dimensions. GMIA spent 50.52%, 8.53%, and 0.18% of its run time for simulations in **restaurant_125**, **restaurant_25**, and **restaurant_5**, respectively. This reflects that as the problem's dimension increases the precision matrix becomes denser and the linear algebra in GMIA becomes more costly. For the **restaurant_125** problem, Table 3 shows that GMIA actually outperforms rGMIA by terminating sooner. In this case, the simulation is relatively more expensive than the linear algebra; thus, it is more important to select good solutions to simulate at each iteration from the entire solution space than reducing the cost of linear algebra by restricting the search. For the **restaurant_25** experiments, however, the mean time until termination of GMIA increases compared with the **restaurant_125** experiments, whereas that of rGMIA decreases. This is because the simulation is now cheaper and linear algebra is more expensive; thus, rapid search of rGMIA pays off. Recall that this combination of large computational overhead and relatively smaller

Table 3. Fixed-Precision Results Averaged from 30 Macroreplications of rGMIA ($|S| = 50$) and GMIA Applied to the restaurant_125, restaurant_25, and restaurant_5 Problems

restaurant_125																					
0.01																					
Problem Algorithm	0.1			0.05			0.01			Adaptive	GMIA										
	$p = 10$	$p = 25$	$p = 50$	$p = 100$	$p = 200$	$p = 500$	$p = 1000$	$p = 2000$	$p = 5000$			$p = 10000$	$p = 20000$								
Mean time until termination (s)	4596 (752)	4396 (716)	4309 (700)	5268 (855)	6403 (1042)	4338 (710)	3816 (629)	5047 (822)	4672 (757)	5620 (910)	7004 (1136)	5603 (809)	4221 (686)	6594 (939)	6275 (894)	6304 (828)	7077 (926)	8899 (1168)	7471 (756)	Adaptive	GMIA
Maximum time until termination (s)	9044	8518	8387	9934	12458	8670	7582	9612	8986	10344	13331	13438	8134	11360	10809	10398	11041	14267	13438	Adaptive	GMIA
Mean optimality gap	0.10 (0.02)	0.10 (0.02)	0.12 (0.02)	0.10 (0.02)	0.09 (0.02)	0.12 (0.02)	0.10 (0.02)	0.10 (0.02)	0.09 (0.02)	0.08 (0.02)	0.07 (0.02)	0.11 (0.02)	0.11 (0.02)	0.08 (0.02)	0.07 (0.01)	0.07 (0.01)	0.06 (0.01)	0.06 (0.01)	0.08 (0.01)	Adaptive	GMIA
Maximum optimality gap	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.31	0.31	0.31	0.31	0.31	0.31	Adaptive	GMIA
Mean number of iterations	5219 (851)	5228 (852)	5591 (910)	11181 (1820)	22334 (3636)	5148 (841)	5168 (851)	5783 (939)	5828 (945)	6139 (995)	12134 (1967)	24294 (3939)	9596 (2569)	5756 (935)	9050 (1280)	10113 (1332)	16304 (2133)	30641 (4020)	13965 (2434)	Adaptive	GMIA

restaurant_25																					
0.01																					
Problem Algorithm	0.1			0.05			0.01			Adaptive	GMIA										
	$p = 10$	$p = 25$	$p = 50$	$p = 100$	$p = 200$	$p = 500$	$p = 1000$	$p = 2000$	$p = 5000$			$p = 10000$	$p = 20000$								
Mean time until termination (s)	3618 (477)	2367 (313)	1447 (192)	2194 (288)	3866 (508)	2365 (324)	11356 (1581)	4592 (477)	3036 (314)	1905 (183)	2893 (269)	5123 (476)	3068 (301)	13952 (1665)	5594 (424)	3298 (215)	4415 (221)	7208 (363)	5242 (374)	Adaptive	GMIA
Maximum time until termination (s)	5831	3904	2504	3633	6285	4315	26093	6542	4338	2788	3967	6929	4723	28569	7339	4246	5192	8301	12735	Adaptive	GMIA
Mean optimality gap	0.06 (0.01)	0.08 (0.01)	0.07 (0.01)	0.07 (0.01)	0.06 (0.01)	0.07 (0.01)	0.08 (0.01)	0.07 (0.01)	0.08 (0.01)	0.06 (0.01)	0.07 (0.01)	0.05 (0.01)	0.07 (0.01)	0.07 (0.01)	0.04 (0.01)	0.05 (0.01)	0.05 (0.01)	0.04 (0.00)	0.07 (0.01)	Adaptive	GMIA
Maximum optimality gap	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.12	0.15	0.12	0.09	0.18	Adaptive	GMIA
Mean number of iterations	6189 (815)	6250 (822)	6469 (851)	13074 (1720)	26308 (3460)	6224 (819)	6092 (803)	7961 (819)	8081 (829)	8746 (814)	17304 (1611)	34848 (3240)	8314 (756)	7601 (853)	18107 (1339)	19933 (1262)	28981 (1475)	50161 (2517)	19356 (3813)	Adaptive	GMIA

restaurant_5																						
0.01																						
Problem Algorithm	0.1			0.05			0.01			Adaptive	GMIA											
	$p = 10$	$p = 25$	$p = 50$	$p = 100$	$p = 200$	$p = 500$	$p = 1000$	$p = 2000$	$p = 5000$			$p = 10000$	$p = 20000$									
Mean time until termination (s)	43699 (7595)	22664 (3952)	7221 (1256)	10212 (1760)	11717 (2015)	20095 (3836)	> 48hr (8444)	59750 (8444)	33392 (4471)	11487 (1406)	15075 (1966)	16890 (2047)	29970 (4282)	123798 (7759)	78670 (4348)	29537 (1608)	28435 (1026)	27360 (541)	56981 (3936)	Adaptive	GMIA	
Maximum time until termination (s)	92035	49310	15737	21135	23450	55710	> 48hr	105517	57054	18369	24979	26599	62941	152494	98836	37269	32685	30624	81357	Adaptive	GMIA	
Mean optimality gap	0.06 (0.01)	0.07 (0.01)	0.07 (0.01)	0.07 (0.01)	0.06 (0.01)	0.07 (0.01)	N/A	0.06 (0.01)	0.05 (0.01)	0.05 (0.01)	0.06 (0.01)	0.05 (0.00)	0.07 (0.01)	N/A	0.04 (0.00)	0.04 (0.00)	0.03 (0.00)	0.03 (0.00)	0.06 (0.01)	Adaptive	GMIA	
Maximum optimality gap	0.31	0.31	0.31	0.31	0.31	0.31	N/A	0.17	0.12	0.14	0.15	0.11	0.15	N/A	0.12	0.11	0.08	0.07	0.19	Adaptive	GMIA	
Mean number of iterations	5769 (1008)	5980 (1043)	6169 (1072)	12638 (2196)	25614 (4448)	5430 (995)	N/A	8262 (1172)	8833 (1168)	9511 (1161)	18408 (2414)	38568 (4708)	8538 (1124)	N/A	19163 (1198)	20499 (1059)	22066 (1122)	35151 (1244)	65248 (478)	17167 (1056)	Adaptive	GMIA

Notes. Standard errors of mean values are provided in parentheses. Many macroreplications of running GMIA on restaurant_5 had long run times such that the mean time elapsed until termination was > 48 hours. For this reason, full results were omitted, but this illustrates rGMIA's ability to solve problems unable to be solved by GMIA.

simulation effort is the setting for which rGMIA was proposed. Finally, the **restaurant_5** problem is higher dimensional to push the limits of what GMIA can solve. With a mean run time of over two days across 30 macroreplications for $\delta = 0.1$, GMIA effectively was unable to terminate. rGMIA was able to return an estimated optimal solution within $\delta = 0.1$ in two hours on average.

7.2. rGMIA's Performance Sensitivity to n_g and p

In this section, we investigate how rGMIA's performance is affected by the search set size. In the previous section, all experiments used search set size $n_g = 50$, and $p = 50$ rapid-search iterations showed good performance across all problems. We now vary the search set size as $n_g = 50, 100, 200$ and evaluate the performance of different choices for p , as well as the adaptive scheme, under the fixed-precision setting. We provide complete results for all of the test problems in Tables 6–14 in Appendix D of the online supplement and summarize our findings here.

Tables 6–14 show that for a given search set size n_g , $p = n_g$ is the best choice. We confirmed that in many cases when $p = n_g$, all solutions in \mathcal{S} are simulated at the end of each rapid search. We speculate that this is because the spatial diversity among the solutions in the search set overwhelms the stochastic error at each solution, which causes CEI to rank not-yet-visited solutions higher than already-simulated solutions. As a result, rGMIA tends to include many unvisited solutions in the search set at each global-search iteration and then explores all of them rather than revisiting a solution multiple times. Therefore, when $p < n_g$, we do not fully exploit the computational benefit of rapid-search iterations because there is still value in simulating the remaining unvisited solutions in \mathcal{S} . On the other hand, when $p > n_g$, rGMIA is forced to simulate the same solutions in \mathcal{S} more than once instead of exploring new solutions. Thus, the adaptive scheme does not outperform $p = n_g$.

Nonetheless, we speculate the adaptive scheme may be useful when δ is small. For example, we can observe in Table 14 in Appendix D of the online supplement that for smaller δ , the relative performance difference between the adaptive scheme and $p = n_g$ becomes smaller. This is because for smaller δ , rGMIA must evaluate more solutions to achieve the smaller acceptable optimality gap, and later iterations tend to explore solutions with poor conditional means and high uncertainty. Once some of these solutions are simulated during the rapid-search iterations, rGMIA may realize that these are in fact bad solutions and it is sensible to break out of the search set early. On the other hand, when the search set contains very good solutions then it may be worth exploiting the search set for more than p iterations to confirm a small achieved

optimality gap. This situation will also favor using the adaptive scheme over a fixed p .

From the experiment results, the best choice of n_g appears problem specific. Nevertheless, the run times indicate that the performance is not sensitive to the choice of n_g . This suggests that there is little penalty in choosing a suboptimal n_g , given that $p = n_g$.

8. Conclusions

A lingering barrier to large-scale DOvS is the inability to exploit strong problem structure to efficiently dispense with large portions of the space of feasible solutions. Inferential optimization is promising in characterizing DOvS structure *statistically* and thereby deemphasizing large portions of the space of feasible solutions with high confidence. Although the DOvS problems that can be addressed in this way are still small in dimension and number of feasible solutions relative to mathematical programming, gains thus far have been substantial.

GMIA (Salemi et al. 2019) is the current state-of-the-art in inferential optimization for DOvS. The focus of Salemi et al. (2019) was identifying and parameterizing an advantageous GP (the discrete GMRFs) and creating an acquisition function suitable for stochastic simulation (CEI). The focus of this paper is improved computational efficiency via smart computational linear algebra to greatly extend the reach of GMIA without degrading the inference. The result is a specific algorithm, rGMIA. However, the central idea of partitioning a feasible region into a search set and fixed set, and updating the conditional distributions efficiently, is generally applicable to DOvS problems that use the GP conditional distribution for inference.

To realize the full potential of inferential optimization, future work will need to address some open questions. Clearly, we need an effective strategy for allocating simulation effort (i.e., replications) to solutions. More specifically, rGMIA simulates two solutions, \tilde{x} and x^{CEI} , on each iteration, so we need to specify the number of replications to be obtained to promote search progress without wasting effort. This problem is challenging as neither EI nor CEI account for the cost of simulation or the downstream progress of the search. And although the alternative KG acquisition function does look ahead, it is only one step ahead and it does not provide optimality-gap inference.

We have thus far constructed the search set \mathcal{S} by simply selecting \tilde{x} and the solutions with the $n_g - 1$ largest CEI values. Although this method seems to be effective, there is potential for alternative constructions that might be better. This is a topic of ongoing research.

Presently, GMIA and rGMIA both assume a sequential search; that is, simulation replications are obtained sequentially on a single processor. With the

proliferation of parallel computation, it is natural to extend both algorithms to a parallel paradigm where multiple solutions or replications can be simulated simultaneously. This involves deciding which solutions to simulate in parallel and how to efficiently update relevant statistics and CEI values once the solutions have been simulated.

Finally, at the present state of development high dimension is more challenging than number of feasible solutions: \bar{Q} becomes less sparse with dimension d . Salemi et al. (2019) consider projecting less-active dimensions onto active dimensions; although this seems promising, creative ideas for addressing large d are clearly needed.

Acknowledgments

The authors thank the area editor, associate editor, and two referees for their valuable and timely feedback.

References

- Bingham D, Surjanovic S (2017) Virtual library of simulation experiments. Accessed November 7, 2019, <http://www.sfu.ca/ssurjano/griewank.html>.
- Chen Y, Ryzhov IO (2019) Complete expected improvement converges to an optimal budget allocation. *Adv. Appl. Probab.* 51(1): 209–235.
- Frazier P, Powell W, Dayanik S (2009) The knowledge-gradient policy for correlated normal beliefs. *INFORMS J. Comput.* 21(4):599–613.
- Glynn P, Juneja S (2004) A large deviations perspective on ordinal optimization. Ingalls RG, Rossetti MD, Smith JS, Peters BA, eds. *Proc. 2004 Winter Simulation Conf.* (IEEE, New York), 577–585.
- Hoffman M, Song E, Brundage M, Kumara S (2018) Condition-based maintenance policy optimization using genetic algorithms and Gaussian Markov improvement algorithm. Bregon A, Orchard M, eds. *Proc. Annual Conf. PHM Soc.*, vol. 10(1).
- Hunter SR, Nelson BL (2017) Parallel ranking and selection. Tolk A, Fowler J, Shao G, Yücesan E, eds. *Advances in Modeling and Simulation: Seminal Research from 50 Years of Winter Simulation Conferences* (Springer International Publishing, Cham, Switzerland), 249–275.
- Johnson CR (1982) Inverse M-matrices. *Linear Algebra Appl.* 47:195–216.
- Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. *J. Global Optim.* 13(4):455–492.
- Koenig LW, Law AM (1985) A procedure for selecting a subset of size m containing the l best of k independent normal populations, with applications to simulation. *Comm. Statist.* B14(3):719–734.
- Pasupathy R, Henderson SG (2006) A testbed of simulation-optimization problems. Perrone LF, Lawson BG, Liu J, Wieland FP, eds. *Proc. 2006 Winter Simulation Conf.* (IEEE, New York), 255–263.
- Rue H, Held L (2005) *Gaussian Markov Random Fields: Theory and Applications* (Chapman and Hall/CRC, New York).
- Salemi P, Song E, Nelson BL, Staum J (2019) Gaussian Markov random fields for discrete optimization via simulation: Framework and algorithms. *Oper. Res.* 67(1):250–266.
- Santner TJ, Williams BJ, Notz W (2003) *The Design and Analysis of Computer Experiments* (Springer, New York).
- Schenk O, Gärtner K (2018) PARDISO User Guide Version 6.0.0. Accessed May 19, 2020, <https://pardiso-project.org/manual/manual.pdf>.
- Semelhago M, Nelson BL, Wächter A, Song E (2017) Computational methods for optimization via simulation using Gaussian Markov random fields. Chan WKV, D’Ambrogio A, Zacharewicz G, Mustafee N, Wainer G, Page E, eds. *Proc. 2017 Winter Simulation Conf.* (IEEE, New York), 2080–2091.
- Sun L, Hong LJ, Hu Z (2014) Balancing exploitation and exploration in discrete optimization via simulation through a Gaussian process-based search. *Oper. Res.* 62(6):1416–1438.
- Xie J, Frazier P, Chick SE (2016) Bayesian optimization via simulation with pairwise sampling and correlated prior beliefs. *Oper. Res.* 64(2):542–559.