Submitted to *INFORMS Journal on Computing*
manuscript JOC-2019-07-OA-165.R1

Rapid Discrete Optimization via Simulation with Gaussian Markov Random Fields

Mark Semelhago, Barry L. Nelson
Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois, 60208
mark.semelhago@u.northwestern.edu, nelsonb@northwestern.edu

Eunhye Song
Department of Industrial and Manufacturing Engineering, The Pennsylvania State University, University Park, Pennsylvania, 16802, eus358@psu.edu

Andreas Wächter
Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois, 60208
andreas.waechter@northwestern.edu

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.

Key words: design of experiments; efficiency; statistical analysis
History: Received July 2019; revisions received November 2019, February 2020.

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 con-

trollable 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 **?**; 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 to a deterministic computer experiment. In*

combination, the computational overhead for inference is no longer negligible compared to the simulation cost.

An example of the class of problems we consider is condition-based maintenance-policy optimization, as studied in **?**: 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), $L-1,\ldots,0$ (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 (**?**). **?** 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 **??**, we review the use of GPs in DOvS algorithms. Section **??** provides the necessary background on GMRFs and complete expected improvement, a functional of the conditional distribution of the GP that

guides the search. Section **??** restates GMIA as presented in **?**. In Section **??**, we introduce rGMIA and delve into its computational details in Section **??**. In particular, we analyze the computational complexity of rGMIA relative to GMIA, and prove its global convergence. Section **??** shows numerical results, evaluating rGMIA against GMIA on carefully selected test problems, and Section **??** 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 (**?**). 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.

? 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. Since 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). **?**

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 **?** 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 **?**. Therefore, not all solutions need to be simulated to make optimality-gap inference.

Employing a very different approach, **?** 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 $j = 1, 2, \ldots,$ 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 $\mathbf{y} = [y(\mathbf{x}_1), y(\mathbf{x}_2), \ldots, y(\mathbf{x}_n)]^\top$ as a multivariate normal random vector $\mathbb{Y} = [\mathbb{Y}_1, \mathbb{Y}_2, \ldots, \mathbb{Y}_n]^\top$ with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. A GMRF, a special case of GP, is a non-degenerate $n \times 1$ 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 **?**. 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 $\mathbf{\Sigma}$ of $\mathbb{Y}$.

In general, the diagonal entries Q_{ii} of a precision matrix are such that $\text{Var}(\mathbb{Y}_i \mid \mathbb{Y}_{\mathcal{V}\setminus\{i\}}) = 1/Q_{ii}$, 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 $\text{Corr}(\mathbb{Y}_i, \mathbb{Y}_j \mid \mathbb{Y}_{\mathcal{V}\setminus\{i,j\}}) = -Q_{ij}/\sqrt{Q_{ii}Q_{jj}}$, 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 non-zero pattern of the precision matrix $\mathbf{Q}$, and vice versa, since 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: $\mathbb{Y}_i \perp \mathbb{Y}_{\mathcal{V}\setminus\{i,\mathcal{N}(i)\}} \mid \mathbb{Y}_{\mathcal{N}(i)}$ for every $i \in \mathcal{V}$, where $\mathcal{N}(i) = \{j\colon\ \{i, j\} \in \mathcal{E}\}$. This local Markovian property incapsulates 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. **?** show that a particularly effective choice is based on the ℓ_2 distance, $\mathcal{N}(\mathbf{x}) = \{\mathbf{x}' \in \mathcal{X}\colon\ ||\mathbf{x} - \mathbf{x}'||_2 = 1\}$, which implies that the fraction of non-zero 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 $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_d]^\top$. 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 jth 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 jth 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 non-stationary.

Since the conditional precisions must be positive, it follows that $\theta_0 > 0$. We also want neighbors to have non-negative conditional correlations, so $\theta_1, \theta_2, \ldots, \theta_d$ are chosen to be non-negative. Additionally, $\mathbf{Q}$ should be positive definite. With these conditions, $\mathbf{Q}$ is a non-singular M-matrix so its inverse is nonnegative (**?**). 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, $\mathbf{\Sigma} = \mathbf{Q}^{-1}$, is typically dense, as it should be.

Based on our GMRF model, the prior joint distribution of $\mathbb{Y}$ is $\mathrm{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$. We adopt non-informative 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 $\bar{\mathbf{Y}}$ 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 $\bar{\mathbf{Y}}$ as a realization of the GMRF $\mathbb{Y}^{\boldsymbol{\epsilon}} = \mathbb{Y} + \boldsymbol{\epsilon}$, 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 $\mathbb{Y}^{\boldsymbol{\epsilon}}$ is $\mathrm{N}(\boldsymbol{\mu}, (\mathbf{Q} + \mathbf{Q}_{\boldsymbol{\epsilon}})^{-1})$. We choose to simulate all solutions independently (no CRN), which makes $\mathbf{Q}_{\boldsymbol{\epsilon}}$ a diagonal matrix so that the sparsity pattern of $\mathbf{Q}$ is preserved for $\mathbf{Q} + \mathbf{Q}_{\boldsymbol{\epsilon}}$. If solution $\mathbf{x}$ has been simulated, the corresponding diagonal element of $\mathbf{Q}_{\boldsymbol{\epsilon}}$ 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 $\mathbf{Q}_{\boldsymbol{\epsilon}}$ is set to 0.

? prove that the conditional distribution of $\mathbb{Y} | \mathbb{Y}^{\boldsymbol{\epsilon}} = \bar{\mathbf{Y}}$ is

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

where $\bar{\mathbf{Q}} = \mathbf{Q} + \mathbf{Q}_{\boldsymbol{\epsilon}}$ 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 (??) 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, $\mathbf{Q}_\epsilon$, 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 **?** and our rGMIA guide their search and (possibly) termination using *complete expected improvement* (CEI), which is defined in **?**. At any iteration, the estimated optimal solution is $\tilde{\mathbf{x}} = \arg\min_{\{\mathbf{x}\in\mathcal{X}:r(\mathbf{x})>0\}} \bar{Y}(\mathbf{x})$, where $\bar{Y}(\mathbf{x})$ is the component of $\bar{\mathbf{Y}}$ 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 to $\tilde{\mathbf{x}}$, 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 $\text{CEI}(\tilde{\mathbf{x}}, \mathbf{x}) = \mathbb{E}\left[\max\left(\mathbb{Y}(\tilde{\mathbf{x}}) - \mathbb{Y}(\mathbf{x}), 0\right) | \mathbb{Y}^{\boldsymbol{\epsilon}} = \bar{\mathbf{Y}}\right]$, where the expectation is conditional on $\mathbb{Y}^{\boldsymbol{\epsilon}} = \bar{\mathbf{Y}}$, the simulation output that has been collected. CEI is an extension of the EI acquisition function (**?**) tailored for stochastic simulation (**?**). 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 (**??**) 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 $V(\tilde{\mathbf{x}}, \mathbf{x}) \equiv V(\tilde{\mathbf{x}}) + V(\mathbf{x}) - 2C(\tilde{\mathbf{x}}, \mathbf{x})$.

? show that the CEI of $\mathbf{x}$ can be expressed as

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

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, **?** prove that under simplified conditions (R&S with independent and normally distributed simulation output with known variances), CEI satisfies the conditions found in **?** 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 **?**, argue that CEI is a good acquisition function for inference-based optimization.

Let $\mathbf{M}(\mathbf{x}_{\mathcal{X}}) = [M(\mathbf{x}_1), M(\mathbf{x}_2), \ldots, M(\mathbf{x}_n)]^\top$, $\mathbf{V}(\mathbf{x}_{\mathcal{X}}) = [V(\mathbf{x}_1), V(\mathbf{x}_2), \ldots, V(\mathbf{x}_n)]^\top$, and $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{X}}) = [C(\tilde{\mathbf{x}}, \mathbf{x}_1), C(\tilde{\mathbf{x}}, \mathbf{x}_2), \ldots, C(\tilde{\mathbf{x}}, \mathbf{x}_n)]^\top$. 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.

? introduced a multi-resolution framework in which the feasible solution space is divided into non-overlapping 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 multi-resolution 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.

? propose an efficient way to compute the diagonal elements of $\bar{\mathbf{Q}}^{-1}$ without full inversion when $\bar{\mathbf{Q}}$ is sparse. PARDISO (**?**), a linear solver specialized for parallel computation using state-of-the-art algorithms, was employed to perform this calculation. However, the **?** 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 **??**, 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 (**??**) given the simulation outputs from the initial design and computes the CEIs of all solutions in $\mathcal{X}$. While the stopping criterion is not satisfied, GMIA simulates the current sample-best solution, $\tilde{\mathbf{x}}$, and the solution with the largest CEI, $\mathbf{x}^{\text{CEI}}$, 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 $\mathbf{x}^{\text{CEI}}$, respectively.

There are two stopping paradigms in OvS: fixed-precision and fixed-budget (**?**). 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 **??**, 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 a R&S procedure the computational budget is specified as the allowable number of simulation replications, since 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 non-simulation time.

Algorithm 1: GMIA

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 $\bar{\mathbf{Q}} = \mathbf{Q} + \mathbf{Q}_{\boldsymbol{\epsilon}}$ and $\bar{\mathbf{Y}}$;
while *Stopping criterion not reached* **do**
  Find $\tilde{\mathbf{x}} = \arg\min_{\{\mathbf{x}\in\mathcal{X}:r(\mathbf{x})>0\}} \bar{Y}(\mathbf{x})$;
  Compute Cholesky factor of $\bar{\mathbf{Q}}$: $\mathbf{L}_{\bar{\mathbf{Q}}}$;
  Compute $\mathbf{V}(\mathbf{x}_{\mathcal{X}}) = \text{diag}(\bar{\mathbf{Q}}^{-1})$, using $\mathbf{L}_{\bar{\mathbf{Q}}}$;
  Compute $\mathbf{C}(\tilde{\mathbf{x}},\mathbf{x}_{\mathcal{X}}) = \bar{\mathbf{Q}}^{-1}\mathbf{e}_{\tilde{\mathbf{x}}}$, using $\mathbf{L}_{\bar{\mathbf{Q}}}$;
  Compute $\mathbf{M}(\mathbf{x}_{\mathcal{X}}) = \boldsymbol{\mu} + \bar{\mathbf{Q}}^{-1}\mathbf{Q}_{\boldsymbol{\epsilon}}(\bar{\mathbf{Y}} - \boldsymbol{\mu})$, using $\mathbf{L}_{\bar{\mathbf{Q}}}$;
  Calculate $\text{CEI}(\tilde{\mathbf{x}},\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$;
  Find $\mathbf{x}^{\text{CEI}} = \arg\max_{\mathbf{x}\in\mathcal{X}\setminus\tilde{\mathbf{x}}} \text{CEI}(\tilde{\mathbf{x}},\mathbf{x})$;
  Simulate at $\tilde{\mathbf{x}}$ and $\mathbf{x}^{\text{CEI}}$. Update $\bar{\mathbf{Y}}$, $\mathbf{Q}_{\epsilon}$, and $\bar{\mathbf{Q}}$ by incorporating the new simulation outputs;
end
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 **??**, Steps 4 and 5 are the most expensive in terms of non-simulation overhead. As mentioned in the previous section, **?** 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. Due to 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 the Online Supplement (Appendix **??**), 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 $\mathbf{x}^{\text{CEI}}$. 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. Since 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 **??**. 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 **??** illustrates the steps of rGMIA including the necessary computation required at each step. We defer discussion of the derivation of these results to Section **??** 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 **??** 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, $\max_{\mathbf{x}\in\mathcal{X}\setminus\tilde{\mathbf{x}}} \mathrm{CEI}(\tilde{\mathbf{x}},\mathbf{x}) \leq \delta$, is used. At each global-search iteration (Steps **??**–**??**), 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, $\tilde{\mathbf{x}} = \arg\min_{\{\mathbf{x}\in\mathcal{X}:r(\mathbf{x})>0\}} \bar{Y}(\mathbf{x})$, 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 **??**–**??**), 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 **??**. 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., $\tilde{\mathbf{x}} = \arg\min_{\{\mathbf{x}\in\mathcal{S}:r(\mathbf{x})>0\}} \bar{Y}(\mathbf{x})$. 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 **?** 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.

Algorithm 2: rGMIA

Choose $n_0 \ll n$ initial solutions. Simulate at each solution and compute MLEs for the GMRF parameters $(\boldsymbol{\mu}, \boldsymbol{\theta})$. Construct $\bar{\mathbf{Q}} = \mathbf{Q} + \mathbf{Q}_\epsilon$;
Find $\tilde{\mathbf{x}} = \arg\min_{\{\mathbf{x} \in \mathcal{X}: r(\mathbf{x}) > 0\}} \bar{Y}(\mathbf{x})$;
Compute Cholesky factor of $\bar{\mathbf{Q}}$: $\mathbf{L}_{\bar{\mathbf{Q}}}$;
Compute $\mathbf{V}(\mathbf{x}_\mathcal{X}) = \text{diag}(\bar{\mathbf{Q}}^{-1})$, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{X}) = \bar{\mathbf{Q}}^{-1}\mathbf{e}_{\tilde{\mathbf{x}}}$, $\mathbf{M}(\mathbf{x}_\mathcal{X}) = \boldsymbol{\mu} + \bar{\mathbf{Q}}^{-1}\mathbf{Q}_\epsilon(\bar{\mathbf{Y}} - \boldsymbol{\mu})$, using $\mathbf{L}_{\bar{\mathbf{Q}}}$. Go to Step **??**;
while *global-search termination criterion not reached* **do**
  while *rapid-search termination criterion not reached* **do**
    Simulate at $\tilde{\mathbf{x}}$, $\mathbf{x}^{\text{CEI}}$. Update simulation information by updating $\bar{Y}(\tilde{\mathbf{x}})$, $\bar{Y}(\mathbf{x}^{\text{CEI}})$, $\mathbf{Q}_\epsilon$, $\bar{\mathbf{Q}}$, $\bar{\mathbf{Q}}_{\mathcal{SS}}$;
    Find $\tilde{\mathbf{x}} = \arg\min_{\{\mathbf{x} \in \mathcal{S}: r(\mathbf{x}) > 0\}} \bar{Y}(\mathbf{x})$;
    Compute $\mathbf{V}(\mathbf{x}_\mathcal{S})$, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{S})$ by computing $\boldsymbol{\Sigma}_{\mathcal{SS}} = (\bar{\mathbf{Q}}_{\mathcal{SS}} - \mathbf{B})^{-1}$;
    Compute $\mathbf{M}(\mathbf{x}_\mathcal{S}) = \boldsymbol{\mu}_\mathcal{S} + \boldsymbol{\Sigma}_{\mathcal{SS}}([\mathbf{Q}_\epsilon]_{\mathcal{SS}}(\bar{\mathbf{Y}}(\mathbf{x}_\mathcal{S}) - \boldsymbol{\mu}_\mathcal{S}) - \mathbf{a})$;
    Calculate $\text{CEI}(\tilde{\mathbf{x}}, \mathbf{x}), \forall \mathbf{x} \in \mathcal{S}$;
    Find $\mathbf{x}^{\text{CEI}} = \arg\max_{\mathbf{x} \in \mathcal{S} \setminus \tilde{\mathbf{x}}} \text{CEI}(\tilde{\mathbf{x}}, \mathbf{x})$;
  end
  Simulate at $\tilde{\mathbf{x}}$, $\mathbf{x}^{\text{CEI}}$. Update simulation information by updating $\bar{Y}(\tilde{\mathbf{x}})$, $\bar{Y}(\mathbf{x}^{\text{CEI}})$, $\mathbf{Q}_\epsilon$, $\bar{\mathbf{Q}}$, $\bar{\mathbf{Q}}_{\mathcal{SS}}$;
  Find $\tilde{\mathbf{x}} = \arg\min_{\{\mathbf{x} \in \mathcal{X}: r(\mathbf{x}) > 0\}} \bar{Y}(\mathbf{x})$;
  Compute $\mathbf{V}(\mathbf{x}_\mathcal{S})$ from $\boldsymbol{\Sigma}_{\mathcal{SS}} = (\bar{\mathbf{Q}}_{\mathcal{SS}} - \mathbf{B})^{-1}$;
  Compute $\mathbf{M}(\mathbf{x}_\mathcal{S}) = \boldsymbol{\mu}_\mathcal{S} + \boldsymbol{\Sigma}_{\mathcal{SS}}([\mathbf{Q}_\epsilon]_{\mathcal{SS}}(\bar{\mathbf{Y}}(\mathbf{x}_\mathcal{S}) - \boldsymbol{\mu}_\mathcal{S}) - \mathbf{a})$;
  Compute $\mathbf{V}(\mathbf{x}_\mathcal{F}) = \text{diag}(\bar{\mathbf{Q}}^{-1}_{\mathcal{FF}}) + \text{diag}(\mathbf{A}\boldsymbol{\Sigma}_{\mathcal{SS}}\mathbf{A}^\top)$, using $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{FF}}}$;
  Compute $\mathbf{M}(\mathbf{x}_\mathcal{F}) = \boldsymbol{\mu}_\mathcal{F} + \bar{\mathbf{Q}}^{-1}_{\mathcal{FF}}[\mathbf{Q}_\epsilon]_{\mathcal{FF}}(\bar{\mathbf{Y}}(\mathbf{x}_\mathcal{F}) - \boldsymbol{\mu}_\mathcal{F}) - \mathbf{A}(\mathbf{M}(\mathbf{x}_\mathcal{S}) - \boldsymbol{\mu}_\mathcal{S})$, using $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{FF}}}$;
  if $\tilde{\mathbf{x}} \in \mathcal{S}$ **then**
    Compute $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{S}) = [\boldsymbol{\Sigma}_{\mathcal{SS}}]_{\cdot\tilde{\mathbf{x}}}$;
    Compute $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{F}) = -\mathbf{A}[\boldsymbol{\Sigma}_{\mathcal{SS}}]_{\cdot\tilde{\mathbf{x}}}$;
  else
    Compute $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{S}) = -\boldsymbol{\Sigma}_{\mathcal{SS}}[\mathbf{A}^\top]_{\cdot\tilde{\mathbf{x}}}$;
    Compute $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{F}) = \bar{\mathbf{Q}}^{-1}_{\mathcal{FF}}\mathbf{e}_{\tilde{\mathbf{x}}} + \mathbf{A}\boldsymbol{\Sigma}_{\mathcal{SS}}[\mathbf{A}^\top]_{\cdot\tilde{\mathbf{x}}}$, using $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{FF}}}$;
  end
  Calculate $\text{CEI}(\tilde{\mathbf{x}}, \mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$;
  Find $\mathbf{x}^{\text{CEI}} = \arg\max_{\mathbf{x} \in \mathcal{X} \setminus \tilde{\mathbf{x}}} \text{CEI}(\tilde{\mathbf{x}}, \mathbf{x})$;
  Construct $\{\mathcal{F}, \mathcal{S}\}$ partition of $\bar{\mathbf{Q}}$ into $\bar{\mathbf{Q}}_{\mathcal{FF}}$, $\bar{\mathbf{Q}}_{\mathcal{FS}}$, $\bar{\mathbf{Q}}_{\mathcal{SS}}$;
  Compute Cholesky factor of $\bar{\mathbf{Q}}_{\mathcal{FF}}$: $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{FF}}}$;
  Compute $\mathbf{A} = \bar{\mathbf{Q}}^{-1}_{\mathcal{FF}}\bar{\mathbf{Q}}_{\mathcal{FS}}$, using $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{FF}}}$, $\mathbf{B} = \bar{\mathbf{Q}}^\top_{\mathcal{FS}}\mathbf{A}$, $\mathbf{a} = \mathbf{A}^\top([\mathbf{Q}_\epsilon]_{\mathcal{FF}}(\bar{\mathbf{Y}}(\mathbf{x}_\mathcal{F}) - \boldsymbol{\mu}_\mathcal{F}))$;
end
Return $\tilde{\mathbf{x}}$ as the estimated optimal solution;

Initialization Rapid search

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 **??** and **??**, respectively. Section **??** then compares rGMIA to GMIA and proves global convergence.

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

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

we obtain the following expression for $\boldsymbol{\Sigma}$ via standard block-matrix inversion:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathcal{F}\mathcal{F}} & \boldsymbol{\Sigma}_{\mathcal{F}\mathcal{S}} \\ \boldsymbol{\Sigma}_{\mathcal{F}\mathcal{S}}^\top & \boldsymbol{\Sigma}_{\mathcal{S}\mathcal{S}} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} + \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1}\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}\boldsymbol{\Sigma}_{\mathcal{S}\mathcal{S}}\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}^\top\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} & -\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1}\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}\boldsymbol{\Sigma}_{\mathcal{S}\mathcal{S}} \\ -\boldsymbol{\Sigma}_{\mathcal{S}\mathcal{S}}\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}^\top\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} & \boldsymbol{\Sigma}_{\mathcal{S}\mathcal{S}} \end{bmatrix}, \tag{3}$$

where $\boldsymbol{\Sigma}_{\mathcal{S}\mathcal{S}} = (\bar{\mathbf{Q}}_{\mathcal{S}\mathcal{S}} - \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}^\top\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1}\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}})^{-1}$ is the covariance matrix of the search set. Our focus is on $\boldsymbol{\Sigma}_{\mathcal{S}\mathcal{S}}$ 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 $\mathbf{V}(\mathbf{x}_\mathcal{S})$ and $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{S})$ during rapid-search iterations without updating $\mathbf{V}(\mathbf{x}_\mathcal{F})$ and $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{F})$. Since only solutions in $\mathcal{S}$ are simulated, $\mathbf{B} = \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}^\top\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1}\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}$ 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 $\mathbf{A} = \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1}\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}$ since 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 $\mathbf{M}(\mathbf{x}_\mathcal{X})$:

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

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 $\mathbf{Q}_\epsilon$ 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,

$$\mathbf{M}(\mathbf{x}_\mathcal{S}) = \boldsymbol{\mu}_\mathcal{S} + \boldsymbol{\Sigma}_{\mathcal{S}\mathcal{S}}\left([\mathbf{Q}_\epsilon]_{\mathcal{S}\mathcal{S}}(\bar{\mathbf{Y}}(\mathbf{x}_\mathcal{S}) - \boldsymbol{\mu}_\mathcal{S}) - \mathbf{A}^\top[\mathbf{Q}_\epsilon]_{\mathcal{F}\mathcal{F}}(\bar{\mathbf{Y}}(\mathbf{x}_\mathcal{F}) - \boldsymbol{\mu}_\mathcal{F})\right). \tag{5}$$

During the rapid search, only $[\mathbf{Q}_\epsilon]_{\mathcal{S}\mathcal{S}}$ and $\bar{\mathbf{Y}}(\mathbf{x}_\mathcal{S})$ change, while $\mathbf{a} = \mathbf{A}^\top[\mathbf{Q}_\epsilon]_{\mathcal{F}\mathcal{F}}(\bar{\mathbf{Y}}(\mathbf{x}_\mathcal{F}) - \boldsymbol{\mu}_\mathcal{F})$ 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 $\mathbf{\Sigma}_{\mathcal{SS}}$. From (**??**), $\mathbf{\Sigma}_{\mathcal{SS}} = (\bar{\mathbf{Q}}_{\mathcal{SS}} - \mathbf{B})^{-1}$, which is performed in Step 9 of Algorithm **??**. By construction, $\bar{\mathbf{Q}}_{\mathcal{SS}}$ is a sparse matrix, but $\mathbf{B}$ may be dense. Hence, the floating point operation (flop) count of computing $\mathbf{\Sigma}_{\mathcal{SS}}$ is $\mathcal{O}(n_{\mathcal{S}}^3)$. Following directly from (**??**), Step 10 computes $\mathbf{M}(\mathbf{x}_{\mathcal{S}})$ by multiplying the dense $n_{\mathcal{S}} \times n_{\mathcal{S}}$ matrix $\mathbf{\Sigma}_{\mathcal{SS}}$ by a vector, which costs $\mathcal{O}(n_{\mathcal{S}}^2)$. Thus, the overall cost of a single rapid-search iteration is $\mathcal{O}(n_{\mathcal{S}}^3)$. Compared to 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 to 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 **??**. 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 $\max_{\mathbf{x} \in \mathcal{S} \setminus \tilde{\mathbf{x}}} \mathrm{CEI}(\tilde{\mathbf{x}}, \mathbf{x}) < \gamma$, 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 3–4 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 $\mathbf{V}(\mathbf{x}_{\mathcal{S}})$ and $\mathbf{M}(\mathbf{x}_{\mathcal{S}})$ 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 (**??**),

$$\boldsymbol{\Sigma}_{\mathcal{F}\mathcal{F}} = \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} + \mathbf{A}(\bar{\mathbf{Q}}_{\mathcal{S}\mathcal{S}} - \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{S}}^{\top}\mathbf{A})^{-1}\mathbf{A}^{\top} = \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1} + \mathbf{A}\boldsymbol{\Sigma}_{\mathcal{S}\mathcal{S}}\mathbf{A}^{\top}. \tag{6}$$

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

$$\mathbf{M}(\mathbf{x}_{\mathcal{F}}) = \boldsymbol{\mu}_{\mathcal{F}} + \bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}^{-1}[\mathbf{Q}_{\epsilon}]_{\mathcal{F}\mathcal{F}}(\bar{\mathbf{Y}}(\mathbf{x}_{\mathcal{F}}) - \boldsymbol{\mu}_{\mathcal{F}}) - \mathbf{A}(\mathbf{M}(\mathbf{x}_{\mathcal{S}}) - \boldsymbol{\mu}_{\mathcal{S}}). \tag{7}$$

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.

Since $\tilde{\mathbf{x}}$ 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, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{S}})$ and $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{F}})$, are computed. When $\tilde{\mathbf{x}} \in \mathcal{S}$, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{S}})$ is simply $[\boldsymbol{\Sigma}_{\mathcal{SS}}]_{\cdot\tilde{\mathbf{x}}}$, the column of $\boldsymbol{\Sigma}_{\mathcal{SS}}$ corresponding to $\tilde{\mathbf{x}}$. Also, from (**??**)

$$\boldsymbol{\Sigma}_{\mathcal{FS}} = -\bar{\mathbf{Q}}_{\mathcal{FF}}^{-1}\bar{\mathbf{Q}}_{\mathcal{FS}}(\bar{\mathbf{Q}}_{\mathcal{SS}} - \bar{\mathbf{Q}}_{\mathcal{FS}}^{\top}\mathbf{A})^{-1} = -\bar{\mathbf{Q}}_{\mathcal{FF}}^{-1}\bar{\mathbf{Q}}_{\mathcal{FS}}\boldsymbol{\Sigma}_{\mathcal{SS}} = -\mathbf{A}\boldsymbol{\Sigma}_{\mathcal{SS}}.$$

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

When $\tilde{\mathbf{x}} \in \mathcal{F}$, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{S}})$ is a column of $\boldsymbol{\Sigma}_{\mathcal{SF}}$ corresponding to $\tilde{\mathbf{x}}$. Since $\boldsymbol{\Sigma}_{\mathcal{SF}} = -\boldsymbol{\Sigma}_{\mathcal{SS}}\mathbf{A}^{\top}$, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{S}}) = -\boldsymbol{\Sigma}_{\mathcal{SS}}[\mathbf{A}^{\top}]_{\cdot\tilde{\mathbf{x}}}$. Similarly, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{F}})$ is a column of $\boldsymbol{\Sigma}_{\mathcal{FF}}$ corresponding to $\tilde{\mathbf{x}}$. From (**??**), $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_{\mathcal{F}}) = \bar{\mathbf{Q}}_{\mathcal{FF}}^{-1}\mathbf{e}_{\tilde{\mathbf{x}}} + \mathbf{A}\boldsymbol{\Sigma}_{\mathcal{SS}}[\mathbf{A}^{\top}]_{\cdot\tilde{\mathbf{x}}}$. Again, $\bar{\mathbf{Q}}_{\mathcal{FF}}^{-1}\mathbf{e}_{\tilde{\mathbf{x}}}$ can be computed efficiently by solving $\bar{\mathbf{Q}}_{\mathcal{FF}}\mathbf{z} = \mathbf{e}_{\tilde{\mathbf{x}}}$ 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 $\bar{\mathbf{Q}}_{\mathcal{FF}}$, performing a selected inverse to compute $\operatorname{diag}\left(\bar{\mathbf{Q}}_{\mathcal{FF}}^{-1}\right)$ and solving a linear system of equations with $\bar{\mathbf{Q}}_{\mathcal{FF}}$. We use the PARDISO software (**?**) to perform these calculations, which improves their efficiency by pre-processing large matrices such as $\bar{\mathbf{Q}}_{\mathcal{FF}}$. 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 the Online Supplement (Appendix **??**) for the results.

Despite the lack of explicit flop counts for PARDISO calculations, we can still characterize the computational savings attained by rGMIA compared to GMIA by parameterizing the flop counts for computing $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{FF}}}$, the Cholesky factor of $\bar{\mathbf{Q}}_{\mathcal{FF}}$, for performing a selected inverse to obtain $\operatorname{diag}(\bar{\mathbf{Q}}_{\mathcal{FF}}^{-1})$ given $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{FF}}}$, and for solving a single-column right-hand-side linear system involving $\bar{\mathbf{Q}}_{\mathcal{FF}}$ given $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{FF}}}$; 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 $\bar{\mathbf{Q}}_{\mathcal{FF}}$ and thus determines the cost of performing these matrix operations.

As previously characterized for rapid-search iterations, computing $\mathbf{V}(\mathbf{x}_{\mathcal{S}})$ and $\mathbf{M}(\mathbf{x}_{\mathcal{S}})$ costs $\mathcal{O}(n_{\mathcal{S}}^3)$ flops. To compute $\mathbf{V}(\mathbf{x}_{\mathcal{F}})$, it costs C_I for $\operatorname{diag}\left(\bar{\mathbf{Q}}_{\mathcal{FF}}^{-1}\right)$ and $\mathcal{O}(n_{\mathcal{S}}^3)$ flops for $\operatorname{diag}\left(\mathbf{A}\boldsymbol{\Sigma}_{\mathcal{SS}}^{-1}\mathbf{A}^{\top}\right)$. For $\mathbf{M}(\mathbf{x}_{\mathcal{F}})$, it costs C_L for solving a system of linear equations and

$\mathcal{O}(n_{\mathcal{S}}n)$ for the matrix-vector multiplication in (**??**). 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 $C_L + \mathcal{O}(n_{\mathcal{S}}n)$ flops. In our numerical experiments we observed that $\tilde{x}$ tends to remain in $\mathcal{S}$ in later iterations. Finally, computing the intermediate matrices requires $C_F + n_{\mathcal{S}}C_L + \mathcal{O}(n_{\mathcal{S}}^2 n)$.

To summarize, a single global-search iteration incurs a cost of $C_F + C_I + (n_{\mathcal{S}} + 2)C_L + \mathcal{O}(n_{\mathcal{S}}^2 n)$ flops. See the Online Supplement (Appendix **??**) 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 $\mathbf{M}(\mathbf{x}_{\mathcal{X}})$. 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 **??**. 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 $\mathcal{O}\left(n_{\mathcal{S}}^3 + (C_F + C_I + n_{\mathcal{S}}C_L + n_{\mathcal{S}}^2 n)/p\right)$. Recall that $n_{\mathcal{S}}$ 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 the Online Supplement, Appendix **??**, 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 vs. 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 $n_{\mathcal{S}} = 100$. 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 $n_{\mathcal{S}} \times n_{\mathcal{S}}$ matrix. Using MATLAB, with $n_{\mathcal{S}} = 100$, 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 the Online Supplement (Appendix **??**).

? 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 **??**—rGMIA still achieves global convergence for either the fixed-p or adaptive schemes; see the Online Supplement, Appendix **??**, for the proofs. To begin, we introduce the following lemma:

LEMMA 1. *At any iteration of GMIA or global-search iteration of rGMIA, 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 $\gamma = \max_{\mathbf{x} \in \mathcal{F}} \mathrm{CEI}_t(\tilde{\mathbf{x}}, \mathbf{x})$ will be positive with probability 1 after any finite number of iterations of rGMIA. With the aid of Lemma **??**, we establish global convergence of rGMIA using only the assumptions presented in **?** to prove convergence of GMIA as stated below.

THEOREM 1. *Assume: (i)* $y(\mathbf{x}) > -\infty, \forall \mathbf{x} \in \mathcal{X}$*, (ii)* $0 < Var[Y(\mathbf{x})] < \infty, \forall \mathbf{x} \in \mathcal{X}$ *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 **?**, which has characteristics of a practical DOvS problem and has already been used to test the behavior of GMRF-based optimization algorithms in **?**. 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 $s \times (S-s) = [1,2,\ldots,100] \times [1,2,\ldots,100]$, and **inventory_150** covering solutions $s \times (S-s) = [1,2,\ldots,150] \times [1,2,\ldots,150]$. 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 **?** for a description. The Griewank function is a popular test problem due to 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 (colored in blue) with the global minimum at $(0,0)$. The range of the function is $[0, 2.5490]$. The 4 local minima have response values of 0.6828, compared to 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 (**?**): Suppose a restaurant has the objective of maximizing profit (or minimizing negative profit). There are d different sizes of tables, $s_i, i = 1,2,\ldots,d$, 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 \$1000s, per person. Groups that find no available table upon arrival leave without waiting. Keeping a size-s_i table costs $c_{s_i} \times$ \$1000/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 **??** in the Online Supplement (Appendix **??**) 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 256GB of RAM, and a head node that has 20 cores and 256GB 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 **?**. 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 1 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 $n_{\mathcal{S}} = 50$ with $p = 10, 25, 50, 100, 200$ rapid-search iterations per global-search iteration, and the adaptive scheme. Results in Tables **??**–**??** indicate that $p = 50$ performs especially well. For (favorable) comparisons of the GMIA approach with other Bayesian optimization algorithms see **?**. The focus of this paper is providing a computationally superior way to achieve the same search progress and inference.

Table 1 Fixed-precision results averaged from 30 macro-replications of rGMIA and GMIA applied to the inventory_100 and inventory_150 problems. Standard errors of mean values are provided in parentheses.

Problem	inventory_100																				
δ	0.1							0.05							0.01						
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	$p=10$	$p=25$	$p=50$	$p=100$	$p=200$	Adaptive	GMIA
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 (4)	1802 (47)
Max 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)
Max 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)
Problem	**inventory_150**																				
δ	0.1							0.05							0.01						
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	$p=10$	$p=25$	$p=50$	$p=100$	$p=200$	Adaptive	GMIA
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)
Max 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)
Max 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)

Reported results for **inventory_150** is averaged across 29 macro-replication with outlier macro-replication removed. This is further explained in Section **??**.

Table 2 Fixed-budget results averaged from 30 macro-replications of rGMIA ($|\mathcal{S}|=50$) and GMIA applied to the griewank_101, griewank_201, griewank_301 and griewank_401 problems, given a 1 hour time budget. Standard errors of mean values are provided in parentheses.

Problem	griewank_101							griewank_201						
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
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)
Max 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**							**griewank_401**						
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
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)
Max 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)

Table **??** contains the results of fixed-precision GMIA and rGMIA applied to the inventory problem. In each subtable, 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 to other test problems. However, even in this setting with relatively cheap computational overhead, Table **??** 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** vs. **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 (10068.06s). 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 **??** 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 after the 1 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 **??** 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 **??** in the Online Supplement, Appendix **??**, 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 **??** 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 to 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 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 2 days across 30 macro-replications for $\delta = 0.1$, GMIA effectively was unable to terminate. rGMIA was able to return an estimated optimal solution within $\delta = 0.1$ in 2 hours on average.

Table 3 Fixed-precision results averaged from 30 macro-replications of rGMIA ($|\mathcal{S}| = 50$) and GMIA applied to the restaurant_125, restaurant_25 and restaurant_5 problems. Standard errors of mean values are provided in parentheses.

Problem	**restaurant_125**																				
δ	0.1							0.05							0.01						
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	$p=10$	$p=25$	$p=50$	$p=100$	$p=200$	Adaptive	GMIA
Mean Time until Termination (s)	4596 (752)	4396 (716)	4309 (700)	5268 (855)	6403 (1042)	4338 (710)	3816 (629)	5047 (822)	4849 (787)	4672 (757)	5620 (910)	7004 (1136)	5603 (809)	4221 (686)	6594 (939)	6275 (894)	6304 (828)	7077 (926)	8899 (1168)	7471 (756)	5808 (825)
Max Time until Termination (s)	9044	8518	8387	9934	12458	8670	7582	9612	8986	8940	10344	13331	13438	8134	11360	10809	10398	11041	14267	13438	10353
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)	0.09 (0.02)	0.08 (0.02)	0.07 (0.02)	0.11 (0.02)	0.11 (0.02)	0.08 (0.01)	0.07 (0.01)	0.07 (0.01)	0.06 (0.01)	0.06 (0.01)	0.08 (0.01)	0.08 (0.01)
Max 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	0.44	0.31	0.31	0.31	0.31	0.31	0.31	0.31
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)	8853 (1255)	9050 (1280)	10113 (1332)	16304 (2133)	30641 (4020)	13965 (2434)	8808 (1247)
Problem	**restaurant_25**																				
δ	0.1							0.05							0.01						
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	$p=10$	$p=25$	$p=50$	$p=100$	$p=200$	Adaptive	GMIA
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)	8557 (728)	5594 (424)	3298 (215)	4415 (221)	7208 (363)	5242 (374)	27505 (2389)
Max Time until Termination (s)	5831	3904	2504	3633	6285	4315	26093	6542	4338	2788	3967	6929	4723	28569	11733	7339	4246	5192	8301	12735	44670
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)	0.06 (0.01)	0.04 (0.01)	0.05 (0.01)	0.05 (0.01)	0.04 (0.00)	0.07 (0.01)	0.06 (0.01)
Max 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	0.12	0.15	0.12	0.09	0.18	0.13
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)	16632 (1393)	18107 (1339)	19933 (1262)	28981 (1475)	50161 (2517)	19356 (3813)	15985 (1340)
Problem	**restaurant_5**																				
δ	0.1							0.05							0.01						
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	$p=10$	$p=25$	$p=50$	$p=100$	$p=200$	Adaptive	GMIA
Mean Time until Termination (s)	43699 (7595)	22664 (3952)	7221 (1256)	10212 (1760)	11717 (2015)	20095 (3836)	>48hr	59750 (8444)	33392 (4471)	11487 (1406)	15075 (1966)	16890 (2047)	29970 (4282)	>48hr	123798 (7759)	78670 (4348)	29537 (1608)	28435 (1026)	27360 (541)	56981 (3936)	>48hr
Max Time until Termination (s)	92035	49310	15737	21135	23450	55710	>48hr	105517	57054	18369	24979	26599	62941	>48hr	152494	98836	37269	32685	30624	81357	>48hr
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)	0.05 (0.01)	0.04 (0.00)	0.03 (0.00)	0.03 (0.00)	0.06 (0.01)	N/A
Max 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	0.15	0.11	0.08	0.07	0.19	N/A
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)	N/A

Many macro-replications of running GMIA on **restaurant_5** had long run times such that the mean time elapsed until termination was >48hr. For this reason, full results were omitted, but this illustrates rGMIA's ability to solve problems unable to be solved by GMIA.

7.2. rGMIA's Performance Sensitivity to $n_{\mathcal{S}}$ 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 $n_{\mathcal{S}} = 50$, and $p = 50$ rapid-search iterations showed good performance across all problems. We now vary the search set size as $n_{\mathcal{S}} = 50, 100, 200$ 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 the Online Supplement (Tables **??**–**??** of Appendix **??**), and summarize our findings here.

Tables **??**–**??** show that for a given search set size $n_{\mathcal{S}}$, $p = n_{\mathcal{S}}$ is the best choice. We confirmed that in many cases when $p = n_{\mathcal{S}}$ 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 $p < n_{\mathcal{S}}$ 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 $p > n_{\mathcal{S}}$, 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 $p = n_{\mathcal{S}}$.

Nonetheless, we speculate the adaptive scheme may be useful when δ is small. For example, we can observe in Table **??** that for smaller δ, the relative performance difference between the adaptive scheme and $p = n_{\mathcal{S}}$ 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 $n_{\mathcal{S}}$ appears problem specific. Nevertheless, the run times indicate the performance is not sensitive to the choice of $n_{\mathcal{S}}$. This suggests that there is little penalty in choosing a suboptimal $n_{\mathcal{S}}$, given that $p = n_{\mathcal{S}}$.

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. While 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 (**?**) is the current state-of-the-art in inferential optimization for DOvS. The focus of **?** 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, $\tilde{\mathbf{x}}$ and $\mathbf{x}^{\text{CEI}}$, 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 while 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 $\tilde{\mathbf{x}}$ and the solutions with the $n_{\mathcal{S}} - 1$ largest CEI values. While 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*: $\bar{\mathbf{Q}}$ becomes less sparse with dimension d. **?** consider projecting less-active dimensions onto active dimensions, and while this seems promising, creative ideas for addressing large d are clearly needed.

Acknowledgments

This research was supported by National Science Foundation Grant Number DMS-1854562. The authors thank the Area Editor, Associate Editor and two referees for their valuable and timely feedback.

Online Supplement

Appendix A: rGMIA Convergence Proof

In this appendix we provide the proofs of Lemma 1 and Theorem 1. We do not introduce any additional assumptions beyond those already presented in **?**, which we restate here:

1. $y(\mathbf{x}) > -\infty, \forall \mathbf{x} \in \mathcal{X}$.
2. $0 < \text{Var}[Y(\mathbf{x})] < \infty, \forall \mathbf{x} \in \mathcal{X}$.
3. The initially estimated $\mathbf{Q} = \mathbf{Q}(\widehat{\boldsymbol{\theta}})$ is positive definite and not updated, where $\widehat{\boldsymbol{\theta}}$ are parameter estimates.

As in **?** we have implicitly assumed that $Y(\mathbf{x})$ is continuous-valued, in which case the assumptions above imply that $\bar{\mathbf{Q}}$ is always positive definite: $\bar{\mathbf{Q}} = \mathbf{Q} + \mathbf{Q}_\epsilon$, $\mathbf{Q}$ is positive definite by Assumption 3, and $\mathbf{Q}_\epsilon$ is a diagonal matrix with finite, non-negative elements on its diagonal with probability 1. However, it is possible that $Y(\mathbf{x})$ is discrete-valued, in which case there is a positive probability that $S^2(\mathbf{x}) = 0$ for some $\mathbf{x} \in \mathcal{X}$ during some iterations. Therefore, when the output $Y(\mathbf{x})$ is discrete-valued, we set the diagonal elements of $\mathbf{Q}_\epsilon$ to $r(\mathbf{x})/\max\{S^2(\mathbf{x}), \eta\}$, for some very small $\eta > 0$, whenever $r(\mathbf{x}) > 0$.

An immediate consequence is that $V(\tilde{\mathbf{x}}, \mathbf{x}) > 0$ and finite for all $\mathbf{x} \in \mathcal{X}$. Furthermore, Assumptions 1 and 2 imply that $\infty < \bar{Y}(\mathbf{x}) < \infty, \forall \mathbf{x} \in \mathcal{X}$ with probability 1, so the conditional means are also finite, $-\infty < M(\mathbf{x}) < \infty, \forall \mathbf{x} \in \mathcal{X}$, with probability 1. We use these insights in the proof that follows.

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

Proof Recall from Equation (**??**),

$$\text{CEI}(\tilde{\mathbf{x}}, \mathbf{x}) = (M(\tilde{\mathbf{x}}) - M(\mathbf{x}))\, \Phi\left(\frac{M(\tilde{\mathbf{x}}) - M(\mathbf{x})}{\sqrt{V(\tilde{\mathbf{x}}, \mathbf{x})}}\right) + \sqrt{V(\tilde{\mathbf{x}}, \mathbf{x})}\; \phi\left(\frac{M(\tilde{\mathbf{x}}) - M(\mathbf{x})}{\sqrt{V(\tilde{\mathbf{x}}, \mathbf{x})}}\right).$$

To show that this expression is positive with probability 1 is equivalent to proving the following inequality holds with probability 1:

$$\frac{-\left(M(\tilde{\mathbf{x}}) - M(\mathbf{x})\right)}{\sqrt{V(\tilde{\mathbf{x}}, \mathbf{x})}} < \frac{\phi\left(\frac{M(\tilde{\mathbf{x}}) - M(\mathbf{x})}{\sqrt{V(\tilde{\mathbf{x}}, \mathbf{x})}}\right)}{\Phi\left(\frac{M(\tilde{\mathbf{x}}) - M(\mathbf{x})}{\sqrt{V(\tilde{\mathbf{x}}, \mathbf{x})}}\right)}. \tag{8}$$

First, we need to show that $(-M(\tilde{\mathbf{x}}) - M(\mathbf{x}))/\sqrt{V(\tilde{\mathbf{x}}, \mathbf{x})} < \infty, \forall \mathbf{x} \in \mathcal{X}$, with probability 1. This follows immediately from the fact that $V(\tilde{\mathbf{x}}, \mathbf{x}) > 0$ and $-\infty < M(\mathbf{x}) < \infty, \forall \mathbf{x} \in \mathcal{X}$, with probability 1.

Next, recall that $\tilde{\mathbf{x}}$ is chosen as the solution with the smallest *sample* mean. Therefore, it is possible that: (i) $M(\tilde{\mathbf{x}}) - M(\mathbf{x}) \geq 0$ or (ii) $M(\tilde{\mathbf{x}}) - M(\mathbf{x}) < 0$. Assume (i); then it is clear that (**??**) holds with probability 1 since for any finite argument both $\phi(\cdot) > 0$ and $\Phi(\cdot) > 0$. Now assume (ii); then

$$\begin{aligned}
\frac{\phi\left(\frac{M(\tilde{\mathbf{x}}) - M(\mathbf{x})}{\sqrt{V(\tilde{\mathbf{x}}, \mathbf{x})}}\right)}{\Phi\left(\frac{M(\tilde{\mathbf{x}}) - M(\mathbf{x})}{\sqrt{V(\tilde{\mathbf{x}}, \mathbf{x})}}\right)} &= \frac{\phi\left(\frac{-(M(\tilde{\mathbf{x}}) - M(\mathbf{x}))}{\sqrt{V(\tilde{\mathbf{x}}, \mathbf{x})}}\right)}{1 - \Phi\left(\frac{-(M(\tilde{\mathbf{x}}) - M(\mathbf{x}))}{\sqrt{V(\tilde{\mathbf{x}}, \mathbf{x})}}\right)} \\
&= \mathbb{E}\left[Z \mid Z > \frac{-(M(\tilde{\mathbf{x}}) - M(\mathbf{x}))}{\sqrt{V(\tilde{\mathbf{x}}, \mathbf{x})}}\right], \text{ where } Z \sim \text{N}(0,1) \\
&> \frac{-(M(\tilde{\mathbf{x}}) - M(\mathbf{x}))}{\sqrt{V(\tilde{\mathbf{x}}, \mathbf{x})}}.
\end{aligned}$$

The last inequality follows since $-(M(\tilde{\mathbf{x}}) - M(\mathbf{x}))/\sqrt{V(\tilde{\mathbf{x}}, \mathbf{x})} < \infty$, with probability 1. □

We next prove Theorem 1:

THEOREM 1. *rGMIA, implemented with either the adaptive or fixed-$p < \infty$ scheme, 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.*

Proof In proving Theorem 1, we draw on Theorem 2 from **?** which proves an identical result for the GMIA algorithm, which we restate here:

GMIA 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.

To prove Theorem 1, we consider rGMIA when p is fixed and finite and for our adaptive scheme.

Fixed p : When we fix $p < \infty$ it means that rGMIA will cycle between $p-1$ rapid-search iterations and a single global-search iteration until termination is reached. Global-search iterations are simply GMIA iterations, for which convergence has already by proven in Theorem 2 of **?**. The $p-1$ rapid-search iterations simply force more solutions to be simulated more often than GMIA would, and therefore do not change the convergence result.

Adaptive : Proving convergence in the adaptive scheme requires proving that the number of rapid-search iterations between global-search iterations is finite with probability 1. Recall that γ is the threshold such that whenever $\max_{\mathbf{x} \in \mathcal{S}} \text{CEI}(\tilde{\mathbf{x}}, \mathbf{x}) < \gamma$ the rapid-search iterations end. In the adaptive scheme, $\gamma = \max_{\mathbf{x} \in \mathcal{F}} \text{CEI}_t(\tilde{\mathbf{x}}, \mathbf{x})$, where $\text{CEI}_t(\tilde{\mathbf{x}}, \mathbf{x})$ are the CEI values computed during preceding global-search iteration, t. With this choice of γ, Lemma 1 implies that for finite t we are guaranteed that $\gamma > 0$.

When the search set $\mathcal{S}$ is constructed, the subgraph it induces is itself a GMRF for which we define $\tilde{\mathbf{x}} = \arg\min_{\mathbf{x} \in \mathcal{S}} \bar{Y}(\mathbf{x})$ and $\mathbf{x}^{\text{CEI}} = \arg\max_{\mathbf{x} \in \mathcal{S} \setminus \tilde{\mathbf{x}}} \text{CEI}(\tilde{\mathbf{x}}, \mathbf{x})$ to execute rapid-search iterations of rGMIA. These iterations on $\mathcal{S}$ are GMIA iterations restricted to the induced subgraph, which means that Theorem 2 from **?** yields the same convergence guarantee on the subgraph. This means that if we restrict the search to rapid-search iterations then $\text{CEI}_t(\tilde{\mathbf{x}}, \mathbf{x}) \to 0$ as the number of rapid-search iterations $t \to \infty$ for all $\mathbf{x} \in \mathcal{S}$, with probability 1. This further implies that for each sample path $\omega \in \Omega$, there exists $t^\star(\omega)$ such that for $t \geq t^\star(\omega)$, $\text{CEI}_t(\mathbf{x}, \tilde{\mathbf{x}}) < \gamma$, for any $\gamma > 0$ and for all $\mathbf{x} \in \mathcal{S}$. Therefore, we will attain termination of rapid search in a finite number of iterations with probability 1. □

Appendix B: Analysis of Computational Effort in rGMIA

We analyze the number of flops required to implement rGMIA, in Algorithm **??**, by breaking the algorithm into three sections: initialization, rapid search and global search, analyzing the matrix computations of each section separately. Recall that we make the approximation $n_{\mathcal{F}} \approx n$, since $\mathcal{S}$ is constructed to be small. We use C_F, C_I and C_L to represent the flop count to perform a matrix factorization, selected inverse of diagonal elements and single-column right-hand-side system solve involving matrices of size $n \times n$ and $n_{\mathcal{F}} \times n_{\mathcal{F}}$. In Appendix **??**, we time these three operations performed by PARDISO on $\bar{\mathbf{Q}}$ corresponding to problems with varying feasible region sizes and dimension to estimate how expensive these operations are to implement from a timing standpoint.

B.1. Initialization

1 Choose $n_0 \ll n$ initial solutions. Simulate at each solution and compute MLEs for the GMRF parameters $(\boldsymbol{\mu}, \mathbf{Q})$. 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 $\mathbf{V}(\mathbf{x}_\mathcal{X}) = \text{diag}(\bar{\mathbf{Q}}^{-1})$, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{X}) = \bar{\mathbf{Q}}^{-1}\mathbf{e}_{\tilde{\mathbf{x}}}$, $\mathbf{M}(\mathbf{x}_\mathcal{X}) = \boldsymbol{\mu} + \bar{\mathbf{Q}}^{-1}\mathbf{Q}_\epsilon(\bar{\mathbf{Y}} - \boldsymbol{\mu})$, using $\mathbf{L}_{\bar{\mathbf{Q}}}$. Go to Step 27;

Initialization in rGMIA is a one-time sequence of steps that allows us to set up global and rapid search by constructing $\mathcal{F}$ and $\mathcal{S}$. Step 3 costs C_F flops to compute the Cholesky factor. Step 4 costs C_I, C_L and $C_L + 3n$ flops to compute the conditional variances, covariances and means, respectively.

Therefore, the total flop count for initialization in rGMIA is $C_F + C_I + 2C_L + 3n$.

B.2. Rapid Search

7 Simulate at $\tilde{\mathbf{x}}$, $\mathbf{x}^{\text{CEI}}$. Update simulation information by updating $\bar{Y}(\tilde{\mathbf{x}})$, $\bar{Y}(\mathbf{x}^{\text{CEI}})$, $\mathbf{Q}_\epsilon$, $\bar{\mathbf{Q}}$, $\bar{\mathbf{Q}}_{\mathcal{SS}}$;
8 Find $\tilde{\mathbf{x}} = \arg\min_{\{\mathbf{x}\in\mathcal{S}: r(\mathbf{x})>0\}} \bar{Y}(\mathbf{x})$;
9 Compute $\mathbf{V}(\mathbf{x}_\mathcal{S})$, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{S})$ by computing $\boldsymbol{\Sigma}_{\mathcal{SS}} = (\bar{\mathbf{Q}}_{\mathcal{SS}} - \mathbf{B})^{-1}$;
10 Compute $\mathbf{M}(\mathbf{x}_\mathcal{S}) = \boldsymbol{\mu}_\mathcal{S} + \boldsymbol{\Sigma}_{\mathcal{SS}}([\mathbf{Q}_\epsilon]_{\mathcal{SS}}(\bar{\mathbf{Y}}(\mathbf{x}_\mathcal{S}) - \boldsymbol{\mu}_\mathcal{S}) - \mathbf{a})$;
11 Calculate $\text{CEI}(\tilde{\mathbf{x}}, \mathbf{x}), \forall \mathbf{x} \in \mathcal{S}$;
12 Find $\mathbf{x}^{\text{CEI}} = \arg\max_{\mathbf{x}\in\mathcal{S}\setminus\tilde{\mathbf{x}}} \text{CEI}(\tilde{\mathbf{x}}, \mathbf{x})$;

Within the rapid-search steps, only Steps 9 and 10 are computationally intensive. Step 9 computes $\boldsymbol{\Sigma}_{\mathcal{SS}}^{-1}$, a dense matrix. This involves subtracting two dense $n_\mathcal{S} \times n_\mathcal{S}$ matrices, computing the Cholesky factorization and performing a forward and backward substitution to solve the resulting systems. Note that since $|\mathcal{S}|$ is small, and we require the diagonal elements of $\boldsymbol{\Sigma}_{\mathcal{SS}}$, we compute $\boldsymbol{\Sigma}_{\mathcal{SS}}$ in its entirety. This costs $n_\mathcal{S}^2$, $\frac{1}{3}n_\mathcal{S}^3$, $n_\mathcal{S}^3$ and $n_\mathcal{S}^3$ flops, respectively. Step 10 requires one $n_\mathcal{S} \times n_\mathcal{S}$ diagonal matrix-vector multiplication and three $n_\mathcal{S} \times 1$ vector addition/subtractions, which cost $n_\mathcal{S}$ flops each, and a dense $n_\mathcal{S} \times n_\mathcal{S}$ matrix-vector multiplication, which costs $2n_\mathcal{S}^2$ flops.

Therefore, the total flop count for a single rapid-search iteration is $\frac{7}{3}n_\mathcal{S}^3 + 3n_\mathcal{S}^2 + 4n_\mathcal{S}$.

B.3. Global Search

14 Simulate at $\tilde{\mathbf{x}}$, $\mathbf{x}^{\text{CEI}}$. Update simulation information by updating $\bar{Y}(\tilde{\mathbf{x}})$, $\bar{Y}(\mathbf{x}^{\text{CEI}})$, $\mathbf{Q}_\epsilon$, $\bar{\mathbf{Q}}$, $\bar{\mathbf{Q}}_{\mathcal{SS}}$;
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 $\boldsymbol{\Sigma}_{\mathcal{SS}} = (\bar{\mathbf{Q}}_{\mathcal{SS}} - \mathbf{B})^{-1}$;
17 Compute $\mathbf{M}(\mathbf{x}_\mathcal{S}) = \boldsymbol{\mu}_\mathcal{S} + \boldsymbol{\Sigma}_{\mathcal{SS}}([\mathbf{Q}_\epsilon]_{\mathcal{SS}}(\bar{\mathbf{Y}}(\mathbf{x}_\mathcal{S}) - \boldsymbol{\mu}_\mathcal{S}) - \mathbf{a})$;
18 Compute $\mathbf{V}(\mathbf{x}_\mathcal{F}) = \text{diag}(\bar{\mathbf{Q}}_{\mathcal{FF}}^{-1}) + \text{diag}(\mathbf{A}\boldsymbol{\Sigma}_{\mathcal{SS}}\mathbf{A}^\top)$, using $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{FF}}}$;
19 Compute $\mathbf{M}(\mathbf{x}_\mathcal{F}) = \boldsymbol{\mu}_\mathcal{F} + \bar{\mathbf{Q}}_{\mathcal{FF}}^{-1}[\mathbf{Q}_\epsilon]_{\mathcal{FF}}(\bar{\mathbf{Y}}(\mathbf{x}_\mathcal{F}) - \boldsymbol{\mu}_\mathcal{F}) - \mathbf{A}(\mathbf{M}(\mathbf{x}_\mathcal{S}) - \boldsymbol{\mu}_\mathcal{S})$, using $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{FF}}}$;
20 **if** $\tilde{\mathbf{x}} \in \mathcal{S}$ **then**
21 | Compute $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{S}) = [\boldsymbol{\Sigma}_{\mathcal{SS}}]_{\cdot\tilde{\mathbf{x}}}$;
22 | Compute $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{F}) = -\mathbf{A}[\boldsymbol{\Sigma}_{\mathcal{SS}}]_{\cdot\tilde{\mathbf{x}}}$;
23 **else**
24 | Compute $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{S}) = -\boldsymbol{\Sigma}_{\mathcal{SS}}[\mathbf{A}^\top]_{\cdot\tilde{\mathbf{x}}}$;
25 | Compute $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{F}) = \bar{\mathbf{Q}}_{\mathcal{FF}}^{-1}\mathbf{e}_{\tilde{\mathbf{x}}} + \mathbf{A}\boldsymbol{\Sigma}_{\mathcal{SS}}[\mathbf{A}^\top]_{\cdot\tilde{\mathbf{x}}}$, using $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{FF}}}$;
26 **end**
27 Calculate $\text{CEI}(\tilde{\mathbf{x}}, \mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$;
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 $\bar{\mathbf{Q}}_{\mathcal{FF}}$, $\bar{\mathbf{Q}}_{\mathcal{FS}}$, $\bar{\mathbf{Q}}_{\mathcal{SS}}$;
30 Compute Cholesky factor of $\bar{\mathbf{Q}}_{\mathcal{FF}}$: $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{FF}}}$;
31 Compute $\mathbf{A} = \bar{\mathbf{Q}}_{\mathcal{FF}}^{-1}\bar{\mathbf{Q}}_{\mathcal{FS}}$, using $\mathbf{L}_{\bar{\mathbf{Q}}_{\mathcal{FF}}}$, $\mathbf{B} = \bar{\mathbf{Q}}_{\mathcal{FS}}^\top\mathbf{A}$, $\mathbf{a} = \mathbf{A}^\top([\mathbf{Q}_\epsilon]_{\mathcal{FF}}(\bar{\mathbf{Y}}(\mathbf{x}_\mathcal{F}) - \boldsymbol{\mu}_\mathcal{F}))$;

To compute $\boldsymbol{\Sigma}_{\mathcal{SS}}$ in Step 16, we incur a cost of $n_\mathcal{S}^2 + \frac{7}{3}n_\mathcal{S}^3$ flops, identical to Step 9 in the rapid-search iterations. However, to compute $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{X})$, we either incur $2n_\mathcal{S}n$ flops if $\tilde{\mathbf{x}} \in \mathcal{S}$ or $C_L + 4n_\mathcal{S}^2 + 2nn_\mathcal{S}$ flops if

$\tilde{\mathbf{x}} \in \mathcal{F}$. In computing $\mathbf{V}(\mathbf{x}_{\mathcal{F}})$, we pay a cost of C_I flops to evaluate $\text{diag}(\bar{\mathbf{Q}}^{-1}_{\mathcal{F}\mathcal{F}})$ and $\frac{1}{3}n_{\mathcal{S}}^3 + 2n_{\mathcal{S}}^2 n + n_{\mathcal{S}} n$ flops to compute $\text{diag}(\mathbf{A}\boldsymbol{\Sigma}_{\mathcal{S}\mathcal{S}}\mathbf{A}^\top)$. To implement the latter computation, we first compute the Cholesky factorization of $\boldsymbol{\Sigma}_{\mathcal{S}\mathcal{S}}$ and premultiply $\mathbf{A}$ to the resulting factor. Finally, we compute the squared norm of each of the columns of the resulting product. The squared norm of the ith column gives the ith diagonal element in $\mathbf{A}\boldsymbol{\Sigma}_{\mathcal{S}\mathcal{S}}\mathbf{A}^\top$. Since $\mathcal{S}$ is small and we only require the diagonal elements, this is more efficient than computing $\mathbf{A}\boldsymbol{\Sigma}_{\mathcal{S}\mathcal{S}}\mathbf{A}^\top$, which would cost $4n_{\mathcal{S}}^2 n$ flops. To compute $\mathbf{M}(\mathbf{x}_{\mathcal{X}})$, we incur $C_L + 4n + 5n_{\mathcal{S}} + 2n_{\mathcal{S}} n + 2n_{\mathcal{S}}^2$ flops. Finally, we incur a flop cost of C_F to factorize $\bar{\mathbf{Q}}_{\mathcal{F}\mathcal{F}}$ in Step 30 and a cost of $n_{\mathcal{S}} C_L$, $2n_{\mathcal{S}}^2 n$ and $2n + 2n_{\mathcal{S}} n$ to compute $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{a}$, respectively in Step 31.

If we assume a worst-case cost, where $\tilde{\mathbf{x}} \in \mathcal{F}$, we incur a total cost of $C_F + C_I + (n_{\mathcal{S}} + 2)C_L + 4n_{\mathcal{S}}^2 n + 7n_{\mathcal{S}} n + 6n + \frac{8}{3}n_{\mathcal{S}}^3 + 7n_{\mathcal{S}}^2 + 5n_{\mathcal{S}}$ flops in a single global-search iteration.

B.4. Analysis of PARDISO Operations

Figures **??**, **??** and **??** are log-log plots of the times to compute the Cholesky factorization, selected inverse and solving a single-column right-hand-side linear system in PARDISO for $\bar{\mathbf{Q}}$, with the structure as described in Section **??**, for problems of dimension $d = 2, 3, \ldots, 7$. For each dimension, d, we perform a linear regression on the points corresponding to operations that took longer than 0.25s to run (to mitigate effects of overhead on the trend). The slope of each regression allows us to estimate the power term corresponding to how these operations scale in time (i.e., a slope of m on the log-log plot indicates the operation scales $\mathcal{O}(n^m)$), where n is the number of feasible solutions.

We then plot these slopes in Figure **??** from which we can estimate how factorizing, performing a selected inverse and solving a linear system grow in time. Factorizing grows approximately linearly until $d = 6$ implying growth of $\mathcal{O}(n^{0.253d+0.931})$ until the problem becomes high-dimensional at which point the matrices lose much of their sparsity. Solving a linear system with PARDISO given the Cholesky factorization of the matrix remains relatively constant as dimension is increased meaning the growth scales $\mathcal{O}(n^1)$. Performing a selected inverse, however, is more difficult to characterize but varies between $\mathcal{O}(n^{1.133})$ and $\mathcal{O}(n^{1.592})$.

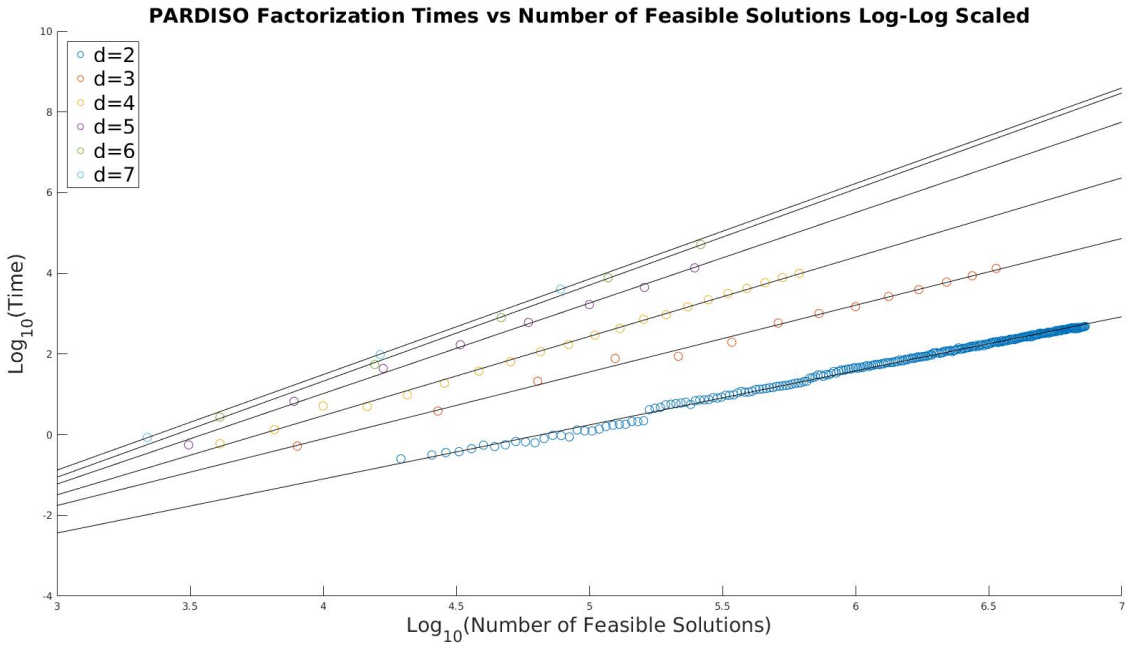


Figure 1 **Log-log plot of times to compute Cholesky factorizations of $\bar{\mathbf{Q}}$ for problems of different dimension, d, and number of feasible solutions, n.**

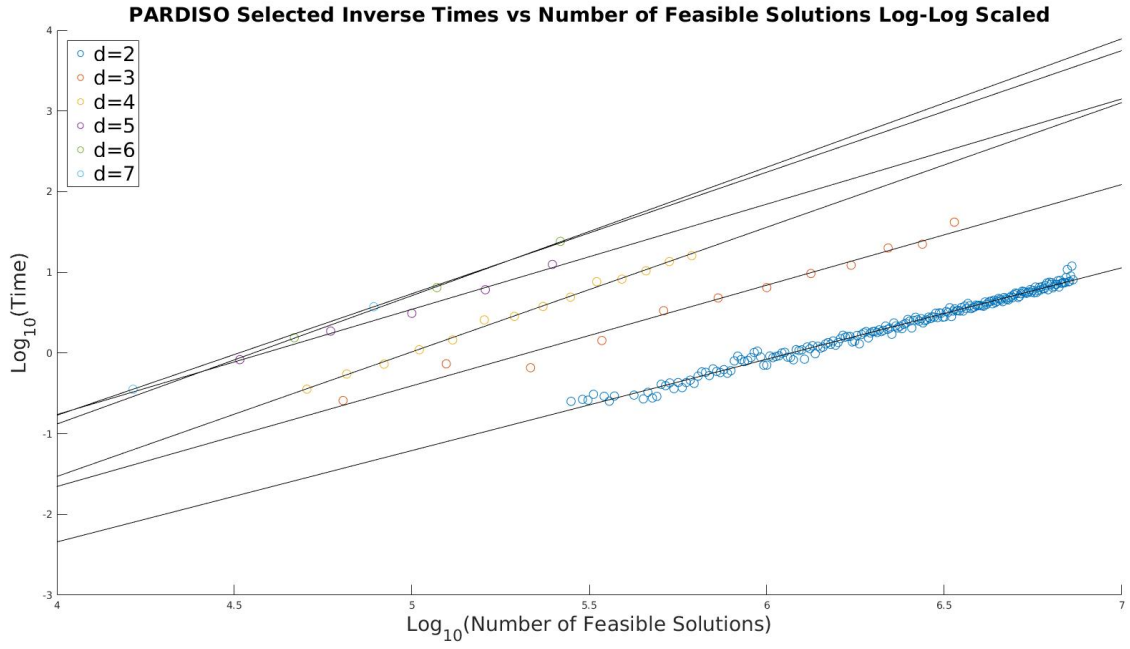


Figure 2 **Log-log plot of times to compute selected inverse of $\bar{\mathbf{Q}}$ for problems of different dimension, d, and number of feasible solutions, n.**

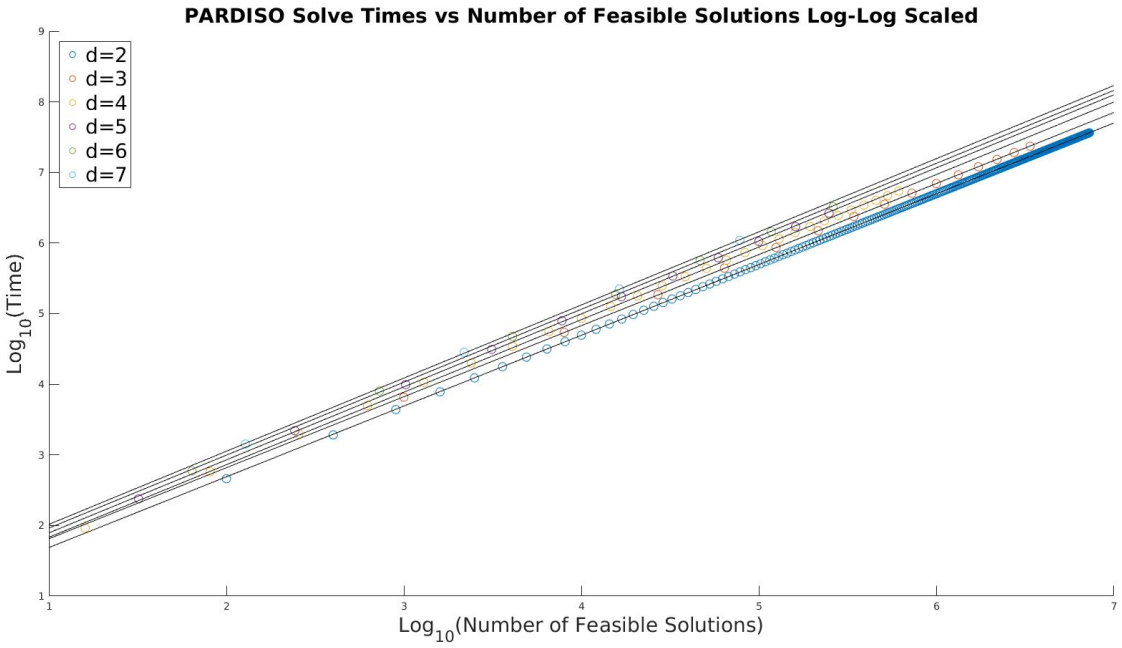


Figure 3 **Log-log plot of times to solve a single-column right-hand-side vector system with $\bar{\mathbf{Q}}$ for problems of different dimension, d, and number of feasible solutions, n.**

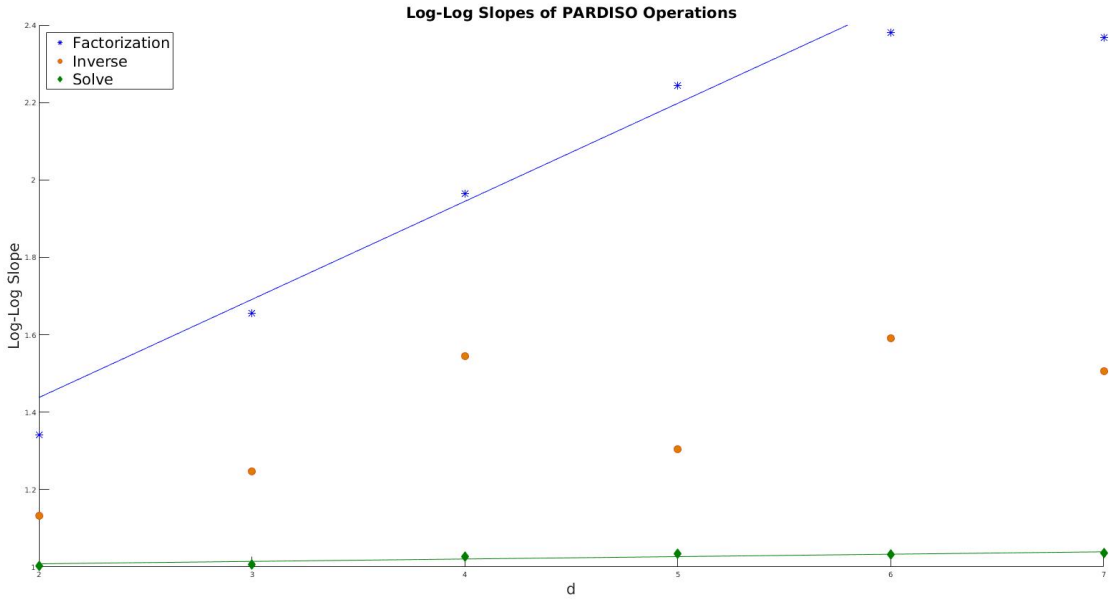


Figure 4 **Slopes of log-log linear regression of PARDISO operations times vs. number of feasible solutions as a function of dimension, d.**

Appendix C: Sherman-Morrison-Woodbury Update for GMIA

At each iteration of GMIA, exactly two solutions are simulated resulting in a rank-2 update for $\bar{\mathbf{Q}}$. Thus, we may apply the Sherman-Morrison-Woodbury (SMW) formula to efficiently update $\bar{\mathbf{Q}}^{-1}$ without factorizing and inverting $\bar{\mathbf{Q}}$ at every iteration. Suppose we factorized $\bar{\mathbf{Q}}$ and computed the entire $\bar{\mathbf{Q}}^{-1}$ in the previous iteration. These operations cost C_F and nC_L, respectively. After simulating $\tilde{\mathbf{x}}$ and $\mathbf{x}^{\text{CEI}}$, $\bar{\mathbf{Q}}$ is updated to $\bar{\bar{\mathbf{Q}}} = \bar{\mathbf{Q}} + \Delta_{\tilde{\mathbf{x}}}\mathbf{e}_{\tilde{\mathbf{x}}}\mathbf{e}_{\tilde{\mathbf{x}}}^\top + \Delta_{\mathbf{x}^{\text{CEI}}}\mathbf{e}_{\mathbf{x}^{\text{CEI}}}\mathbf{e}_{\mathbf{x}^{\text{CEI}}}^\top$, where $\Delta_{\tilde{\mathbf{x}}}$ and $\Delta_{\mathbf{x}^{\text{CEI}}}$ are some scalars reflecting the change in $\mathbf{Q}_\epsilon$ corresponding to $\tilde{\mathbf{x}}$ and $\mathbf{x}^{\text{CEI}}$, respectively. Then, the SMW formula gives

$$\bar{\bar{\mathbf{Q}}}^{-1} = \bar{\mathbf{Q}}^{-1} - \underbrace{\bar{\mathbf{Q}}^{-1}\mathbf{E}\left(\mathbf{I}_{2\times 2} + \mathbf{E}^\top\bar{\mathbf{Q}}^{-1}\boldsymbol{\Delta}\right)^{-1}\boldsymbol{\Delta}^\top\bar{\mathbf{Q}}^{-1}}_{(*)},$$

where $\mathbf{E} = \left[\mathbf{e}_{\tilde{\mathbf{x}}},\, \mathbf{e}_{\mathbf{x}^{\text{CEI}}}\right]$ and $\boldsymbol{\Delta} = \left[\Delta_{\tilde{\mathbf{x}}}\mathbf{e}_{\tilde{\mathbf{x}}},\, \Delta_{\mathbf{x}^{\text{CEI}}}\mathbf{e}_{\mathbf{x}^{\text{CEI}}}\right]$. Notice that $(*)$ is a product of $n\times 2$ matrix $\bar{\mathbf{Q}}^{-1}\mathbf{E}$ and $2\times n$ matrix $\left(\mathbf{I}_{2\times 2} + \mathbf{E}^\top\bar{\mathbf{Q}}^{-1}\boldsymbol{\Delta}\right)^{-1}\boldsymbol{\Delta}^\top\bar{\mathbf{Q}}^{-1}$. The former is simply a matrix of two column vectors of $\bar{\mathbf{Q}}^{-1}$ corresponding to $\tilde{\mathbf{x}}$ and $\mathbf{x}^{\text{CEI}}$, which is available for free from the previous iteration.

Further, $\mathbf{E}^\top\bar{\mathbf{Q}}^{-1}\boldsymbol{\Delta}$ is a 2×2 diagonal matrix, where the diagonal elements are the products of the diagonal elements of $\bar{\mathbf{Q}}^{-1}$ corresponding to $\tilde{\mathbf{x}}$ and $\mathbf{x}^{\text{CEI}}$, and $\Delta_{\tilde{\mathbf{x}}}$ and $\Delta_{\mathbf{x}^{\text{CEI}}}$, respectively; this costs only two flops. Since $\mathbf{I}_{2\times 2} + \mathbf{E}^\top\bar{\mathbf{Q}}^{-1}\boldsymbol{\Delta}$ is a diagonal matrix, its inverse only costs two flops to compute and $\left(\mathbf{I}_{2\times 2} + \mathbf{E}^\top\bar{\mathbf{Q}}^{-1}\boldsymbol{\Delta}\right)^{-1}\boldsymbol{\Delta}^\top\bar{\mathbf{Q}}^{-1}$ is simply rescaling the columns of $\mathbf{E}^\top\bar{\mathbf{Q}}^{-1}$, which costs $2n$ flops. Finally, the product of $n\times 2$ and $2\times n$ matrices costs $\mathcal{O}(n^2)$ (for $(*)$) and so does subtracting $(*)$ from $\bar{\mathbf{Q}}^{-1}$.

The conditional means of solutions can be computed from (**??**) using the updated $\bar{\bar{\mathbf{Q}}}$, which costs $\mathcal{O}(n^2)$. Notice that for this computation, we perform a matrix-vector multiplication instead of solving a linear system of equations since we did not factorize $\bar{\bar{\mathbf{Q}}}$.

Overall, the SMW scheme costs $C_F + nC_L$ for computing $\bar{\mathbf{Q}}^{-1}$ for one iteration, then costs $\mathcal{O}(n^2)$ for each iteration thereafter. This is certainly cheaper than directly updating the conditional distribution as expressed in (**??**), but it still requires *all* elements of the inverse precision matrix to be updated at each iteration. This forces us to recompute $\bar{\mathbf{Q}}^{-1}$ periodically which diminishes the computational gain.

We refer to this version of GMIA that uses the SMW formula to quickly update the conditional distribution as SMW_GMIA and present it in Algorithm **??**. The primary difference between GMIA and SMW_GMIA is that instead of refactorizing, inverting and solving a system with $\bar{\bar{\mathbf{Q}}}$ from scratch at each iteration, we compute, in full, and update at each iteration $\bar{\mathbf{Q}}^{-1}$. We use the notation $\mathbf{P} = \bar{\mathbf{Q}}^{-1}$ to make explicit that computing an expression such as $\mathbf{Px}$, for an appropriately sized $\mathbf{x}$, results in a matrix-vector multiplication and not a system solve.

SMW_GMIA has the advantage of not requiring $\bar{\mathbf{Q}}$ to be factorized each iteration saving computational overhead, but by computing all elements of $\bar{\mathbf{Q}}^{-1}$, it loses the savings generated by using sparse linear algebra computations. Refer to Table **??** for the results comparing SMW_GMIA to rGMIA (with $|\mathcal{S}| = 50$, $p = 50$) and GMIA, all applied to the **inventory_100** problem. We see that SMW_GMIA results in mean runtimes that are well over an order of magnitude larger than corresponding runtimes rGMIA and are even much larger than corresponding runtimes of GMIA applied to the same problem.

Additionally, the **inventory_100** problem with a 100×100 integer lattice feasible region was the largest problem that could be solved via SMW_GMIA with the memory resources available to us. To make this

point concrete, the **inventory_100** problem results in $\bar{\mathbf{Q}}$ and $\mathbf{P}$ matrices that are both of size 10000×10000. However, since the former is sparse, in MATLAB, it requires about 0.874 MB to store, while the latter requires 800 MB. Therefore, for most users, this memory requirement results in a very strict constraint in problem size that can be solved.

Algorithm 3: SMW_GMIA

Choose $n_0 \ll n$ initial solutions. Simulate at each solution and compute MLEs for the GMRF parameters $(\boldsymbol{\mu}, \boldsymbol{\theta})$. Construct $\bar{\mathbf{Q}} = \mathbf{Q} + \mathbf{Q}_\epsilon$;
Find $\tilde{\mathbf{x}} = \arg\min_{\{\mathbf{x} \in \mathcal{X}: r(\mathbf{x}) > 0\}} \bar{Y}(\mathbf{x})$;
Compute Cholesky factorization of $\bar{\mathbf{Q}}$: $\mathbf{L}_{\bar{\mathbf{Q}}}$;
Compute $\mathbf{P} = \bar{\mathbf{Q}}^{-1}$, using $\mathbf{L}_{\bar{\mathbf{Q}}}$;
Extract $\mathbf{V}(\mathbf{x}_\mathcal{X}) = \text{diag}\left(\bar{\mathbf{Q}}^{-1}\right)$, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{X}) = \bar{\mathbf{Q}}^{-1}\mathbf{e}_{\tilde{\mathbf{x}}}$ and compute $\mathbf{M}(\mathbf{x}_\mathcal{X}) = \boldsymbol{\mu} + \bar{\mathbf{Q}}^{-1}\mathbf{Q}_\epsilon(\bar{\mathbf{Y}} - \boldsymbol{\mu})$, using $\mathbf{L}_{\bar{\mathbf{Q}}}$;
Calculate $\text{CEI}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{X})$;
while *Termination criterion not reached* **do**
  Simulate at $\tilde{\mathbf{x}}$, $\mathbf{x}^{\text{CEI}}$. Update simulation information by updating $\bar{Y}(\tilde{\mathbf{x}})$, $\bar{Y}(\mathbf{x}^{\text{CEI}})$, $\mathbf{Q}_\epsilon$;
  Construct $\mathbf{E} = [\mathbf{e}_{\tilde{\mathbf{x}}}, \mathbf{e}_{\mathbf{x}^{\text{CEI}}}]$ and $\boldsymbol{\Delta} = [\Delta_{\tilde{\mathbf{x}}}\mathbf{e}_{\tilde{\mathbf{x}}}, \Delta_{\mathbf{x}^{\text{CEI}}}\mathbf{e}_{\mathbf{x}^{\text{CEI}}}]$;
  Update $\bar{\bar{\mathbf{Q}}} \leftarrow \bar{\mathbf{Q}} + \mathbf{E}\boldsymbol{\Delta}^\top$;
  Update $\mathbf{P} \leftarrow \mathbf{P} - \mathbf{P}\mathbf{E}(\mathbf{I}_{2\times 2} + \mathbf{E}^\top\mathbf{P}\boldsymbol{\Delta})^{-1}\boldsymbol{\Delta}^\top\mathbf{P}$;
  Find $\tilde{\mathbf{x}} = \arg\min_{\mathbf{x} \in \mathcal{X}: r(\mathbf{x}) > 0} \bar{Y}(\mathbf{x})$;
  Extract $\mathbf{V}(\mathbf{x}_\mathcal{X}) = \text{diag}(\mathbf{P})$, $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{X}) = [\mathbf{P}]_{\cdot\tilde{\mathbf{x}}}$ and compute $\mathbf{M}(\mathbf{x}_\mathcal{X}) = \boldsymbol{\mu} + \mathbf{P}\mathbf{Q}_\epsilon(\bar{\mathbf{Y}} - \boldsymbol{\mu})$;
  Calculate $\text{CEI}(\tilde{\mathbf{x}}, \mathbf{x}_\mathcal{X})$;
  Find $\mathbf{x}^{\text{CEI}} = \arg\max_{\mathbf{x} \in \mathcal{X} \setminus \tilde{\mathbf{x}}} \text{CEI}(\tilde{\mathbf{x}}, \mathbf{x})$;
  Let $\bar{\mathbf{Q}} \leftarrow \bar{\bar{\mathbf{Q}}}$;
end

Table 4 Fixed-precision results averaged from 30 macro-replications of SMW_GMIA, rGMIA ($|\mathcal{S}| = 50$, $p = 50$) and GMIA applied to the inventory_100 problem. Standard errors of mean values are provided in parentheses.

Algorithm	SMW_GMIA			rGMIA ($\lvert\mathcal{S}\rvert = 50$, $p = 50$)			GMIA		
δ	0.1	0.05	0.01	0.1	0.05	0.01	0.1	0.05	0.01
Mean Time Elapsed (s)	4840 (296)	5434 (317)	7338 (398)	102 (4)	112 (4)	140 (4)	1186 (42)	1333 (41)	1802 (47)
Max Time Elapsed (s)	8379	9029	10845	144	154	178	1618	1744	2305
Mean Optimality Gap	0.13 (0.03)	0.16 (0.04)	0.05 (0.01)	0.08 (0.02)	0.04 (0.01)	0.03 (0.01)	0.13 (0.03)	0.16 (0.04)	0.05 (0.01)
Max Optimality Gap	0.77	0.93	0.30	0.64	0.23	0.12	0.77	0.93	0.30
Mean Number of Iterations	6722 (238)	7592 (236)	10498 (279)	7146 (230)	7868 (229)	10233 (247)	6722 (238)	7592 (236)	10498 (279)

Appendix D: Additional Results Evaluating rGMIA

Complete tables for all of our experiments are found in this appendix.

Table 5 Parameters used in the restaurant problem.

Problem	**restaurant_125**	**restaurant_25**	**restaurant_5**
Feasible Region	125×125	$25 \times 25 \times 25$	$5 \times 5 \times 5 \times 5 \times 5 \times 5$
Table Sizes Available (s)	$[1\ 3]$	$[1\ 3\ 5]$	$[1\ 3\ 5\ 7\ 9\ 11]$
Time (T)	1	1	1
Arrival Rate (λ)	250	50	10
Service Rate (μ)	10	10	10
Revenue/person in $1000s (r)	0.01	0.01	0.01
Cost of Table/hr in $1000s (c)	$[0.005\ 0.015]$	$[0.005\ 0.015\ 0.025]$	$[0.005\ 0.015\ 0.025\ 0.035\ 0.045\ 0.055]$

Table 6 **Fixed-precision results averaged from 30 macro-replications of rGMIA and GMIA applied to the inventory_100 problem. Standard errors of mean values are provided in parentheses.**

$\lvert\mathcal{S}\rvert$	50											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	102 (4)	124 (4)	173 (6)	140 (5)	112 (4)	134 (4)	186 (6)	154 (5)	140 (4)	155 (4)	213 (6)	182 (4)
Max Time Elapsed (s)	144	174	232	191	154	183	242	204	178	200	260	223
Mean Optimality Gap	0.08 (0.02)	0.07 (0.01)	0.04 (0.01)	0.24 (0.09)	0.04 (0.01)	0.09 (0.02)	0.03 (0.01)	0.08 (0.02)	0.03 (0.01)	0.05 (0.01)	0.02 (0.00)	0.06 (0.02)
Max Optimality Gap	0.64	0.14	0.13	2.08	0.23	0.49	0.13	0.53	0.12	0.23	0.11	0.53
Mean Number of Iterations	7146 (230)	14034 (452)	27988 (904)	6709 (236)	7868 (229)	15238 (442)	30168 (872)	7530 (237)	10233 (247)	17818 (394)	34561 (799)	9994 (261)

$\lvert\mathcal{S}\rvert$	100											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	126 (5)	105 (4)	126 (5)	126 (4)	141 (5)	114 (4)	136 (5)	140 (4)	188 (7)	144 (5)	161 (7)	169 (5)
Max Time Elapsed (s)	174	144	187	170	187	156	199	186	276	193	236	215
Mean Optimality Gap	0.12 (0.03)	0.10 (0.02)	0.06 (0.02)	0.13 (0.03)	0.07 (0.01)	0.08 (0.02)	0.03 (0.01)	0.06 (0.01)	0.04 (0.01)	0.03 (0.01)	0.02 (0.01)	0.05 (0.01)
Max Optimality Gap	0.80	0.49	0.57	0.52	0.22	0.38	0.13	0.22	0.13	0.14	0.13	0.14
Mean Number of Iterations	6754 (230)	7354 (214)	14434 (423)	6715 (237)	7558 (232)	8084 (214)	15581 (404)	7564 (243)	10218 (298)	10444 (248)	18288 (484)	10149 (285)

$\lvert\mathcal{S}\rvert$	200											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	182 (7)	126 (6)	113 (4)	137 (5)	207 (8)	145 (8)	126 (5)	155 (6)	296 (13)	209 (12)	183 (12)	221 (15)
Max Time Elapsed (s)	269	187	161	201	313	230	189	224	457	344	362	555
Mean Optimality Gap	0.07 (0.02)	0.12 (0.03)	0.06 (0.01)	0.12 (0.04)	0.05 (0.01)	0.09 (0.02)	0.07 (0.02)	0.11 (0.03)	0.04 (0.01)	0.03 (0.01)	0.03 (0.01)	0.03 (0.01)
Max Optimality Gap	0.49	0.91	0.27	1.05	0.32	0.57	0.36	0.91	0.19	0.14	0.14	0.12
Mean Number of Iterations	6709 (233)	6834 (229)	7741 (216)	6803 (241)	7518 (231)	7671 (235)	8408 (203)	7651 (243)	10176 (229)	10361 (263)	10761 (273)	10964 (425)

Table 7 **Fixed-precision results averaged from 29 macro-replications of rGMIA and GMIA applied to the inventory_150 problem. Standard errors of mean values are provided in parentheses.**

$\lvert\mathcal{S}\rvert$	50											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	352 (20)	412 (25)	540 (34)	403 (26)	386 (20)	450 (24)	588 (34)	436 (26)	471 (19)	522 (24)	674 (32)	495 (26)
Max Time Elapsed (s)	535	628	899	591	568	652	922	619	633	696	964	676
Mean Optimality Gap	0.05 (0.01)	0.05 (0.01)	0.03 (0.01)	0.12 (0.03)	0.08 (0.03)	0.06 (0.02)	0.03 (0.01)	0.05 (0.01)	0.03 (0.01)	0.02 (0.01)	0.01 (0.00)	0.03 (0.01)
Max Optimality Gap	0.14	0.23	0.14	0.64	0.74	0.49	0.14	0.13	0.25	0.13	0.07	0.19
Mean Number of Iterations	14770 (803)	29235 (1624)	58477 (3215)	14122 (804)	16199 (777)	31784 (1543)	63470 (3079)	15546 (802)	19775 (692)	36718 (1469)	72367 (2832)	19313 (745)
$\lvert\mathcal{S}\rvert$	100											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	470 (28)	318 (18)	383 (21)	341 (23)	520 (28)	348 (17)	416 (20)	372 (23)	658 (27)	416 (15)	480 (19)	429 (23)
Max Time Elapsed (s)	783	491	652	510	785	496	657	541	873	531	706	589
Mean Optimality Gap	0.12 (0.04)	0.07 (0.03)	0.08 (0.02)	0.14 (0.04)	0.09 (0.02)	0.04 (0.01)	0.05 (0.01)	0.06 (0.02)	0.06 (0.02)	0.02 (0.01)	0.04 (0.01)	0.02 (0.01)
Max Optimality Gap	0.80	0.80	0.53	0.91	0.60	0.14	0.23	0.52	0.32	0.13	0.19	0.13
Mean Number of Iterations	14149 (797)	15053 (768)	29842 (1547)	14139 (793)	15608 (774)	16439 (749)	32291 (1487)	15550 (780)	19592 (698)	19660 (644)	37084 (1346)	19240 (727)
$\lvert\mathcal{S}\rvert$	200											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	753 (50)	458 (27)	333 (17)	367 (25)	839 (49)	511 (27)	366 (16)	401 (25)	1076 (43)	652 (23)	449 (13)	475 (24)
Max Time Elapsed (s)	1337	822	468	565	1360	828	501	594	1480	876	555	635
Mean Optimality Gap	0.11 (0.03)	0.06 (0.02)	0.14 (0.04)	0.12 (0.05)	0.06 (0.02)	0.08 (0.02)	0.11 (0.05)	0.07 (0.01)	0.04 (0.01)	0.03 (0.01)	0.02 (0.01)	0.03 (0.01)
Max Optimality Gap	0.57	0.37	1.01	1.54	0.49	0.39	1.50	0.23	0.32	0.13	0.14	0.11
Mean Number of Iterations	14079 (803)	14298 (784)	15642 (745)	14267 (777)	15518 (791)	15746 (772)	17029 (707)	15648 (782)	19273 (702)	19422 (690)	20263 (590)	19465 (710)

Reported results for **inventory_150** is averaged across 29 macro-replication with outlier macro-replication removed. This is further explained in Section **??**.

Table 8 **Fixed-precision results averaged from 30 macro-replications of rGMIA and GMIA applied to the griewank_101 problem. Standard errors of mean values are provided in parentheses.**

$\lvert\mathcal{S}\rvert$	50											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	33 (2)	37 (2)	38 (2)	33 (2)	71 (1)	80 (1)	81 (3)	75 (1)	113 (1)	128 (2)	134 (4)	117 (2)
Max Time Elapsed (s)	48	54	69	53	80	91	115	86	120	143	176	133
Mean Optimality Gap	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Max Optimality Gap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mean Number of Iterations	2871 (166)	5734 (331)	11601 (660)	2339 (125)	6406 (45)	12801 (88)	25621 (175)	5899 (44)	9861 (19)	19721 (38)	39454 (74)	9617 (19)
$\lvert\mathcal{S}\rvert$	100											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	41 (2)	31 (2)	36 (2)	33 (2)	94 (1)	61 (1)	73 (1)	71 (1)	168 (1)	102 (1)	131 (1)	119 (1)
Max Time Elapsed (s)	57	43	52	52	103	68	83	83	175	110	139	131
Mean Optimality Gap	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Max Optimality Gap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mean Number of Iterations	2499 (131)	3158 (185)	6361 (362)	2456 (106)	5948 (45)	6664 (44)	13308 (81)	5904 (46)	9601 (19)	9841 (18)	19694 (36)	9627 (20)
$\lvert\mathcal{S}\rvert$	200											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	66 (4)	43 (2)	38 (2)	38 (2)	157 (2)	95 (1)	75 (2)	81 (1)	361 (3)	246 (2)	174 (1)	211 (2)
Max Time Elapsed (s)	105	62	59	62	189	109	97	97	381	262	185	221
Mean Optimality Gap	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Max Optimality Gap	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00
Mean Number of Iterations	2356 (128)	2611 (135)	3634 (199)	2543 (102)	5898 (45)	6004 (47)	7101 (42)	5946 (48)	9598 (19)	9618 (19)	9954 (21)	9666 (22)

Table 9 **Fixed-precision results averaged from 30 macro-replications of rGMIA and GMIA applied to the griewank_201 problem. Standard errors of mean values are provided in parentheses.**

$\lvert\mathcal{S}\rvert$	50											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	487 (28)	515 (29)	581 (33)	531 (35)	1136 (20)	1205 (22)	1351 (26)	1282 (25)	1812 (28)	1913 (34)	2122 (40)	2046 (37)
Max Time Elapsed (s)	714	755	826	855	1286	1369	1537	1539	1986	2111	2381	2282
Mean Optimality Gap	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Max Optimality Gap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mean Number of Iterations	10769 (580)	21548 (1154)	43074 (2311)	9499 (502)	24651 (172)	49314 (343)	98674 (682)	23437 (175)	38879 (73)	77761 (146)	155541 (293)	38136 (75)

$\lvert\mathcal{S}\rvert$	100											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	632 (34)	410 (23)	445 (25)	433 (32)	1558 (14)	926 (7)	1008 (13)	1025 (21)	2229 (11)	1419 (7)	1506 (13)	1609 (32)
Max Time Elapsed (s)	917	584	654	716	1696	1003	1124	1228	2291	1472	1570	1877
Mean Optimality Gap	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Max Optimality Gap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mean Number of Iterations	9669 (526)	11381 (643)	22794 (1286)	9530 (485)	23523 (172)	25041 (166)	50128 (338)	23393 (173)	37968 (75)	38618 (72)	77241 (143)	38004 (74)

$\lvert\mathcal{S}\rvert$	200											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	848 (42)	499 (27)	351 (19)	438 (30)	2017 (31)	1247 (30)	807 (17)	1085 (21)	3039 (33)	1882 (34)	1191 (19)	1697 (33)
Max Time Elapsed (s)	1305	730	529	741	2334	1539	1015	1330	3350	2172	1369	1987
Mean Optimality Gap	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Max Optimality Gap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mean Number of Iterations	9449 (523)	9951 (533)	12441 (694)	9881 (430)	23373 (174)	23684 (168)	25961 (151)	23410 (175)	37958 (75)	37988 (75)	38681 (70)	38034 (77)

Table 10 **Fixed-precision results averaged from 30 macro-replications of rGMIA and GMIA applied to the griewank_301 problem. Standard errors of mean values are provided in parentheses.**

$\lvert\mathcal{S}\rvert$	50											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	562 (35)	720 (53)	676 (34)	665 (37)	4007 (109)	4835 (182)	4834 (83)	4891 (146)	6745 (92)	7931 (213)	8108 (51)	7966 (153)
Max Time Elapsed (s)	1228	1748	1390	1256	5992	6643	5602	6225	8670	10741	8552	9376
Mean Optimality Gap	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Max Optimality Gap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mean Number of Iterations	7931 (303)	16014 (725)	32294 (1472)	7958 (354)	51973 (782)	103921 (1574)	207888 (3144)	50314 (749)	88474 (103)	176968 (208)	353908 (415)	87427 (104)
$\lvert\mathcal{S}\rvert$	100											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	865 (47)	610 (40)	753 (35)	619 (29)	5937 (197)	3878 (122)	4355 (65)	4441 (140)	9470 (240)	6047 (168)	6763 (89)	7021 (195)
Max Time Elapsed (s)	1653	1477	1383	1061	8347	5420	4876	5733	13365	8206	7495	8082
Mean Optimality Gap	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Max Optimality Gap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mean Number of Iterations	7424 (287)	8888 (414)	17908 (825)	8507 (380)	50061 (762)	52341 (754)	104654 (1507)	50004 (730)	86573 (104)	87738 (97)	175434 (195)	86774 (102)
$\lvert\mathcal{S}\rvert$	200											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	1194 (63)	725 (39)	789 (33)	514 (18)	9247 (351)	5674 (193)	3859 (65)	3377 (88)	15147 (343)	9629 (290)	5871 (99)	5305 (71)
Max Time Elapsed (s)	2160	1511	1408	857	13757	8048	4883	3864	19669	13226	7378	5614
Mean Optimality Gap	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Max Optimality Gap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mean Number of Iterations	7113 (349)	7498 (363)	11121 (439)	10126 (291)	49301 (1180)	50334 (736)	53474 (730)	49630 (969)	86571 (106)	86598 (104)	87534 (94)	86682 (106)

Table 11 **Fixed-precision results averaged from 30 macro-replications of rGMIA and GMIA applied to the griewank_401 problem. Standard errors of mean values are provided in parentheses.**

$\lvert\mathcal{S}\rvert$	200											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	4369 (202)	2421 (154)	1782 (105)	1774 (128)	32987 (632)	17239 (814)	10140 (517)	11571 (469)	55813 (655)	28420 (1274)	15920 (779)	17981 (691)
Max Time Elapsed (s)	8484	4743	2816	4434	39881	25320	14392	16158	60565	41018	21650	26225
Mean Optimality Gap	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Max Optimality Gap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mean Number of Iterations	13068 (596)	13544 (617)	17814 (662)	17563 (612)	88483 (1354)	89061 (1299)	93054 (1309)	88721 (1282)	153633 (185)	153648 (185)	154994 (174)	153784 (183)

We were unable to run $n_S = 50, 100$ **griewank_401** experiments long enough to satisfy the termination criterion given memory constraints.

Table 12 **Fixed-precision results averaged from 30 macro-replications of rGMIA and GMIA applied to the restaurant_125 problem. Standard errors of mean values are provided in parentheses.**

$\lvert\mathcal{S}\rvert$	50											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	4538 (689)	5564 (844)	6796 (1033)	4247 (703)	5034 (727)	6074 (875)	7640 (1095)	5139 (761)	6727 (772)	7628 (862)	9597 (1090)	6806 (777)
Max Time Elapsed (s)	8387	9934	12458	8670	8940	10344	13331	9625	10398	11041	14267	10846
Mean Optimality Gap	0.12 (0.02)	0.10 (0.02)	0.09 (0.02)	0.13 (0.02)	0.09 (0.02)	0.08 (0.02)	0.07 (0.02)	0.10 (0.02)	0.06 (0.01)	0.06 (0.01)	0.06 (0.01)	0.08 (0.01)
Max Optimality Gap	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.31	0.31	0.31	0.31
Mean Number of Iterations	5946 (904)	11894 (1809)	23761 (3614)	5465 (834)	6851 (968)	13508 (1909)	27068 (3825)	8028 (1596)	11068 (1249)	17981 (2009)	33814 (3790)	12036 (1575)
$\lvert\mathcal{S}\rvert$	100											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	4787 (731)	4773 (737)	6334 (990)	4168 (691)	5319 (776)	5257 (773)	7003 (1049)	6061 (1439)	7324 (851)	7096 (835)	8896 (1056)	9197 (1918)
Max Time Elapsed (s)	9019	9252	12030	8595	9441	9645	12690	43466	11340	11494	13706	48419
Mean Optimality Gap	0.13 (0.02)	0.12 (0.02)	0.10 (0.02)	0.12 (0.02)	0.11 (0.02)	0.10 (0.02)	0.08 (0.02)	0.11 (0.02)	0.06 (0.01)	0.07 (0.01)	0.06 (0.01)	0.08 (0.01)
Max Optimality Gap	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.31	0.31	0.31	0.31
Mean Number of Iterations	5639 (857)	6051 (920)	12061 (1834)	5525 (842)	6483 (917)	6921 (977)	13721 (1936)	11253 (4601)	10874 (1228)	11458 (1293)	18134 (2015)	24702 (10411)
$\lvert\mathcal{S}\rvert$	200											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	4703 (724)	4593 (709)	4837 (740)	4142 (685)	5262 (770)	5112 (753)	5320 (775)	6368 (1591)	7394 (863)	7094 (837)	7408 (852)	10325 (2402)
Max Time Elapsed (s)	9193	8861	9011	8536	9733	9409	9395	49033	11607	11259	11698	65572
Mean Optimality Gap	0.11 (0.02)	0.12 (0.02)	0.13 (0.02)	0.12 (0.02)	0.09 (0.02)	0.10 (0.02)	0.09 (0.02)	0.10 (0.02)	0.06 (0.01)	0.05 (0.01)	0.06 (0.01)	0.08 (0.02)
Max Optimality Gap	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.31	0.31	0.31	0.31
Mean Number of Iterations	5599 (852)	5671 (862)	6261 (951)	5446 (830)	6483 (916)	6578 (927)	7154 (1007)	17517 (10109)	10816 (1216)	11104 (1261)	12328 (1369)	36199 (16196)

Table 13 **Fixed-precision results averaged from 30 macro-replications of rGMIA and GMIA applied to the restaurant_25 problem. Standard errors of mean values are provided in parentheses.**

$\lvert\mathcal{S}\rvert$	50											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	1447 (192)	2194 (288)	3866 (508)	2365 (324)	1905 (183)	2893 (269)	5123 (476)	3068 (301)	3298 (215)	4415 (221)	7208 (363)	5242 (374)
Max Time Elapsed (s)	2504	3633	6285	4315	2788	3967	6929	4723	4246	5192	8301	12735
Mean Optimality Gap	0.07 (0.01)	0.07 (0.01)	0.06 (0.01)	0.07 (0.01)	0.06 (0.01)	0.07 (0.01)	0.05 (0.01)	0.07 (0.01)	0.05 (0.01)	0.05 (0.01)	0.04 (0.00)	0.07 (0.01)
Max Optimality Gap	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.15	0.12	0.09	0.18
Mean Number of Iterations	6469 (851)	13074 (1720)	26308 (3460)	6224 (819)	8746 (814)	17304 (1611)	34848 (3240)	8314 (756)	19933 (1262)	28981 (1475)	50161 (2517)	19356 (3813)
$\lvert\mathcal{S}\rvert$	100											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	2024 (267)	1801 (241)	2405 (319)	1962 (263)	2641 (252)	2375 (227)	3180 (303)	2567 (235)	4597 (308)	4297 (243)	4765 (270)	4332 (186)
Max Time Elapsed (s)	3359	3209	4014	3663	3801	3529	4464	3936	5981	5841	6130	7383
Mean Optimality Gap	0.07 (0.01)	0.07 (0.01)	0.07 (0.01)	0.09 (0.01)	0.07 (0.01)	0.07 (0.01)	0.06 (0.01)	0.06 (0.01)	0.05 (0.01)	0.05 (0.01)	0.04 (0.00)	0.06 (0.01)
Max Optimality Gap	0.19	0.26	0.19	0.27	0.19	0.20	0.19	0.19	0.10	0.12	0.10	0.16
Mean Number of Iterations	6254 (823)	6488 (853)	13101 (1722)	6222 (814)	8441 (785)	8784 (814)	17434 (1616)	8620 (669)	19328 (1219)	22101 (1157)	29741 (1523)	17968 (1692)
$\lvert\mathcal{S}\rvert$	200											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	2162 (288)	1873 (248)	1793 (237)	1993 (263)	2826 (274)	2478 (235)	2457 (210)	2784 (195)	5057 (348)	4416 (244)	4235 (241)	4589 (312)
Max Time Elapsed (s)	3748	3268	2991	3563	4116	3596	3360	3867	6831	5472	5738	13010
Mean Optimality Gap	0.08 (0.01)	0.08 (0.01)	0.07 (0.01)	0.06 (0.01)	0.06 (0.01)	0.06 (0.01)	0.06 (0.01)	0.07 (0.01)	0.05 (0.00)	0.05 (0.01)	0.05 (0.01)	0.05 (0.00)
Max Optimality Gap	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.11	0.16	0.12	0.12
Mean Number of Iterations	6263 (824)	6288 (826)	6628 (870)	6367 (807)	8404 (783)	8504 (788)	9328 (772)	9968 (751)	19764 (1256)	20641 (1078)	23001 (1273)	22176 (4143)

Table 14 **Fixed-precision results averaged from 30 macro-replications of rGMIA ($|\mathcal{S}| = 50$) and GMIA applied to the restaurant_5 problem. Standard errors of mean values are provided in parentheses.**

$\lvert\mathcal{S}\rvert$	50											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	7221 (1256)	10212 (1760)	11717 (2015)	20095 (3836)	11487 (1406)	15075 (1966)	16890 (2047)	29970 (4282)	29537 (1608)	28435 (1026)	27360 (541)	56981 (3936)
Max Time Elapsed (s)	15737	21135	23450	55710	18369	24979	26599	62941	37269	32685	30624	81357
Mean Optimality Gap	0.07 (0.01)	0.07 (0.01)	0.06 (0.01)	0.07 (0.01)	0.05 (0.01)	0.06 (0.01)	0.05 (0.00)	0.07 (0.01)	0.04 (0.00)	0.03 (0.00)	0.03 (0.00)	0.06 (0.01)
Max Optimality Gap	0.31	0.31	0.31	0.31	0.14	0.15	0.11	0.15	0.11	0.08	0.07	0.19
Mean Number of Iterations	6169 (1072)	12638 (2196)	25614 (4448)	5430 (995)	9511 (1161)	18408 (2414)	38568 (4708)	8538 (1124)	22066 (1122)	35151 (1244)	65248 (478)	17167 (1056)
$\lvert\mathcal{S}\rvert$	100											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	13057 (2256)	6551 (1129)	7301 (1220)	13254 (2484)	19254 (2551)	10805 (1215)	11391 (1288)	17940 (2693)	45365 (2381)	26672 (750)	21210 (400)	33102 (2195)
Max Time Elapsed (s)	27228	14303	15567	34919	31874	17024	17912	39207	55587	32732	23673	49310
Mean Optimality Gap	0.08 (0.01)	0.08 (0.01)	0.08 (0.01)	0.08 (0.01)	0.05 (0.01)	0.06 (0.01)	0.06 (0.01)	0.06 (0.01)	0.04 (0.00)	0.05 (0.01)	0.03 (0.00)	0.04 (0.01)
Max Optimality Gap	0.31	0.31	0.31	0.31	0.14	0.17	0.14	0.16	0.11	0.15	0.11	0.13
Mean Number of Iterations	6026 (1048)	6154 (1065)	12754 (2151)	5484 (980)	8989 (1182)	9954 (1114)	19788 (2225)	8237 (1136)	21783 (1100)	24578 (387)	36548 (340)	17775 (890)
$\lvert\mathcal{S}\rvert$	200											
δ	0.1				0.05				0.01			
Algorithm	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive	$p=50$	$p=100$	$p=200$	Adaptive
Mean Time Elapsed (s)	14295 (2484)	7118 (1247)	4370 (713)	8053 (1518)	21323 (2805)	11510 (1343)	6957 (725)	11061 (1586)	51018 (2688)	28675 (930)	15889 (547)	19821 (1262)
Max Time Elapsed (s)	31907	16406	8682	20995	36308	19255	10082	24049	61548	38992	22223	29329
Mean Optimality Gap	0.07 (0.01)	0.07 (0.01)	0.07 (0.01)	0.07 (0.01)	0.05 (0.01)	0.06 (0.01)	0.06 (0.01)	0.06 (0.01)	0.05 (0.01)	0.04 (0.00)	0.04 (0.01)	0.05 (0.01)
Max Optimality Gap	0.31	0.31	0.31	0.31	0.14	0.19	0.24	0.14	0.16	0.13	0.16	0.12
Mean Number of Iterations	5973 (1039)	6041 (1046)	6361 (1048)	5549 (1007)	8923 (1172)	9808 (1102)	10474 (1066)	8504 (1090)	21858 (1093)	24561 (313)	25354 (562)	17779 (885)

Appendix E: Leveraging Sparsity of $\bar{\mathbf{Q}}$

This appendix restates the smart sparse linear algebra techniques outlined in **?** with corrections made to the example that was presented previously. GMIA was enhanced to tackle problems with larger feasible regions by using sparse linear algebra techniques to compute CEI values more efficiently. Proposed in **?**, this strategy leverages the sparsity pattern of $\bar{\mathbf{Q}}$ to compute the diagonal elements of its inverse, $\boldsymbol{\Sigma}$, which proved to be particularly fruitful for use in computations involving GMRFs, as shown in **?** and was incorporated in both GMIA and rGMIA.

Suppose we have a sparse precision matrix, $\bar{\mathbf{Q}}$, corresponding to a GMRF, from which we want to compute the conditional variances of the response of the GMRF. Without leveraging the sparsity of $\bar{\mathbf{Q}}$, one could take the inverse of $\bar{\mathbf{Q}}$ and extract the required diagonal elements. However, this is expensive from both a memory and computational standpoint.

Since $\bar{\mathbf{Q}}$ is symmetric and positive definite, $\bar{\mathbf{Q}}$ has an LDL factorization. That is, there exists a lower triangular matrix, $\mathbf{L}_{\bar{\mathbf{Q}}}$, and a diagonal matrix, $\mathbf{D}_{\bar{\mathbf{Q}}}$, such that $\bar{\mathbf{Q}} = \mathbf{L}_{\bar{\mathbf{Q}}}\mathbf{D}_{\bar{\mathbf{Q}}}\mathbf{L}_{\bar{\mathbf{Q}}}^{\top}$. The sparse nature of $\bar{\mathbf{Q}}$ implies that $\mathbf{L}_{\bar{\mathbf{Q}}}$ is also relatively sparse (or can be transformed to be sparse after some number of column/row permutations to reduce fill-in). Further detailed discussion can be found in **?**. For the covariance matrix, $\boldsymbol{\Sigma} = \bar{\mathbf{Q}}^{-1}$, from which we want to extract information, **?** arrive at the following identity:

$$\boldsymbol{\Sigma} = \mathbf{D}_{\bar{\mathbf{Q}}}^{-1}\mathbf{L}_{\bar{\mathbf{Q}}}^{-1} + (\mathbf{I} - \mathbf{L}_{\bar{\mathbf{Q}}}^{\top})\boldsymbol{\Sigma} \tag{9}$$

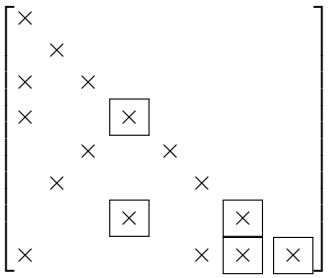
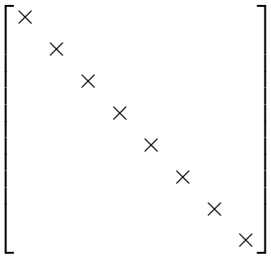
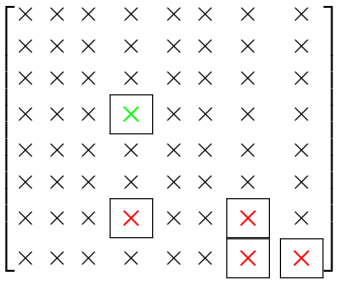
In an LDL representation, $\mathbf{L}_{\bar{\mathbf{Q}}}$ has ones on its diagonal, which implies that $\mathbf{I} - \mathbf{L}_{\bar{\mathbf{Q}}}^{\top}$ is strictly upper triangular, while $\mathbf{D}_{\bar{\mathbf{Q}}}^{-1}\mathbf{L}_{\bar{\mathbf{Q}}}^{-1}$ is lower triangular. This, combined with the fact that $\bar{\mathbf{Q}}$ (and, therefore, $\boldsymbol{\Sigma}$) is constructed to be symmetric, and Equation (**??**), yields the following result, which can be used to compute $\boldsymbol{\Sigma}_{ij}$, the element in the ith row and jth column of $\boldsymbol{\Sigma}$:

$$\boldsymbol{\Sigma}_{ij} = -\sum_{k>i} [\mathbf{L}_{\bar{\mathbf{Q}}}]_{ki} \boldsymbol{\Sigma}_{kj}, \qquad \forall i < j \tag{10}$$

$$\boldsymbol{\Sigma}_{ii} = [\mathbf{D}_{\bar{\mathbf{Q}}}]_{ii}^{-1} - \sum_{k>i} [\mathbf{L}_{\bar{\mathbf{Q}}}]_{ki} \boldsymbol{\Sigma}_{ki}, \qquad \forall i. \tag{11}$$

Since both summations in Equations (**??**) and (**??**) only contain as many summand terms as there are nonzero elements in the ith column of $\mathbf{L}_{\bar{\mathbf{Q}}}$, the number of necessary computations is greatly reduced. Notice that this strongly justifies the use of permutations of $\bar{\mathbf{Q}}$ to reduce fill-in of $\mathbf{L}_{\bar{\mathbf{Q}}}$. The method above is implemented in the PARDISO software package (**?**).

To illustrate this principle, consider a small example in which we are given an 8×8 sparse precision matrix, $\bar{\mathbf{Q}}$, together with $\mathbf{L}_{\bar{\mathbf{Q}}}$ and $\mathbf{D}_{\bar{\mathbf{Q}}}$ corresponding to its LDL decomposition. From this matrix, we wish to compute $\boldsymbol{\Sigma}_{44}$, that is, the element in the 4th row and 4th column in its inverse $\boldsymbol{\Sigma}$. Suppose $\mathbf{L}_{\bar{\mathbf{Q}}}$ and $\mathbf{D}_{\bar{\mathbf{Q}}}$ have the sparsity patterns illustrated below, where a blank space represents the value 0 and a $\times$ represents a potentially nonzero element in that position, which must be computed and stored in memory ($\times$ may be 0 as the result of computations and cancellations).

$\mathbf{L}_{\bar{\mathbf{Q}}} =$  $\mathbf{D}_{\bar{\mathbf{Q}}} =$  $\mathbf{\Sigma} =$ 

A direct approach would involve computing all elements in $\mathbf{\Sigma}$, which is dense. However, using Equations (**??**) and (**??**), we can generate the following set of equations:

$$\Sigma_{88} = [D_{\bar{\mathbf{Q}}}]_{88}^{-1} \qquad \Sigma_{87} = \Sigma_{78} = -[L_{\bar{\mathbf{Q}}}]_{87}\Sigma_{88} \qquad \Sigma_{77} = [D_{\bar{\mathbf{Q}}}]_{77}^{-1} - [L_{\bar{\mathbf{Q}}}]_{87}\Sigma_{87}$$
$$\Sigma_{74} = \Sigma_{47} = -[L_{\bar{\mathbf{Q}}}]_{74}\Sigma_{77} \qquad \Sigma_{44} = [D_{\bar{\mathbf{Q}}}]_{44}^{-1} - [L_{\bar{\mathbf{Q}}}]_{74}\Sigma_{74}.$$

Therefore, due to the sparsity pattern of $\mathbf{L}_{\bar{\mathbf{Q}}}$, we only need to compute four additional elements of $\mathbf{\Sigma}$.