Proceedings of the 2017 Winter Simulation Conference
W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.

COMPUTATIONAL METHODS FOR OPTIMIZATION VIA SIMULATION USING GAUSSIAN MARKOV RANDOM FIELDS

Mark Semelhago
Barry L. Nelson
Andreas Wächter

Department of Industrial Engineering and Management Sciences
Northwestern University
Evanston, IL 60208-3119, USA

Eunhye Song

Department of Industrial and Manufacturing Engineering
The Pennsylvania State University
University Park, PA 16802-4400, USA

ABSTRACT

There has been recent interest, and significant success, in adapting and extending ideas from statistical learning via Gaussian process (GP) regression to optimization via simulation (OvS) problems. At the heart of all such methods is a GP representing knowledge about the objective function whose conditional distribution is updated as more of the feasible region is explored. Calculating the conditional distribution requires inverting a large, dense covariance matrix, and this is the primary bottleneck for applying GP learning to large-scale OvS problems. If the GP is a Gaussian Markov Random Field (GMRF), then the precision matrix (inverse of the covariance matrix) can be constructed to be sparse. In this paper we show how to exploit this sparse-matrix structure to extend the reach of OvS based on GMRF learning for discrete-decision-variable problems.

1 INTRODUCTION

Stochastic simulation models are most often constructed to design and improve the performance of systems that are subject to uncertainty. Therefore, optimizing some measure of system performance with respect to controllable decision variables (which we will also call "feasible solutions") is a typical goal. Here we consider minimizing the expected value of some stochastic performance measure when the expected value can only be estimated via simulation. We refer to this activity as optimization via simulation (OvS), and focus on OvS problems in which the decision variables can only assume integer-ordered values. We refer to these as discrete optimization via simulation (DOvS) problems, and they abound in operations research where discrete units of resources often need to be allocated (e.g., machines in manufacturing, agents in call/contact centers, nurses in healthcare services).

When the number of feasible solutions is finite and small relative to the time required to simulate them, then theoretical and practical success has been achieved using the exhaustive search algorithms of ranking and selection (R&S). R&S simulates all feasible solutions and terminates with either a guaranteed probability of correct selection (frequentist) or posterior assessment of the relative quality of the selected solution (Bayesian). For surveys see Kim (2013) and Frazier (2012).

DOvS problems that cannot possibly be searched exhaustively present particular challenges because tools that are effective in deterministic integer programming—cuts and relaxations—have no counterparts in DOvS when the objective function can only be estimated and fractional solutions (e.g., 2.72 servers) cannot be simulated.

A setting that is more similar to DOvS is system optimization via deterministic computer experiments; see Jones et al. (1998) for the foundation paper, and Santner et al. (2003) for a general reference. The setting is similar in that the "computer experiment" is treated as a black box that can only be evaluated at specific settings of the decision variables, and each evaluation is (often) computationally expensive. Differences include a lack of noise (e.g., the computer experiment is a finite-element code that always produces the same outcome for a fixed setting), continuous-valued decision variables, and objective function evaluations that are so expensive that only a small number (by our standards) of feasible solutions will be explored.

In computer experiments, statistical learning based on modeling the unknown objective function as a Gaussian process (GP) takes the place of exploiting some known or assumed structure of the objective function or feasible region. As feasible solutions are evaluated, the conditional distribution of the GP is updated and employed to guide the search for new feasible solutions in a way that balances exploration and exploitation. This is the framework of the EGO algorithm introduced in Jones et al. (1998), and upon which we build; it has been shown again and again to be remarkably effective and robust. However, calculating the conditional distribution requires inverting a large, dense, and sometimes ill-conditioned covariance matrix, and this is the bottleneck for applying GP learning to large-scale problems.

"GP" is generic for a huge class of stochastic processes with normally distributed marginals. The choice of GP is important: every GP 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 required computations. Examples in the stochastic simulation context include Frazier et al. (2009), Frazier et al. (2011), Jalali et al. (2015), Sun et al. (2011), and Qu et al. (2012).

Recently, Salemi et al. (2014) and Salemi et al. (2017) argued that a Gaussian Markov Random Field (GMRF), appropriately parameterized, is a particularly good choice for DOvS problems. A GMRF is a GP defined on a lattice with the non-zero elements of its precision matrix determined by a user-defined neighborhood structure; we carefully define a GMRF in Section 2.1. Salemi et al. (2017) propose the Gaussian Markov Improvement Algorithm (GMIA), which is a global DOvS algorithm based on a GMRF model, and demonstrate that this model provides better learning inference than other standard choices, specifically continuous-decision-variable GPs that are evaluated only at the integer solutions. The focus of this paper is on reaping the computational benefits of using a GMRF with a sparse precision matrix.

In Section 2, we introduce the concepts of GMRF and GMIA. We also review *complete expected improvement* (CEI), which is adopted as a search and stopping guide in GMIA because of its nice features when combined with a GMRF model. However, there can be considerable computational overhead in calculating CEI. CEI is an extension of the expected improvement (EI) criterion, used in the EGO algorithm, that works well for our application. Section 3 describes how to reduce the overhead by exploiting sparse matrix computation techniques and less frequent update of the conditional distribution. We propose GMIA++, a computationally more efficient version of GMIA, in Section 4, accommodating these changes. Empirical performance of GMIA++ is analyzed in Section 5.

2 OPTIMIZATION VIA SIMULATION AND GMRFS

We consider the global DOvS problem

$$\text{minimize } y(\mathbf{x}) \triangleq \mathrm{E}[Y(\mathbf{x})] \text{ subject to } \mathbf{x} \in \mathscr{X},$$

where the feasible region $\mathscr{X}$ is a finite subset of the d-dimensional integer lattice $\mathbb{Z}^d$; let $n = |\mathscr{X}|$, the number of feasible solutions. The distribution of the random variable $Y(\mathbf{x})$, as a function of the feasible solution $\mathbf{x}$, is unknown. However, the expectation $y(\mathbf{x}) \triangleq \mathrm{E}[Y(\mathbf{x})]$ can be estimated using stochastic simulation. More formally, for any feasible solution $\mathbf{x}$ we are able to observe

$$Y_j(\mathbf{x}) = y(\mathbf{x}) + \varepsilon_j(\mathbf{x}) \tag{1}$$

on replication $j = 1, 2, \ldots$, where $\{\varepsilon_j(\mathbf{x})\}$ are i.i.d. with mean 0 and finite variance that may depend on $\mathbf{x}$.

2.1 Gaussian Markov Random Fields

Any GP-based optimization method 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 $\mathbb{Y} = (\mathbb{Y}_1, \mathbb{Y}_2, \ldots, \mathbb{Y}_n)^\top$. The particular GP process in use will determine the structure of the mean vector μ and covariance matrix Σ of $\mathbb{Y}$, and therefore the structure of its conditional distribution after some feasible solutions $\mathbf{x}$ are simulated. Often, μ and Σ are implicitly defined and approximated conditional on the simulated solutions; we take an approach to explicitly model μ and Σ. The one feature common to all GP-based methods of which we are aware is that the covariance between $\mathbb{Y}_i$ and $\mathbb{Y}_j$ decreases as some measure of distance increases.

A GMRF is a non-degenerate multivariate Gaussian random vector $\mathbb{Y}$ that is associated with an undirected and labeled graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ denotes the set of nodes and $\mathscr{E}$ denotes the set of edges; see Rue and Held (2005). Each node in $\mathscr{V}$ is associated with a unique element of $\mathbb{Y}$. Two nodes in the graph are called neighbors if they are connected by an edge. If we denote the precision matrix of $\mathbb{Y}$ by $\mathbf{Q}$, then the probability density function of the GMRF is

$$f(\mathbf{y} \mid \mu, \mathbf{Q}) = (2\pi)^{-n/2} |\mathbf{Q}|^{1/2} \exp\left(-\frac{1}{2}(\mathbf{y}-\mu)^\top \mathbf{Q}\,(\mathbf{y}-\mu)\right),$$

where $\mathbf{Q}$ is the inverse of the covariance matrix Σ. The diagonal entries of the precision matrix are the conditional precisions $\mathrm{Prec}(\mathbb{Y}_i \mid \mathbb{Y}_{\mathscr{V}\setminus\{i\}}) = \mathrm{Q}_{ii}$, where $\mathbb{Y}_{\mathscr{V}\setminus\{i\}}$ is the vector of values of the GMRF observed only at the nodes $\mathscr{V} \setminus \{i\}$. The scalar precision Q_{ii} is the reciprocal of the conditional variance. The off-diagonal elements are proportional to conditional correlations; specifically $\mathrm{Corr}(\mathbb{Y}_i, \mathbb{Y}_j \mid \mathbb{Y}_{\mathscr{V}\setminus\{i,j\}}) = -\mathrm{Q}_{ij}/\sqrt{\mathrm{Q}_{ii}\mathrm{Q}_{jj}}$, where $\mathbb{Y}_{\mathscr{V}\setminus\{i,j\}}$ is the vector of values of the GMRF observed only at the nodes $\mathscr{V} \setminus \{i,j\}$.

The graph $\mathscr{G}$ determines the non-zero pattern of the precision matrix $\mathbf{Q}$, and vice versa, since for a GMRF $\mathrm{Q}_{ij} \neq 0$ if and only if $\{i,j\} \in \mathscr{E}$. Thus, the precision matrix will be sparse if the set of edges is small. GMRFs are “Markov” because they possess the local Markov property:

$$\mathbb{Y}_i \perp \mathbb{Y}_{\mathscr{V}\setminus\{i,\mathscr{N}(i)\}} \mid \mathbb{Y}_{\mathscr{N}(i)} \quad \text{for every} \quad i \in \mathscr{V},$$

where $\mathscr{N}(i)$ is the set of neighbors of node i in $\mathscr{G}$; that is, $\mathscr{N}(i) = \{j\colon\ \{i,j\} \in \mathscr{E}\}$. This local Markov property makes sense in operations research problems; the quality of a solution can best be inferred from its immediate neighbors and the additional information from other solutions in the solution space adds little.

2.2 Optimization using GMRFs

In a DOvS problem with integer-ordered decision variables, the natural graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ defines the nodes $\mathscr{V}$ to be $\mathscr{X}$. However, the construction of $\mathscr{G}$ also requires a neighborhood. Salemi et al. (2017) show that a particularly effective choice is $\mathscr{N}(\mathbf{x}) = \{\mathbf{x}' \in \mathscr{X} : \|\mathbf{x}-\mathbf{x}'\|_2 = 1\}$, which implies that the fraction of non-zero entries in the precision matrix $\mathbf{Q}$ is bounded above by $2d/n$, which will be very small for large problems. We adopt this neighborhood structure.

We define the non-zero entries of $\mathbf{Q}$ by a function $p(\mathbf{x}, \mathbf{x}'; \theta)$, where θ is a vector of parameters; i.e., $\mathrm{Q}_{ij} \triangleq p(\mathbf{x}_i, \mathbf{x}_j; \theta)$. For the neighborhood $\mathscr{N}(\mathbf{x})$, we use $\theta = (\theta_0, \theta_1, \theta_2, \ldots, \theta_d)^\top$ and

$$p(\mathbf{x}, \mathbf{x}'; \theta) = \begin{cases} \theta_0, & \text{if } \mathbf{x} = \mathbf{x}' \\ -\theta_0\theta_j, & \text{if } |\mathbf{x}-\mathbf{x}'| = \mathbf{e}_j \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

for $\mathbf{x}, \mathbf{x}' \in \mathbb{Z}^d$, where $\mathbf{e}_j$ is the jth standard basis vector and $|\cdot|$ is component-by-component absolute value. 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. Under this parametrization $\mathbf{Q} = \mathbf{Q}(\theta)$, however, we omit θ for notational symplicity.

Since the conditional precisions must be positive, it follows that $\theta_0 > 0$. We also want neighbors to have non-negative conditional correlations, so we restrict the values of $\theta_1, \theta_2, \ldots, \theta_d$ to be non-negative as well. Furthermore, we need $\theta_j \le 1$ for $j = 1, 2, \ldots, d$, as conditional correlations must be less than one. Lastly, $\mathbf{Q}$ should be positive-definite. With this parameterization and these restrictions, $\mathbf{Q}$ is a non-singular M-matrix so its inverse is nonnegative (Johnson 1982). In other words, there are no negative unconditional correlations among nodes in the GMRF, which is a property that makes sense in many DOvS problems. Notice that even though we construct the precision matrix $\mathbf{Q}$ to be sparse, the covariance matrix it implies, $\mathbf{Q}^{-1}$, will typically be dense, as it should be.

At any iteration in the optimization, let $\Xi_2 \subseteq \mathscr{X}$ denote the subset of feasible solutions that have been simulated, and partition $\mathscr{X}$ into the two disjoint sets Ξ_2 and $\Xi_1 = \mathscr{X} \setminus \Xi_2$. Thus, Ξ_1 is the set of feasible solutions that have not yet been explored. For simplicity, we use "1" as a subscript to denote quantities associated with the set Ξ_1 and "2" as a subscript to denote quantities associated with the set Ξ_2. For instance, $n_1 = |\Xi_1|$ and $n_2 = |\Xi_2|$ are the numbers of solutions in each set. Using these disjoint sets, we can partition the vectors $\mathbf{y}, \mathbb{Y}, \mu$, and the precision matrix $\mathbf{Q}$, and write

$$\begin{pmatrix} \mathbb{Y}_1 \\ \mathbb{Y}_2 \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} \\ \mathbf{Q}_{12}^\top & \mathbf{Q}_{22} \end{pmatrix}^{-1} \right).$$

Let $\bar{\mathscr{Y}}_2$ be the vector of sample means of the simulation output at the solutions in Ξ_2. Consistent with the output model (1), we represent $\bar{\mathscr{Y}}_2$ as a realization of the GMRF $\mathbb{Y}_2^\varepsilon = \mathbb{Y}_2 + \varepsilon$, with $\mathbb{Y}_2$ and ε independent, and $\varepsilon \sim \mathrm{N}(\vec{\mathbf{0}}_{n_2 \times 1}, \mathbf{Q}_\varepsilon^{-1})$, where $\mathbf{Q}_\varepsilon$ is the intrinsic precision matrix of the noise inherent to the stochastic simulation output $\bar{\mathscr{Y}}_2$. If we simulate design points independently, then $\mathbf{Q}_\varepsilon$ is a diagonal matrix, whereas if we simulate with common random numbers (CRN), then $\mathbf{Q}_\varepsilon$ is a dense matrix; to preserve sparsity we do not employ CRN. The values in $\mathbf{Q}_\varepsilon$ also depend on how many replications have been averaged, which need not be the same at all design points. In Salemi et al. (2017) we prove the following:

Theorem 1 The conditional distribution of $\mathbb{Y}$ given $\mathbb{Y}_2^\varepsilon = \bar{\mathscr{Y}}_2$ is

$$\mathrm{N}\left(\mu + \bar{\mathbf{Q}}^{-1} \begin{pmatrix} \vec{\mathbf{0}}_{n_1 \times 1} \\ \mathbf{Q}_\varepsilon(\bar{\mathscr{Y}}_2 - \mu_2) \end{pmatrix}, \bar{\mathbf{Q}}^{-1} \right), \tag{3}$$

where

$$\bar{\mathbf{Q}} \triangleq \begin{pmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} \\ \mathbf{Q}_{12}^\top & \mathbf{Q}_{22} \end{pmatrix} + \begin{pmatrix} \mathbf{0}_{n_1 \times n_1} & \mathbf{0}_{n_1 \times n_2} \\ \mathbf{0}_{n_1 \times n_2}^\top & \mathbf{Q}_\varepsilon \end{pmatrix}$$

is the conditional precision matrix, and $\mathbf{0}_{n_i \times n_j}$ is the $n_i \times n_j$ matrix of zeros.

The conditional distribution (3) is the key to learning-based optimization. Notice that it involves the inverse of the precision matrix $\bar{\mathbf{Q}}$, which will change as we simulate additional feasible solutions, and that the sparsity of $\bar{\mathbf{Q}}$ is inherited from the sparsity of $\mathbf{Q}$ and $\mathbf{Q}_\varepsilon$. *Efficiently calculating quantities that depend on (3) for large numbers of feasible solutions n is the topic of this paper.*

Remark: In practice, parameters such as θ and μ are not known; they are usually estimated via maximum likelihood after simulating an initial set of feasible solutions, and possibly updated as the search progresses. The intrinsic precision matrix $\mathbf{Q}_\varepsilon$, on the other hand, is often directly estimated from simulation output data by using the sample variances at simulated solutions.

The GMIA searches a discrete solution space effectively and efficiently based on inference from the underlying GMRF model. A strength of GMIA is its ability to stop further search when it finds a solution whose optimality gap is less than or equal to user defined tolerance $\delta > 0$.

Let us denote the conditional mean and variance-covariance matrix in (3) at the tth iteration of GMIA as $\mathbf{M}^t$ and $\bar{\mathbf{Q}}^t$, respectively. We define $\mathbf{Q}^t_\varepsilon$ and $\bar{\mathscr{Y}}^t_2$, similarly. At each iteration, GMIA selects the solution with the smallest sample mean as the current optimal solution, $\tilde{\mathbf{x}}^t$, and uses it as an anchor to evaluate other feasible solutions, $\mathscr{X}\backslash\tilde{\mathbf{x}}^t$. The complete expected improvement (CEI) measures how much improvement in the objective function is expected to be made by choosing $\mathbf{x} \in \mathscr{X}\backslash\tilde{\mathbf{x}}^t$ as the optimal solution instead of $\tilde{\mathbf{x}}^t$ under Distribution (3). Given the simulation output $\bar{\mathscr{Y}}^t_2$, the conditional joint distribution of $\mathbb{Y}(\tilde{\mathbf{x}}^t)$ and $\mathbb{Y}(\mathbf{x})$ is bivariate normal with parameters from the rows and columns of Distribution (3) corresponding to $\tilde{\mathbf{x}}^t$ and $\mathbf{x}$. By denoting the conditional means by $M^t(\tilde{\mathbf{x}}^t)$ and $M^t(\mathbf{x})$, the conditional variances by $V^t(\tilde{\mathbf{x}}^t)$ and $V^t(\mathbf{x})$, and the conditional covariance by $C^t(\tilde{\mathbf{x}}^t,\mathbf{x})$, we can write the variance of the difference $\mathbb{Y}(\tilde{\mathbf{x}}^t) - \mathbb{Y}(\mathbf{x})$ as

$$V^t(\tilde{\mathbf{x}}^t,\mathbf{x}) \triangleq V^t(\tilde{\mathbf{x}}^t) + V^t(\mathbf{x}) - 2C^t(\tilde{\mathbf{x}}^t,\mathbf{x}). \tag{4}$$

Then, the CEI of $\mathbf{x}$ is defined as

$$\begin{aligned}\mathrm{CEI}^t(\mathbf{x}) &= \mathrm{E}[\max(\mathbb{Y}(\tilde{\mathbf{x}}^t) - \mathbb{Y}(\mathbf{x}),0)] \\ &= \left(M^t(\tilde{\mathbf{x}}^t) - M^t(\mathbf{x})\right)\Phi\left(\frac{M^t(\tilde{\mathbf{x}}^t) - M^t(\mathbf{x})}{\sqrt{V^t(\tilde{\mathbf{x}}^t,\mathbf{x})}}\right) + \sqrt{V^t(\tilde{\mathbf{x}}^t,\mathbf{x})}\,\phi\left(\frac{M^t(\tilde{\mathbf{x}}^t) - M^t(\mathbf{x})}{\sqrt{V^t(\tilde{\mathbf{x}}^t,\mathbf{x})}}\right),\end{aligned} \tag{5}$$

where ϕ and Φ are the density and cumulative distribution function, respectively, of a standard normal random variable.

GMIA uses CEIs of solutions for both search guidance and the stopping decision. If $\max_{\mathbf{x}\in\mathscr{X}\backslash\tilde{\mathbf{x}}^t} \mathrm{CEI}^t(\mathbf{x}) \leq \delta$, then it implies that the largest expected improvement from $\tilde{\mathbf{x}}^t$ we can make is $\leq \delta$, which means that we have obtained the target optimality gap, and therefore can stop further search. Otherwise, GMIA selects the solution with the largest CEI, $\mathbf{x}^t_{\mathrm{CEI}} \triangleq \arg\max_{\mathbf{x}\in\mathscr{X}\backslash\tilde{\mathbf{x}}^t} \mathrm{CEI}^t(\mathbf{x})$, to simulate next.

The computationally expensive calculation for determining the $\mathrm{CEI}^t(\mathbf{x})$ lies in computing conditional variances $V^t(\mathbf{x})$ for all $\mathbf{x}$, which correspond to the diagonal elements of $(\bar{\mathbf{Q}}^t)^{-1}$. In the next section, we discuss obtaining these elements without fully inverting $\bar{\mathbf{Q}}^t$ at each iteration.

3 COMPUTATIONALLY EFFICIENT CALCULATION OF CEI

Much of the power of GMIA has been drawn from a sparse parameterization of the precision matrix, $\bar{\mathbf{Q}}^t$, rather than working with its dense inverse, the covariance matrix, $\Sigma^t = (\bar{\mathbf{Q}}^t)^{-1}$. Computing the entire covariance matrix to calculate the conditional variances required for the CEI calculation would negate much of the benefit from using the sparse precision matrix. However, the CEI does not require all elements of Σ^t. In fact, all we need are a) the diagonal elements of Σ^t, which correspond to $V(\mathbf{x}), \forall\mathbf{x}$, and b) the column of Σ^t that corresponds to $\tilde{\mathbf{x}}^t$ whose elements are $C(\tilde{\mathbf{x}}^t,\mathbf{x}), \forall\mathbf{x} \in \mathscr{X}$. We denote this column by $\Gamma(\tilde{\mathbf{x}}^t)$.

We can obtain $\Gamma(\tilde{\mathbf{x}}^t)$ by simply solving the system $\bar{\mathbf{Q}}^t\Gamma(\tilde{\mathbf{x}}^t) = \mathbf{e}(\tilde{\mathbf{x}}^t)$ where $\mathbf{e}(\tilde{\mathbf{x}}^t)$ is the elementary vector consisting of 0s and a single 1 in the row corresponding to the position of $\tilde{\mathbf{x}}^t$. An efficient way to solve a system of linear equations, $\mathbf{AZ} = \mathbf{B}$ for $\mathbf{Z}$, given symmetric positive definite matrix $\mathbf{A}$, is to *backsolve*, which we represent by '$\backslash\!\backslash$' in the rest of the paper (e.g. $\mathbf{Z} = \mathbf{A}\backslash\!\backslash\mathbf{B}$). The backsolve operation first finds a lower triangular matrix $\mathbf{L}$ and a diagonal matrix $\mathbf{D}$ such that $\mathbf{A} = \mathbf{LDL}^\top$ (LDL factorization), and then sequentially solves $\mathbf{LZ}_1 = \mathbf{B}$ and $\mathbf{L}^\top\mathbf{Z} = \mathbf{D}^{-1}\mathbf{Z}_1$ by forward and backward substitutions, respectively.

More challenging, however, is extracting the diagonal of Σ^t in an efficient manner. In the original GMIA, Salemi et al. (2017) solve $\bar{\mathbf{Q}}^t\backslash\!\backslash\mathbf{I}$ at each iteration to obtain the full inverse, Σ^t, and extract the diagonal elements. This computation takes about $O(n^2)$ for sparse $\bar{\mathbf{Q}}^t$, which can be burdensome when n is large. Instead of obtaining the entire inverse, Takahashi et al. (1973) propose a much more efficient method that computes the elements of Σ^t necessary to calculate its diagonal only, which has been used more recently in the context of GP modeling in Vanhatalo and Vehtari (2012). In the following, we discuss this method in the context of GMRF.

Since Σ^t is, by definition, positive definite and symmetric, its inverse, $\bar{\mathbf{Q}}^t$ is as well, implying there exists a unique LDL factorization of $\bar{\mathbf{Q}}^t$. That is, there exists a lower triangular $\mathbf{L}$, with ones on its diagonal, and a diagonal $\mathbf{D}$, such that $\bar{\mathbf{Q}}^t = \mathbf{L}\mathbf{D}\mathbf{L}^T$. The sparse nature of $\bar{\mathbf{Q}}^t$ implies that $\mathbf{L}$ 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 Golub and Van Loan (2013). Takahashi et al. (1973) arrive at the following identity:

$$\Sigma^t = \mathbf{D}^{-1}\mathbf{L}^{-1} + (\mathbf{I} - \mathbf{L}^\top)\Sigma^t. \tag{6}$$

Recall that since $\mathbf{L}$ has ones on its diagonal from the LDL factorization, $\mathbf{I} - \mathbf{L}^T$ is strictly upper triangular, while $\mathbf{D}^{-1}\mathbf{L}^{-1}$ is lower triangular. Combined with the fact that both $\bar{\mathbf{Q}}^t$ and Σ^t are symmetric matrices, this identity results in the more illuminating relationship for a given element in the ith row and jth column of Σ^t:

$$\Sigma_{ij} = \sum_{k>i} L_{ki}\Sigma_{kj} \quad \forall i < j, \tag{7}$$

$$\Sigma_{ii} = D_{ii}^{-1} - \sum_{k>i} L_{ki}\Sigma_{ki}. \tag{8}$$

For a sufficiently sparse factorization there are significant savings, since both summations only contain as many summand terms as there are nonzero elements in the ith column of $\mathbf{L}$. Note that this strongly justifies the use of permutations and reorderings of $\bar{\mathbf{Q}}^t$ to reduce the fill-in of $\mathbf{L}$. This method is implemented in the PARDISO software package, the user guide for which can be found in Schenk and Gärtner (2014).

As an example, suppose that $\mathbf{L}$, $\mathbf{D}$ and Σ are 8×8 matrices and have the following sparsity patterns, where $\times$ represents nonzero elements. It is important to note that, in practice, nonzero elements (represented by $\times$) may in fact contain the value 0, which may result through certain computations and cancellations. However, any zero element must contain the value 0:

$$\mathbf{L} = \begin{bmatrix} \times &&&&&&& \\ & \times &&&&&& \\ \times && \times &&&&& \\ \times &&& \times &&&& \\ && \times && \times &&& \\ & \times &&&& \times && \\ &&& \times &&& \times & \\ \times &&&&& \times & \times & \times \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} \times &&&&&&& \\ & \times &&&&&& \\ && \times &&&&& \\ &&& \times &&&& \\ &&&& \times &&& \\ &&&&& \times && \\ &&&&&& \times & \\ &&&&&&& \times \end{bmatrix} \Sigma = \begin{bmatrix} \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times \end{bmatrix}. \tag{9}$$

Suppose we wish to calculate Σ_{44} on the diagonal of the covariance matrix. A naïve implementation involves computing Σ in its entirety (all 64 elements) and then extracting Σ_{44}. However, with an LDL factorization of $\bar{\mathbf{Q}}^t$, Σ_{44} can be computed in an efficient manner. From (7)–(8) and the fact that Σ is symmetric ($\Sigma_{ij} = \Sigma_{ji}$), we can generate the following system of equations:

$$\begin{aligned} \Sigma_{88} &= D_{88}^{-1} \\ \Sigma_{87} = \Sigma_{78} &= -L_{88}\Sigma_{88} \\ \Sigma_{68} &= -L_{87}\Sigma_{87} - L_{88}\Sigma_{88} \\ \Sigma_{77} &= D_{77}^{-1} - L_{87}\Sigma_{87} \\ \Sigma_{74} = \Sigma_{47} &= -L_{86}\Sigma_{68} - L_{77}\Sigma_{77} \\ \Sigma_{44} &= D_{44}^{-1} - L_{74}\Sigma_{74}. \end{aligned}$$

Therefore, to calculate Σ_{44}, rather than having to compute 64 elements, we find that only solving for 6 elements in Σ is necessary. PARDISO exploits this.

Although calculating the diagonal elements as above greatly reduces the computational effort compared to inverting Σ^t, it may still be burdensome to compute CEI for all feasible solutions at each iteration if $\mathscr{X}$ is large. Below, we suggest two methods to reduce the computational effort spent for CEI calculation.

First, we may select the solutions with the q largest CEI values, rather than the single solution with the largest CEI at each iteration. We expect this to result in slower improvement in the optimality gap, however for small q, we may save some computational overhead of CEI calculation at a small cost in performance.

Alternatively, we may delay updating $V^t(\mathbf{x})$ for p iterations, which is the bottleneck of the CEI calculation, and compute CEIs approximately. On the other hand, updating $\mathbf{M}^t$ is relatively cheap as it only involves a backsolve with respect to $\bar{\mathbf{Q}}^t$ when LDL factors of $\bar{\mathbf{Q}}^t$ are available. When we update $V^t(\mathbf{x})$ at each iteration, it is necessary to perform the factorization at each iteration. However, when we delay updating $V^t(\mathbf{x})$ for p iterations, it tends to be more efficient to update $\mathbf{M}^{t+1}$ using the factors of $\bar{\mathbf{Q}}^t$ via the following Sherman-Morrison-Woodbury identity instead of factorizing $\bar{\mathbf{Q}}^{t+1}$:

$$(\bar{\mathbf{Q}}^t + \mathbf{U}\mathbf{V}^\top)^{-1} = (\bar{\mathbf{Q}}^t)^{-1} - (\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}(\mathbf{I}_m + \mathbf{V}^\top(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U})^{-1}\mathbf{V}^\top(\bar{\mathbf{Q}}^t)^{-1}, \tag{10}$$

where $\mathbf{U}$ and $\mathbf{V}$ are $n \times m$ matrices for some positive integer m and $\mathbf{I}_m$ is the $m \times m$ identity matrix. In our case, $\mathbf{U}\mathbf{V}^\top = \bar{\mathbf{Q}}^t - \bar{\mathbf{Q}}^{t+1}$, which is a matrix with exactly two nonzero diagonal elements and zeroes elsewhere as we choose $\tilde{\mathbf{x}}^t$ and the largest CEI point at the tth iteration and update the corresponding diagonal elements of $\mathbf{Q}^t_\varepsilon$. Therefore, if we let a and b be the nonzero elements of $\mathbf{U}\mathbf{V}^\top$, then $\mathbf{U}$ can be represented as an $n \times 2$ matrix such that the first column has a and the second column has b at the corresponding elements as the sampled points and zeroes elsewhere and $\mathbf{Z}$ is an $n \times 2$ matrix with ones at the position of nonzero elements of $\mathbf{U}$ and zeros elsewhere. Hence, it is easy to see that inverting $(\mathbf{I}_m + \mathbf{V}^\top(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U})$ in (10) is cheap for $m = 2$. We can extend this trick to compute any backsolve operation with respect to $\bar{\mathbf{Q}}^t$ during the $p-1$ iterations we do not factorize $\bar{\mathbf{Q}}^t$. The number of nonzero diagonal elements of $\mathbf{U}\mathbf{V}^\top$ during these iterations is at most $2(p-1)$. In the algorithm we propose in Section 4, we use $\bar{\bar{\mathbf{Q}}}^t$ to denote the factorized precision matrix to distinguish it from $\bar{\mathbf{Q}}^t$. Note that the covariance vector, $\Gamma^t(\tilde{\mathbf{x}}^t)$, should be computed at each iteration using $\bar{\bar{\mathbf{Q}}}^t$ instead of $\bar{\mathbf{Q}}^t$ to be consistent with the approximate $V^t(\mathbf{x})$ and $V^t(\tilde{\mathbf{x}}^t)$.

However, one should exercise caution when determining termination of GMIA. Computing CEIs using approximate $V^t(\mathbf{x})$ can lead to premature termination of the algorithm. Therefore, should GMIA indicate that the termination criterion is satisfied, the exact CEIs should be computed to ensure this holds true.

In the next section, we present algorithm GMIA++ that incorporates sparse matrix techniques introduced in this section as well as the periodic update of $V^t(\mathbf{x})$. In Section 5, we examine the performance of both modifications proposed above.

4 COMPUTATIONALLY EFFICIENT GAUSSIAN MARKOV IMPROVEMENT ALGORITHM

In the following algorithm, each time a solution, $\mathbf{x}$, is selected, the algorithm simulates r replications at the solution. The user must specify the tolerance on the optimality gap, δ. Note that we use k as a counter to keep track of the cycle to update diagonal elements.

[GMIA++]

0. Generate a set of $n_0 \ll n$ initial design points. Simulate r replications for each design point and use the simulation output to calculate the maximum likelihood estimates (MLEs) for the GMRF parameters. Using the MLEs, set μ^0 and $\mathbf{Q}^0$. Set $t = 0$ and $k = p$.
1. Set $t \leftarrow t+1$. Let $\tilde{\mathbf{x}}^t$, the current sample-best solution, be the design point with the smallest sample mean. Update $\mathbf{Q}^t_\varepsilon$ and $\bar{\mathbf{Q}}^t$. If $k = p$ go to Step 2. Otherwise, go to Step 3.
2. Set $\bar{\bar{\mathbf{Q}}}^t = \bar{\mathbf{Q}}^t$. Extract the diagonal elements of $(\bar{\bar{\mathbf{Q}}}^t)^{-1}$ using the method in Section 3. Compute the conditional means
$$\mathbf{M}^t = \mu^0 + \bar{\bar{\mathbf{Q}}}^t \backslash\!\!\backslash \begin{pmatrix} \vec{\mathbf{0}}_{n_1 \times 1} \\ \mathbf{Q}^t_\varepsilon(\bar{\mathscr{Y}}^t_2 - \mu_2) \end{pmatrix}.$$
Set $k = 1$ and go to Step 4.

3. Set $\bar{\bar{\mathbf{Q}}}^t = \bar{\bar{\mathbf{Q}}}^{t-1}$. Find $\mathbf{U}$ and $\mathbf{V}$ that permit use of (10) to perform an update (ie. find $\mathbf{U}$ and $\mathbf{V}$ such that $\mathbf{U}\mathbf{V}^\top = \bar{\mathbf{Q}}^t - \bar{\bar{\mathbf{Q}}}^t$). Using the notation found in [**GMIA++**], (10) becomes:

$$(\bar{\bar{\mathbf{Q}}}^t + \mathbf{U}\mathbf{V}^\top)^{-1} = (\bar{\bar{\mathbf{Q}}}^t)^{-1} - (\bar{\bar{\mathbf{Q}}}^t)^{-1}\mathbf{U}(\mathbf{I}_m + \mathbf{V}^\top(\bar{\bar{\mathbf{Q}}}^t)^{-1}\mathbf{U})^{-1}\mathbf{V}^\top(\bar{\bar{\mathbf{Q}}}^t)^{-1}.$$

   Without repeating any calculations, compute the conditional mean:

$$\mathbf{M}^t = \mu^0 + \bar{\bar{\mathbf{Q}}}^t \backslash\!\backslash \begin{pmatrix} \vec{\mathbf{0}}_{n_1\times 1} \\ \mathbf{Q}^t_\varepsilon(\bar{\mathscr{Y}}^t_2 - \mu_2) \end{pmatrix} - \bar{\bar{\mathbf{Q}}}^t \backslash\!\backslash \mathbf{U}(\mathbf{I} + \mathbf{V}^\top \bar{\bar{\mathbf{Q}}}^t \backslash\!\backslash \mathbf{U}) \backslash\!\backslash \mathbf{V}^\top \bar{\bar{\mathbf{Q}}}^t \backslash\!\backslash \begin{pmatrix} \vec{\mathbf{0}}_{n_1\times 1} \\ \mathbf{Q}^t_\varepsilon(\bar{\mathscr{Y}}^t_2 - \mu_2) \end{pmatrix}.$$

   Set $k \leftarrow k+1$ and go to Step 4.
4. Compute $\Gamma^t(\tilde{\mathbf{x}}^t) = \bar{\bar{\mathbf{Q}}}^t \backslash\!\backslash e(\tilde{\mathbf{x}}^t)$. For each $\mathbf{x} \in \mathscr{X}$, calculate $V^t(\tilde{\mathbf{x}}^t, \mathbf{x})$ and $\mathrm{CEI}^t(\mathbf{x})$ in (4)–(5).
5. If $\max_{\mathbf{x}\in\mathscr{X}\setminus\tilde{\mathbf{x}}^t} \mathrm{CEI}^t(\mathbf{x}) \le \delta$ and $k = 1$, then go to Step 8. If $\max_{\mathbf{x}\in\mathscr{X}\setminus\tilde{\mathbf{x}}^t} \mathrm{CEI}^t(\mathbf{x}) \le \delta$ and $k > 1$, then set $k = p$ and go to Step 2. Otherwise, go to Step 6.
6. Simulate r replications at both $\tilde{\mathbf{x}}^t$ and $\mathbf{x}^t_{\mathrm{CEI}}$.
7. Update the simulation output at $\tilde{\mathbf{x}}^t$ with the new replications. If $\mathbf{x}^t_{\mathrm{CEI}}$ is a design point then update the simulation output at $\mathbf{x}^t_{\mathrm{CEI}}$ and go to Step 1. If $\mathbf{x}^t_{\mathrm{CEI}}$ is not a design point, then add $\mathbf{x}^t_{\mathrm{CEI}}$ to the set of design points, add the simulation output obtained at $\mathbf{x}^t_{\mathrm{CEI}}$ to the collection of simulation output, $\bar{\mathscr{Y}}^t_2$ and go to Step 1.
8. Return $\tilde{\mathbf{x}}^t$ as the estimated optimal solution.

Remarks: PARDISO supports an efficient backsolve operation as well as diagonal extraction. Note that one may specify the maximum number of iterations, T, and stop the algorithm when either maximum CEI$\le \delta$ or $t = T$ in Step 5. If we wish to select the $q > 1$ largest-CEI solutions to simulation on each iteration, then change Step 6 accordingly and ensure that $p = 1$.

In the next section, we compare the performance of GMIA++ and GMIA empirically.

5 EXPERIMENTS

In this section we use an (s,S) inventory problem to provide timing comparisons between GMIA and GMIA++, as well as provide proof-of-concept demonstrations of the effects of using approximate methods that further reduce computation. *When using GP-based optimization methods, comparisons that only consider the number of simulation replications expended can be misleading: streamlining or avoiding expensive updates of the conditional distribution become even more important on problems with large numbers of feasible solutions, n.* We consider both measures here.

In our version of the (s,S) inventory model, we assume that a firm is subject to periodic demand that follows a Poisson distribution. The firm waits until its inventory level falls below a lower threshold, s, at which point they reorder product to bring the inventory up to S. The objective is to minimize the expected value of cost over a fixed planning horizon with respect to (s,S). To treat the feasible region as a rectangular grid, we let the decision variables be $(x_1,x_2) = (s, S-s)$, as done in Salemi et al. (2017).

5.1 Objective Function Value Comparison of GMIA and GMIA++

We start with a quick check that our new implementation GMIA++ using smart inversion and the sparse matrix solver libraries of PARDISO is actually encoding the same algorithm as the original GMIA that uses the Matlab sparse matrix toolbox.

Figure 1 illustrates the average across 15 trials of 100 iterations of the true objective function values evaluated at the feasible solutions that GMIA and GMIA++ believe to be optimal on each iteration. The feasible region is 50×50, and for this experiment we used $q = 1$ and $p = 1$, meaning we update $\bar{Q}^t$ and simulate only the feasible solution with the largest CEI on each iteration. Common random numbers were used across algorithms, but not within runs. Modulo small differences in numerical error whose impact accumulates later in the run, it is clear GMIA and GMIA++ are the same algorithm.

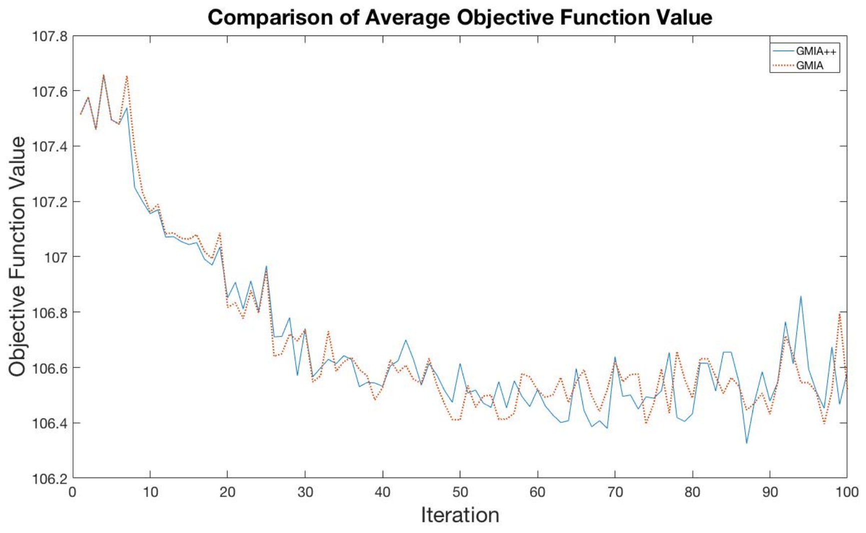

Figure 1: Comparison of GMIA and GMIA++ objective function values averaged across 15 trials.

5.2 Runtime Comparison of GMIA and GMIA++

We now compare the runtimes of GMIA and GMIA++ on the (s,S) inventory problem of three different sizes. In the first the feasible region for (x_1,x_2) is a 50×50 integer lattice, while in the second the feasible region has 100×100 integer solutions and the third is 150×150. GMIA makes use of the standard MATLAB sparse matrix toolbox, while GMIA++ makes use of the sparse matrix solver libraries offered by PARDISO. Again we used $q=1$ and $p=1$, meaning we update $\bar{Q}^t$ and simulate only the feasible solution with the largest CEI on each iteration

First we compare runtimes for the 50×50 problem, letting both algorithms run for 100 iterations, and using common random numbers across the two algorithms so that they generate common simulation output at Step 0, and as similar as possible after that; we do not use common random numbers within a run of either algorithm. Even on this relatively small-scale problem, the gains from using smart sparse linear algebra techniques are immense, with GMIA having a runtime mean, median and standard deviation of 394.83s, 394.64s and 147.21s, respectively and GMIA++ having a runtime mean, median and standard deviation of 4.53s, 3.98s and 1.01s, respectively.

The impact of our tailored sparse matrix techniques is even more dramatic as the problem becomes larger. From 15 trials of GMIA++ with a 100×100 feasible region and 50 iterations, the runtime increases in magnitude only moderately, with the mean, median and standard deviation being 9.41s, 8.72s and 0.99s, respectively. Similarly, for GMIA++ with a 150×150 feasible region and 50 iterations, the runtime mean, median and standard deviation are 22.55s, 22.80s and 1.89s. We cannot show a similar plot for GMIA as a *single* run of the 100×100 problem took 14500s (≈ 4 hours).

5.3 Top q CEI Solutions

Up to this point the precision matrix used for calculating conditional means, variances and covariances has been updated each iteration and used to allocate simulation effort to only what GMIA++ finds to be the current sample best solution and another solution with the largest CEI. A simple variation that reduces the number of matrix factorizations per solution by $1/q$ is to select the top $q>1$ CEI solutions on each iteration for additional simulation.

Figure 2 shows the true objective function averaged across 15 trials, both as a function of the number of matrix factorizations we must perform on the precision matrix and as a function of the number of replications we generate at given feasible solutions. Tracking the objective function value with respect to matrix factorizations confirms what one would expect through intuition: additional simulation effort helps GMIA++ reduce cost at a faster rate. This is true despite the tendency for these additional simulated solutions to cluster near one another. Tracking the objective function with respect to number of replications results in the objective function decreasing in a stepwise fashion, that is, after each set of q solutions is simulated (assuming $r = 10$ replications for each point).

Notice that the improvement per replication is comparable provided q is not too large, but the improvement per factorization is substantially better for $q = 5$ or $q = 10$. Thus, as the number of feasible solutions increases to the point that factorization is more significant than simulation effort, there is clear value to selecting more than one solution to explore on each iteration.

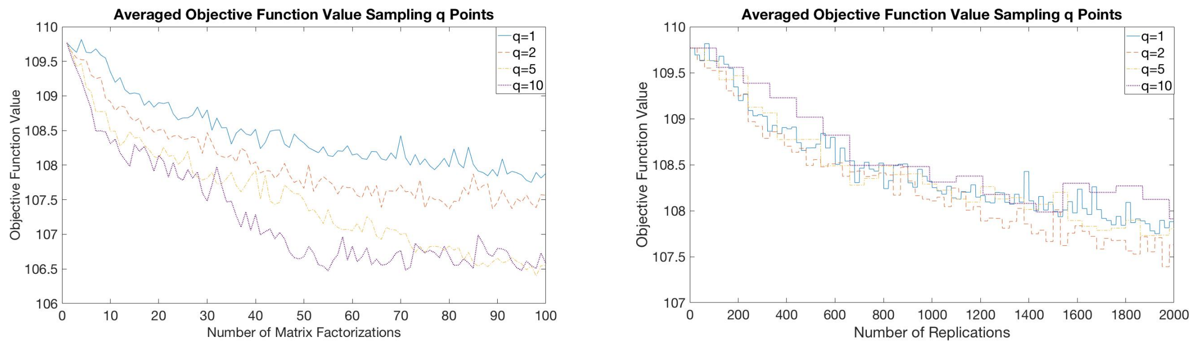

Figure 2: Comparison of GMIA++ objective function values averaged across 15 trials with respect to number of precision matrix factorizations and number of replications with q solutions simulated per iteration.

5.4 Approximate Precision Matrix

An alternative method to reduce the number of factorizations of $\bar{Q}^t$ is only to update it every $p > 1$ iterations, and to use an approximate precision matrix to calculate the $V^t(\tilde{\mathbf{x}}^t, \mathbf{x})$ in between; calculating $V^t(\tilde{\mathbf{x}}^t, \mathbf{x})$ is typically the most computationally expensive step for reasons outlined in Section 3. This reduces the number of complete factorizations per solution by $1/p$.

Figure 3 compares various update increments both per full factorization and per simulation replication expended. While $p = 2$ has comparable performance to $p = 1$ in terms of improvement per replication, larger p is inferior on this metric. However, improvement per factorization for the first 10 full factorizations clearly shows $p > 1$, and perhaps p as large as 10, is advantageous. Again, as the number of feasible solutions increases to the point that factorization is more significant than simulation effort, there is clear value to less frequent updating of $\bar{\mathbf{Q}}^t$.

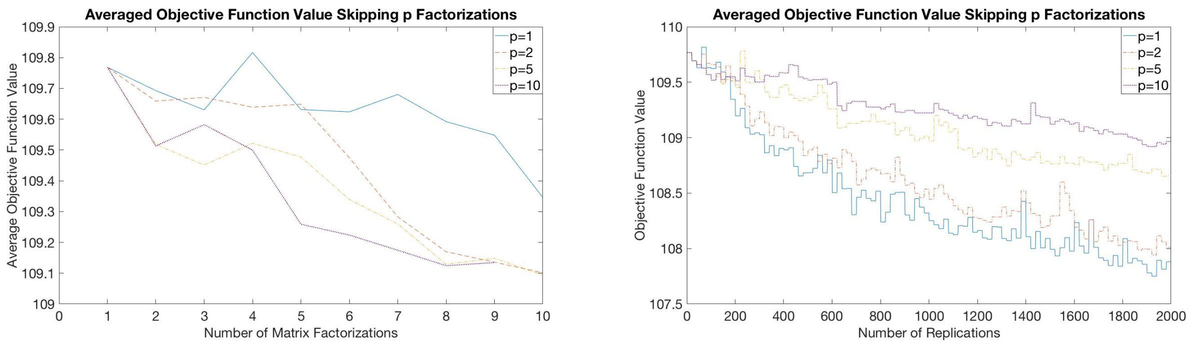

Figure 3: Comparison of GMIA++ objective function values averaged across 15 trials with respect to number of precision matrix factorizations and number of replications with p solutions simulated per iteration.

6 CONCLUSIONS

OvS based on statistical learning has shown great promise, but the conditional distribution (posterior) updates are a computational bottleneck for large problems. In this paper we have shown that tailored sparse matrix methods can signficantly reduce the computation, allowing larger problems to be solved routinely.

ACKNOWLEDGMENTS

This research was partially supported by NSF Grant No. CMMI-1537060 and co-sponsor SAS Institute.

REFERENCES

Frazier, P. 2012. “Tutorial: Optimization Via Simulation with Bayesian Statistics and Dynamic Programming”. In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, 1–16. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Frazier, P., W. Powell, and S. Dayanik. 2009. “The Knowledge-Gradient Policy for Correlated Normal Beliefs”. *INFORMS Journal on Computing* 21 (4): 599–613.

Frazier, P. I., J. Xie, and S. E. Chick. 2011. “Value of Information Methods for Pairwise Sampling with Correlations”. In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, 3979–3991. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Golub, G., and C. Van Loan. 2013. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press.

Jalali, H., I. Van Nieuwenhuyse, and V. Picheny. 2015. “Comparison of Kriging-based Methods for Simulation Optimization with Heterogeneous Noise”. Technical report, KU Leuven.

Johnson, C. R. 1982. “Inverse M-matrices”. *Linear Algebra and its Applications* 47:195–216.

Jones, D. R., M. Schonlau, and W. J. Welch. 1998. “Efficient Global Optimization of Expensive Black-Box Functions”. *Journal of Global Optimization* 13 (4): 455–492.

Kim, S. 2013. “Statistical Ranking and Selection”. In *Encyclopedia of Operations Research and Management Science*, 1459–1469. New York: Springer.

Qu, H., I. O. Ryzhov, and M. C. Fu. 2012. “Ranking and Selection with Unknown Correlation Structures”. In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, 144–155. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Rue, H., and L. Held. 2005. *Gaussian Markov Random Fields: Theory and Applications*. New York: Chapman and Hall/CRC.

Salemi, P., B. L. Nelson, and J. Staum. 2014. "Discrete Optimization via Simulation Using Gaussian Markov Random Fields". In *Proceedings of the 2014 Winter Simulation Conference*, edited by A. Tolk, S. Y. Diallo, L. Y. I. O. Ryzhov, S. Buckley, and J. A. Miller, 3809–3820. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Salemi, P., E. Song, B. L. Nelson, and J. Staum. 2017. "Gaussian Markov Random Fields for Discrete Optimization via Simulation: Framework and Algorithms". Technical report, Northwestern University.

Santner, T. J., B. J. Williams, and W. Notz. 2003. *The Design and Analysis of Computer Experiments*. New York: Springer.

Schenk, O., and K. Gärtner. 2014, February. *PARDISO User Guide Version 5.0.0*.

Sun, L., L. J. Hong, and Z. Hu. 2011. "Optimization via Simulation Using Gaussian Process-based Search". In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, 4139–4150. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Takahashi, K., J. Fagan, and M.-S. Chin. 1973. "Formation of a Sparse Bus Impedance Matrix and Its Application to Short Ciruit Study". In *8th PICA Conference Proceedings*, 16–29. IEEE Power Engineering Society.

Vanhatalo, J., and A. Vehtari. 2012. "Modelling Local and Global Phenomena with Sparse Gaussian Processes". *arXiv preprint arXiv:1206.3290*.

AUTHOR BIOGRAPHIES

MARK SEMELHAGO is a PhD candidate in the Department of Industrial Engineering and Management Sciences at Northwestern University. His research interests include simulation optimization, simulation methodology and computer experiments. His e-mail address is mark.semelhago@u.northwestern.edu.

BARRY L NELSON is the Walter P. Murphy Professor in the Department of Industrial Engineering and Management Sciences at Northwestern University and a Distinguished Visiting Scholar in the Lancaster University Management School. He is a Fellow of INFORMS and IIE. His research centers on the design and analysis of computer simulation experiments on models of stochastic systems, and he is the author of *Foundations and Methods of Stochastic Simulation: A First Course*, from Springer. His e-mail address is nelsonb@northwestern.edu.

ANDREAS WÄCHTER is an Associate Professor in the Department of Industrial Engineering and Management Sciences at Northwestern University. His research centers on the design, analysis, implementation, and application of numerical algorithms for nonlinear optimization. He is a recipient of the J. H. Wilkinson Prize for Numerical Software for the Ipopt open-source optimization package, and of the INFORMS Computing Society Award. His e-mail address is andreas.waechter@northwestern.edu.

EUNHYE SONG is an Assistant Professor in Industrial and Manufacturing Engineering at the Pennsylvania State University. Her research interests include large-scale DOvS, simulation optimization under model risk, input uncertainty quantification, and sensitivity analysis of computer experiments. Her e-mail address is eunhyesong2016@u.northwestern.edu