*Proceedings of the 2022 Winter Simulation Conference*
*B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.*

# OBJECT-ORIENTED IMPLEMENTATION AND PARALLELIZATION OF THE RAPID GAUSSIAN MARKOV IMPROVEMENT ALGORITHM

Mark Semelhago

Amazon.com
New York, NY 10018, USA

Barry L. Nelson

Department of IEMS
Northwestern University
Evanston, IL 60208, USA

Eunhye Song

School of ISyE
Georgia Institute of Technology
Atlanta, GA 30332, USA

Andreas Wächter

Department of IEMS
Northwestern University
Evanston, IL 60208, USA

## ABSTRACT

The Rapid Gaussian Markov Improvement Algorithm (rGMIA) solves discrete optimization via simulation problems by using a Gaussian Markov random field and complete expected improvement as the sampling and stopping criterion. rGMIA has been created as a sequential sampling procedure run on a single processor. In this paper, we extend rGMIA to a parallel computing environment when $q+1$ solutions can be simulated in parallel. To this end, we introduce the $q$-point complete expected improvement criterion to determine a batch of $q+1$ solutions to simulate. This new criterion is implemented in a new object-oriented rGMIA package.

## 1 INTRODUCTION

Discrete optimization via simulation (DOvS) is the practice of minimizing the expected value of a stochastic simulation model output with respect to discrete, controllable decision variables. Many algorithms that solve DOvS problems come with asymptotic convergence guarantees, but there are only a few instances of accessible and efficient software implementations of these rigorously justified algorithms. One such instance is Industrial Strength COMPASS (ISC) in Xu et al. (2010), which is an implementation of the COMPASS framework in Hong and Nelson (2006). Another example is R-SPLINE in Wang et al. (2013), which is able to asymptotically identify a local minimum, the same guarantee offered by ISC.

This paper introduces an efficient and user-friendly implementation of a powerful class of DOvS algorithms that come with proven global convergence guarantees and finite-time inference. The Gaussian Markov Improvement Algorithm (GMIA), first introduced by Salemi et al. (2014) and Salemi et al. (2019), solves DOvS problems by modelling the objective function as a Gaussian Markov random field (GMRF), a type of Gaussian process (GP) defined on a graph. GMIA uses a modified version of the expected improvement (EI) criterion from Jones et al. (1998), which is a popular sampling criterion in the Bayesian optimization literature when function evaluations are deterministic. Complete expected improvement (CEI) is an adaption of EI for use in stochastic problems. Efforts have been made to improve the computational performance of GMIA as feasible regions grow large. Salemi et al. (2019) extend GMIA to a multiresolution framework in which feasible solutions are partitioned into regions and each region is represented by a solution-level GMRF allowing for global and local search guidance, while reducing the computational overhead necessary to perform such a search. Semelhago et al. (2017) propose an efficient way to leverage sparsity in the variance-covariance matrices that characterize the GMRF to improve computational performance of GMIA. More recently, Semelhago et al. (2020) introduce a new algorithm, the rapid Gaussian Markov Improvement Algorithm (rGMIA), based on GMIA. For DOvS problems with large feasible regions, rGMIA repeatedly partitions the feasible region into so-called search and fixed sets. Between infrequent updates to the global posterior distribution of the GMRF (global search), rGMIA restricts simulation to the search set (rapid search). Due to the small cardinality of the search set, updating the posterior distribution for solutions in the search set is computationally fast, resulting in greatly improved runtime performance.

GMIA and its derivatives have assumed that simulation replications are obtained sequentially on a single processor. With the proliferation of parallel computation, there has been an effort to extend sequential algorithms to the parallel paradigm, where $q > 1$ solutions or replications can be simulated simultaneously. The popular knowledge gradient (KG) acquisition function introduced in Frazier et al. (2009) was extended to the parallel KG ($q$-KG) by Wu and Frazier (2017), in which the Bayes-optimal batch of $q$ feasible solutions is selected for simulating. Similarly, a multi-point extension of EI, known as $q$-EI, was proposed by Ginsbourger, Le Riche, and Carraro (2010). However, this quantity was typically estimated via Monte Carlo simulation. Recent work by Chevalier and Ginsbourger (2013) yields a method to compute exact $q$-EI numerically, significantly reducing the computational load involved in evaluating the criterion. In a similar fashion, we extend CEI to the parallel setting, with $q$-CEI.

The contributions of this paper are two-fold. First, we extend GMIA and CEI to the parallel setting through $q$-CEI. To do this, we prove a closed-form expression to efficiently compute the $q$-CEI of a given set of solutions and introduce a heuristic to find the subset of $q$ solutions that maximizes $q$-CEI. The second contribution is a reorganization of GMIA and rGMIA, recasting these algorithms into an object-oriented programming (OOP) framework with benefits discussed in more detail in Section 4.

The remainder of this paper is structured as follows. Section 2 provides the necessary background on GMRFs, the CEI criterion, GMIA and rGMIA. In Section 3, we introduce $q$-CEI, stating its closed-form expression for a fixed set of $q$ solutions as well as a greedy heuristic to select the $q$ solutions with the largest $q$-CEI. Section 4 provides details about implementation of the algorithms in MATLAB and Section 5 summarizes empirical results. Finally, Section 6 contains concluding remarks.

## 2 GMIA AND rGMIA

GMIA and rGMIA solve DOvS problems of the form: $\min_{\boldsymbol{x}\in\mathscr{X}} y(\boldsymbol{x}) = \min_{\boldsymbol{x}\in\mathscr{X}} \mathbb{E}[Y(\boldsymbol{x})]$, where the feasible region $\mathscr{X}$ is a finite subset of the $d$-dimensional integer lattice $\mathbb{Z}^d$; let $n = |\mathscr{X}|$ be the number of feasible solutions. In particular, it is assumed that $\mathscr{X}$ is a $d$-dimensional hyperrectangle. The objective function $y(\boldsymbol{x})$ at each feasible solution $\boldsymbol{x}$ is the unknown mean of the simulation output $Y(\boldsymbol{x})$, which can be estimated from simulation output $Y_j(\boldsymbol{x}) = y(\boldsymbol{x}) + \varepsilon_j(\boldsymbol{x})$ obtained from replication $j = 1,2,\ldots$, where $\{\varepsilon_j(\boldsymbol{x})\}$ are assumed i.i.d. normal with mean 0 and finite (unknown) variance $\sigma^2(\boldsymbol{x})$ that may depend on $\boldsymbol{x}$.

As mentioned in Section 1, GMIA and rGMIA model the objective function as a realization of a GMRF, which is a special class of GP. Specifically, a GMRF is a non-degenerate Gaussian random vector, which we represent as $\mathbb{Y}$, that is associated with an undirected and labeled graph $\mathscr{G} = (\mathscr{V},\mathscr{E})$, where $\mathscr{V}$ and $\mathscr{E}$ denote the node and edge sets, respectively; see Rue and Held (2005) for further information. In our DOvS context, a node is a feasible solution, $\boldsymbol{x} \in \mathscr{X}$, and construction of $\mathscr{E}$ assumes a neighborhood structure, $\mathscr{N}(\boldsymbol{x}) = \{\boldsymbol{x}' \in \mathscr{X} : \|\boldsymbol{x} - \boldsymbol{x}'\|_2 = 1\}$, shown by Salemi et al. (2019) to be an effective choice for DOvS. There is a one-to-one correspondence between nodes in $\mathscr{V}$ and elements of $\mathbb{Y}$ as well as a one-to-one correspondence between solutions in $\mathscr{X}$ and elements in $\mathbb{Y}$.

GMRFs are fully characterized by a mean vector, $\boldsymbol{\mu}$, and a precision matrix, $\boldsymbol{Q}$. In GMIA and rGMIA, maximum likelihood estimates (MLEs) of these quantities are computed before attempting to optimize. The structure of $\mathscr{G}$ determines the sparsity pattern of $\boldsymbol{Q}$ and vice versa in that $Q_{ij} \neq 0$ if and only if $\{i, j\} \in \mathscr{E}$. In general, the diagonal entries $Q_{ii}$ of a precision matrix are such that $Q_{ii} = 1/\mathrm{Var}(\mathbb{Y}_i|\mathbb{Y}_{\mathscr{V}\setminus\{i\}})$.

Off-diagonal elements are proportional to the conditional correlations; specifically $\mathrm{Corr}(\mathbb{Y}_i, \mathbb{Y}_j | \mathbb{Y}_{\mathscr{V}\setminus\{i,j\}}) = -Q_{ij}/\sqrt{Q_{ii}Q_{jj}}$. We define the entries of $\boldsymbol{Q}$ by a vector of parameters $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_d]^\top$. For the neighborhood $\mathscr{N}(\boldsymbol{x})$, we let $Q_{ij} = \theta_0$, if $\boldsymbol{x}_i = \boldsymbol{x}_j$ and $Q_{ij} = -\theta_0\theta_j$, if $|\boldsymbol{x}_i - \boldsymbol{x}_j| = \boldsymbol{e}_j$, where $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathscr{X}$, $\boldsymbol{e}_j$ is the $j$th standard basis vector and $|\cdot|$ is the component-wise absolute value. For all other elements, $Q_{ij} = 0$. Using this neighborhood, $\boldsymbol{Q}$ is very sparse.

As solutions are simulated, the GMRF is updated. Exploiting the inference on $y(\cdot)$ obtained from the GMRF allows us to simulate only a small fraction of the feasible region, while searching for the global optimum. Let $\bar{\boldsymbol{Y}}$ 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 $\mu$, otherwise, where $\mu$ is the prior mean. The vector $\bar{\boldsymbol{Y}}$ is modeled as a realization of the composite GMRF $\mathbb{Y}^{\boldsymbol{\varepsilon}} = \mathbb{Y} + \boldsymbol{\varepsilon}$, where the entries of $\boldsymbol{\varepsilon}$ are jointly normally distributed, if the corresponding solutions have been simulated, and 0, otherwise. The composite prior distribution of $\mathbb{Y}^{\boldsymbol{\varepsilon}}$ is $\mathrm{N}\left(\boldsymbol{\mu}, (\boldsymbol{Q} + \boldsymbol{Q}_\varepsilon)^{-1}\right)$, where $\boldsymbol{Q}_\varepsilon$ is the intrinsic precision matrix of $\boldsymbol{\varepsilon}$, which can be estimated directly from the sample variances of simulation output at simulated feasible solutions. We choose to simulate solutions independently (no common random numbers), which means $\boldsymbol{Q}_\varepsilon$ is a diagonal matrix. Given these modeling decisions, Salemi et al. (2019) prove that the GMRF metamodel has the following conditional distribution, given $\mathbb{Y}^{\boldsymbol{\varepsilon}} = \bar{\boldsymbol{Y}}$:

$$\mathbb{Y} | \mathbb{Y}^{\boldsymbol{\varepsilon}} = \bar{\boldsymbol{Y}} \sim \mathrm{N}\left(\boldsymbol{\mu} + \bar{\boldsymbol{Q}}^{-1}\boldsymbol{Q}_\varepsilon(\bar{\boldsymbol{Y}} - \boldsymbol{\mu}), \bar{\boldsymbol{Q}}^{-1}\right), \tag{1}$$

where $\bar{\boldsymbol{Q}} = \boldsymbol{Q} + \boldsymbol{Q}_\varepsilon$ is the conditional precision matrix. For further information about the use of GMRFs in modeling DOvS problems, refer to Salemi et al. (2019), Semelhago et al. (2017) or Semelhago et al. (2020).

Let $\tilde{\boldsymbol{x}}$ refer to the simulated solution with the lowest sample mean. Then the CEI of a candidate solution $\boldsymbol{x}$ relative to $\tilde{\boldsymbol{x}}$ is:

$$\mathrm{CEI}(\boldsymbol{x}; \tilde{\boldsymbol{x}}) = \mathbb{E}\left[(\mathbb{Y}(\tilde{\boldsymbol{x}}) - \mathbb{Y}(\boldsymbol{x}))^+ | \mathbb{Y}^{\boldsymbol{\varepsilon}} = \bar{\boldsymbol{Y}}\right], \tag{2}$$

where $(\cdot)^+$ is the maximum of its argument and 0. Note that the expectation in (2) is conditional on $\mathbb{Y}^{\boldsymbol{\varepsilon}} = \bar{\boldsymbol{Y}}$, the simulation output that has been collected. CEI accounts for stochasticity in $\bar{Y}(\tilde{\boldsymbol{x}})$, which was previously not fully accounted for by any other EI measure.

Let $M(\boldsymbol{x})$, $V(\boldsymbol{x})$ and $C(\tilde{\boldsymbol{x}}, \boldsymbol{x})$ denote the conditional mean and variance of $\mathbb{Y}(\boldsymbol{x})$ and conditional covariance between $\mathbb{Y}(\boldsymbol{x})$, $\mathbb{Y}(\tilde{\boldsymbol{x}})$, respectively, as specified in (1). Therefore, the conditional variance of $\mathbb{Y}(\tilde{\boldsymbol{x}}) - \mathbb{Y}(\boldsymbol{x})$ is $V(\tilde{\boldsymbol{x}}, \boldsymbol{x}) \equiv V(\tilde{\boldsymbol{x}}) + V(\boldsymbol{x}) - 2C(\tilde{\boldsymbol{x}}, \boldsymbol{x})$. Salemi et al. (2019) then show that the CEI of candidate solution $\boldsymbol{x}$ can be expressed as

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

where $\phi$ and $\Phi$ are the density and cumulative distribution functions, respectively, of a standard normal random variable.

Since GMIA adopts CEI for simulation and stopping criteria, we must compute the CEI values of all solutions, which, in turn, requires updating the conditional distribution (1) at each iteration. This involves updating the GMRF posterior at all feasible solutions; we use $\boldsymbol{x}_{\mathscr{X}}$ to denote the set of all solutions. Once CEI values have been calculated, the solutions with the largest CEI value and lowest sample mean are simulated, which we denote as $\boldsymbol{x}_{\mathrm{CEI}}$ and $\tilde{\boldsymbol{x}}$, respectively. We select $\tilde{\boldsymbol{x}}$ from already simulated solutions; the number of simulations at a solution $\boldsymbol{x} \in \mathscr{X}$ is represented as $r(\boldsymbol{x})$. To compute CEI values for all solutions, we need to compute $\boldsymbol{M}(\boldsymbol{x}_{\mathscr{X}})$, $\boldsymbol{V}(\boldsymbol{x}_{\mathscr{X}})$ and $\boldsymbol{C}(\tilde{\boldsymbol{x}}, \boldsymbol{x}_{\mathscr{X}})$, where these quantities characterize the conditional distribution in (1). However, this becomes computationally expensive for large $\mathscr{X}$.

This computational expense motivated the development of rGMIA in Semelhago et al. (2020), which introduced the idea of rapid search. Rather than computing CEI values and updating the necessary statistics at every iteration for all solutions in $\mathscr{X}$, as done in GMIA, rGMIA partitions $\mathscr{X}$ into a search set, $\mathscr{S}$, and a fixed set, $\mathscr{F}$, such that $|\mathscr{S}| \ll |\mathscr{X}|$. Search and simulation is restricted for a number of iterations to $\mathscr{S}$ and, thus, it is only necessary to compute CEI values and statistics for solutions in $\mathscr{S}$ during rapid search, and not for solutions in $\mathscr{F}$, which is relatively larger, until a rapid-search termination criterion is met (e.g., fixed number of iterations). Therefore, only the sample mean vector and intrinsic and conditional precision matrices corresponding to solutions in $\mathscr{S}$ are used in calculations, which results in faster computation. Note that the intrinsic precision matrix is directly estimated from simulation output by using the sample variances at simulated solutions. It is then used to construct the conditional precision matrix, from which one can estimate the conditional variance at a given solution via (1). We use the notation $\boldsymbol{x}_{\mathscr{S}}$ and $\boldsymbol{x}_{\mathscr{F}}$ to refer to the vectors constructed from $\boldsymbol{x}_{\mathscr{X}}$, in which elements that correspond to solutions in $\mathscr{S}$ and $\mathscr{F}$ are selected, respectively. Similar notation is used for matrices where the corresponding rows and columns are selected corresponding to solutions (e.g., $[\boldsymbol{Q}_\varepsilon]_{\mathscr{S}\mathscr{S}}$ and $\bar{\boldsymbol{Q}}_{\mathscr{F}\mathscr{F}}$ refer to the intrinsic and conditional precision matrices corresponding to solutions in $\mathscr{S}$ and $\mathscr{F}$, respectively). After the rapid-search termination criterion is met, a global search iteration updates the metamodel for all solutions in $\mathscr{X}$, computes CEI values for all solutions and repartitions $\mathscr{X}$ into a new search set, $\mathscr{S}$, and fixed set, $\mathscr{F}$, consisting of solutions that will not be simulated during rapid search. With infrequent global search iterations that require expensive computation, rGMIA is able to perform far faster search than GMIA, while maintaining the same convergence guarantees, since the global termination test is only evaluated with a fully updated metamodel. rGMIA is especially effective in tackling problems with large feasible regions, which results in large computational effort to compute summary statistics relative to the cost of simulation; refer to Algorithm 1 for a high-level description of rGMIA and Semelhago et al. (2020) for further details.

**Algorithm 1:** High-level description of rGMIA

```
1  Simulate feasible solutions and compute MLEs of GMRF parameters $(\boldsymbol{\mu},\boldsymbol{Q})$ and $\boldsymbol{Q}_\varepsilon$. Construct $\bar{\boldsymbol{Q}} = \boldsymbol{Q} + \boldsymbol{Q}_\varepsilon$ and $\bar{\boldsymbol{Y}}$;
2  Find $\tilde{\boldsymbol{x}} = \arg\min_{\{\boldsymbol{x}\in\mathscr{X}:r(\boldsymbol{x})>0\}} \bar{Y}(\boldsymbol{x})$;
3  Compute $\boldsymbol{V}(\boldsymbol{x}_{\mathscr{X}})$, $\boldsymbol{C}(\tilde{\boldsymbol{x}},\boldsymbol{x}_{\mathscr{X}})$ and $\boldsymbol{M}(\boldsymbol{x}_{\mathscr{X}})$ from $\bar{\boldsymbol{Y}}$, $\boldsymbol{Q}_\varepsilon$ and $\bar{\boldsymbol{Q}}$. Go to Step 16;
4  while global-search termination criterion not reached do
5      while rapid-search termination criterion not reached do
6          Simulate at $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}_{\text{CEI}}$ and update $\bar{\boldsymbol{Y}}$, $\boldsymbol{Q}_\varepsilon$ and $\bar{\boldsymbol{Q}}$ by incorporating the new simulation output;
7          Find $\tilde{\boldsymbol{x}} = \arg\min_{\{\boldsymbol{x}\in\mathscr{S}:r(\boldsymbol{x})>0\}} \bar{Y}(\boldsymbol{x})$;
8          Compute $\boldsymbol{V}(\boldsymbol{x}_{\mathscr{S}})$, $\boldsymbol{C}(\tilde{\boldsymbol{x}},\boldsymbol{x}_{\mathscr{S}})$ and $\boldsymbol{M}(\boldsymbol{x}_{\mathscr{S}})$, using $\bar{\boldsymbol{Y}}(\boldsymbol{x}_{\mathscr{S}})$, $[\boldsymbol{Q}_\varepsilon]_{\mathscr{S}\mathscr{S}}$, $\bar{\boldsymbol{Q}}_{\mathscr{S}\mathscr{S}}$ and intermediate matrices;
9          Calculate $\text{CEI}(\boldsymbol{x};\tilde{\boldsymbol{x}}), \forall \boldsymbol{x}\in\mathscr{S}$;
10         Find $\boldsymbol{x}_{\text{CEI}} = \arg\max_{\boldsymbol{x}\in\mathscr{S}\setminus\tilde{\boldsymbol{x}}} \text{CEI}(\boldsymbol{x};\tilde{\boldsymbol{x}})$;
11     end
12     Simulate at $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}_{\text{CEI}}$ and update $\bar{\boldsymbol{Y}}$, $\boldsymbol{Q}_\varepsilon$ and $\bar{\boldsymbol{Q}}$ by incorporating new simulation information;
13     Find $\tilde{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\in\mathscr{X}:r(\boldsymbol{x})>0} \bar{Y}(\boldsymbol{x})$;
14     Compute $\boldsymbol{V}(\boldsymbol{x}_{\mathscr{S}})$, $\boldsymbol{C}(\tilde{\boldsymbol{x}},\boldsymbol{x}_{\mathscr{S}})$ and $\boldsymbol{M}(\boldsymbol{x}_{\mathscr{S}})$, using $\bar{\boldsymbol{Y}}(\boldsymbol{x}_{\mathscr{S}})$, $[\boldsymbol{Q}_\varepsilon]_{\mathscr{S}\mathscr{S}}$, $\bar{\boldsymbol{Q}}_{\mathscr{S}\mathscr{S}}$ and intermediate matrices;
15     Compute $\boldsymbol{V}(\boldsymbol{x}_{\mathscr{F}})$, $\boldsymbol{C}(\tilde{\boldsymbol{x}},\boldsymbol{x}_{\mathscr{F}})$ and $\boldsymbol{M}(\boldsymbol{x}_{\mathscr{F}})$, using $\bar{\boldsymbol{Y}}$, $\boldsymbol{Q}_\varepsilon$, $\bar{\boldsymbol{Q}}$ and intermediate matrices;
16     Calculate $\text{CEI}(\boldsymbol{x};\tilde{\boldsymbol{x}}), \forall \boldsymbol{x}\in\mathscr{X}$;
17     Find $\boldsymbol{x}_{\text{CEI}} = \arg\max_{\boldsymbol{x}\in\mathscr{X}\setminus\tilde{\boldsymbol{x}}} \text{CEI}(\boldsymbol{x};\mathscr{X})$;
18     Construct $\{\mathscr{F},\mathscr{S}\}$ partition of $\mathscr{X}$;
19     Compute intermediate matrices;
20 end
21 Return $\tilde{\boldsymbol{x}} = \arg\min_{\{\boldsymbol{x}\in\mathscr{X}:r(\boldsymbol{x})>0\}} \bar{Y}(\boldsymbol{x})$ as the estimated optimal solution;
```

## 3 PARALLELIZATION

In this section, we discuss parallelization of rGMIA when $q+1$ solutions can be simulated simultaneously. A naïve approach would be to order solutions by their CEI values and simulate $\tilde{\boldsymbol{x}}$ and the solutions with the $q$ largest CEI values. However, this method neglects inference gained about neighboring solutions to those simulated in parallel. The Bayesian optimization literature introduced the notion of $q$-EI, a criterion that accounts for such joint effects (Ginsbourger, David and Le Riche, Rodolphe and Carraro, Laurent 2008). We adapt this idea to a stochastic simulation setting by extending CEI to $q$-CEI, focusing on the synchronous parallel setting, where the next batch of solutions can only be simulated once all solutions in the prior batch have been simulated. This approach makes sense because significant time is consumed to update the posterior, and the best $q$-CEI solutions are selected for their impact as a complete set.

We will choose a relatively modest number of solutions to parallelize, $q \leq 6$, because we have found it tends to be more effective to select a small number of good solutions to parallelize rather than a large number. However, note that this does not restrict the number of parallel processors that can be used as multiple replications of a single solution can be divided among multiple processors.

## 3.1 $q$-CEI

In the same spirit as $q$-EI, we extend the CEI criterion used in a sequential setting to a multi-point $q$-CEI, defined as $q\text{-CEI}(\Omega;\tilde{\boldsymbol{x}}) = \mathbb{E}\left[\left(\mathbb{Y}(\tilde{\boldsymbol{x}}) - \min_{\boldsymbol{x}\in\Omega}\mathbb{Y}(\boldsymbol{x})\right)^+\right]$, for a candidate set of solutions $\Omega$ relative to a current optimal solution $\tilde{\boldsymbol{x}}$. We first discuss computing the $q$-CEI of a given $\Omega$ without relying on Monte Carlo estimation. Constructing a $\Omega$ that maximizes $q$-CEI, will be discussed in Section 3.2.

To develop a closed-form expression for $q$-CEI, we follow steps similar to Chevalier and Ginsbourger (2013) for $q$-EI. We also leverage a result from Tallis (1961), which yields an expression for the mean of a coordinate in a truncated multivariate normal distribution with a general mean vector and covariance matrix. We restate that result here as Proposition 1.

**Proposition 1** Let $\mathbb{Y} = [\mathbb{Y}(\boldsymbol{x}_1), \mathbb{Y}(\boldsymbol{x}_2), \ldots, \mathbb{Y}(\boldsymbol{x}_n)]^\top$ be an $n\times 1$ Gaussian vector with mean $\boldsymbol{M} \in \mathbb{R}^n$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}$. Let $\boldsymbol{b} = [b_{\boldsymbol{x}_1}, b_{\boldsymbol{x}_2}, \ldots, b_{\boldsymbol{x}_n}]^\top \in \mathbb{R}^n$. An expectation of any coordinate $\mathbb{Y}(\boldsymbol{x}_k), 1 \leq k \leq n$ under the constraint $\mathbb{Y}(\boldsymbol{x}_j) \leq b_{\boldsymbol{x}_j}, \forall j = 1, 2, \ldots, n$, denoted by $\mathbb{Y} \leq \boldsymbol{b}$, is:

$$\mathbb{E}\left[\mathbb{Y}(\boldsymbol{x}_k)|\mathbb{Y} \leq \boldsymbol{b}\right] = M_{\boldsymbol{x}_k} - \frac{1}{\Phi_n\left(\boldsymbol{\Sigma}^{-1/2}(\boldsymbol{b}-\boldsymbol{M})\right)}\sum_{i=1}^{n}\Sigma_{\boldsymbol{x}_i,\boldsymbol{x}_k}\phi_{M_{\boldsymbol{x}_i},\Sigma_{\boldsymbol{x}_i,\boldsymbol{x}_i}}(b_{\boldsymbol{x}_i})\Phi_{n-1}\left(\boldsymbol{\Sigma}_{|\boldsymbol{x}_i}^{-1/2}\boldsymbol{c}_{\boldsymbol{x}_i}\right),$$

where $\Phi_m$ represents the $m$-dimensional standard normal c.d.f., $\phi_{\mu,\sigma^2}$ represents the normal p.d.f. with mean $\mu$ and variance $\sigma^2$, and $\boldsymbol{c}_{\boldsymbol{x}_i} \in \mathbb{R}^{(n-1)\times 1}$ is a vector where the element corresponding to solution $\boldsymbol{x}_j$ is $(b_{\boldsymbol{x}_j} - M_{\boldsymbol{x}_j}) - (b_{\boldsymbol{x}_j} - M_{\boldsymbol{x}_i})\Sigma_{\boldsymbol{x}_i,\boldsymbol{x}_j}/\Sigma_{\boldsymbol{x}_i,\boldsymbol{x}_i}$.

Here, $\boldsymbol{\Sigma}_{|\boldsymbol{x}_i}$ is the covariance matrix of $\mathbb{Y}(\boldsymbol{x}_1), \mathbb{Y}(\boldsymbol{x}_2), \ldots, \mathbb{Y}(\boldsymbol{x}_{i-1}), \mathbb{Y}(\boldsymbol{x}_{i+1}), \ldots, \mathbb{Y}(\boldsymbol{x}_n)$, conditional on $\mathbb{Y}(\boldsymbol{x}_i)$, and $\Sigma_{\boldsymbol{x}_i,\boldsymbol{x}_j}$ and $\Sigma_{\boldsymbol{x}_i,\boldsymbol{x}_i}$ are the covariance between $\mathbb{Y}(\boldsymbol{x}_i)$ and $\mathbb{Y}(\boldsymbol{x}_j)$ and variance of $\mathbb{Y}(\boldsymbol{x}_i)$, respectively. Further, for a square matrix $\boldsymbol{A}$, we use the notation $\boldsymbol{A}^{-1/2}$ to refer to the Cholesky factor of $\boldsymbol{A}^{-1}$ (i.e., the matrix $\boldsymbol{L}$ such that $\boldsymbol{L}\boldsymbol{L}^\top = \boldsymbol{A}^{-1}$). The proof of this proposition can be found in Chevalier and Ginsbourger (2013) and Tallis (1961). Proposition 1 is used to prove our main result.

**Theorem 1** Given a set of solutions $\Omega = \{\boldsymbol{x}_k\}_{k=1}^q$, and another solution, $\tilde{\boldsymbol{x}}$, define $\tilde{\Omega} = \tilde{\boldsymbol{x}} \cup \Omega$. Further, let $[\mathbb{Y}(\tilde{\boldsymbol{x}}), \mathbb{Y}(\boldsymbol{x}_1), \mathbb{Y}(\boldsymbol{x}_2), \ldots, \mathbb{Y}(\boldsymbol{x}_q)]^\top$ be a $(q+1)\times 1$ Gaussian vector with mean $\boldsymbol{M} \in \mathbb{R}^{q+1}$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{(q+1)\times(q+1)}$. Then, an expression for $q$-CEI$(\Omega;\tilde{\boldsymbol{x}})$ is

$$q\text{-CEI}(\Omega;\tilde{\boldsymbol{x}}) = \sum_{k=1}^{q}\left(-M_{\tilde{\boldsymbol{x}}}^k\Phi_q\left(-{\boldsymbol{\Sigma}^k}^{-1/2}\boldsymbol{M}^k\right) + \sum_{\boldsymbol{x}\in\tilde{\Omega}\setminus\boldsymbol{x}_k}\Sigma_{\boldsymbol{x},\tilde{\boldsymbol{x}}}^k\phi_{M_{\boldsymbol{x}}^k,\Sigma_{\boldsymbol{x},\boldsymbol{x}}^k}(0)\Phi_{q-1}\left(-{\boldsymbol{\Sigma}_{|\boldsymbol{x}}^k}^{-1/2}\boldsymbol{c}_{\boldsymbol{x}}^k\right)\right), \tag{3}$$

where $\boldsymbol{M}^k = \boldsymbol{D}^k\boldsymbol{M} = \mathbb{E}\left[\mathbb{Y}(\boldsymbol{x}_k)\mathbb{1}_{q\times 1} - \mathbb{Y}(\boldsymbol{x}_{\tilde{\Omega}\setminus\boldsymbol{x}_k})\right]$, $\boldsymbol{\Sigma}^k = \boldsymbol{D}^k\boldsymbol{\Sigma}{\boldsymbol{D}^k}^\top = \mathrm{Var}\left[\mathbb{Y}(\boldsymbol{x}_k)\mathbb{1}_{q\times 1} - \mathbb{Y}(\boldsymbol{x}_{\tilde{\Omega}\setminus\boldsymbol{x}_k})\right]$, $\boldsymbol{c}_{\boldsymbol{x}}^k = -\boldsymbol{M}_{\tilde{\Omega}\setminus\{\boldsymbol{x}_k,\boldsymbol{x}\}}^k + \frac{M_{\boldsymbol{x}}^k}{\Sigma_{\boldsymbol{x},\boldsymbol{x}}^k}\boldsymbol{\Sigma}_{\tilde{\Omega}\setminus\{\boldsymbol{x}_k,\boldsymbol{x}\},\boldsymbol{x}}^k$ and $\boldsymbol{\Sigma}_{|\boldsymbol{x}}^k = \boldsymbol{\Sigma}_{\tilde{\Omega}\setminus\{\boldsymbol{x}_k,\boldsymbol{x}\},\tilde{\Omega}\setminus\{\boldsymbol{x}_k,\boldsymbol{x}\}}^k - \frac{1}{\Sigma_{\boldsymbol{x},\boldsymbol{x}}^k}\boldsymbol{\Sigma}_{\tilde{\Omega}\setminus\{\boldsymbol{x}_k,\boldsymbol{x}\},\boldsymbol{x}}^k\boldsymbol{\Sigma}_{\boldsymbol{x},\tilde{\Omega}\setminus\{\boldsymbol{x}_k,\boldsymbol{x}\}}^k$. Here, $\boldsymbol{D}^k$ is a $q\times(q+1)$ matrix that yields all pairwise differences of each element with the $k$th element.

*Proof.* We proceed by modifying the ramp function with an indicator function, which is activated when the inner expression is non-negative. This expression is only non-negative when at least one of $\mathbb{Y}(\boldsymbol{x}_\Omega)$ is no greater than $\mathbb{Y}(\tilde{\boldsymbol{x}})$. We can decompose the indicator function into a sum of $q$ indicator functions where the $k$th indicator is activated only when $\mathbb{Y}(\boldsymbol{x}_k)$ has the lowest response among the set of responses $\mathbb{Y}(\boldsymbol{x}_{\tilde{\Omega}})$:

$$
\begin{aligned}
&\mathbb{E}\left[\left(\mathbb{Y}(\tilde{\boldsymbol{x}}) - \min_{k=1,\dots,q} \mathbb{Y}(\boldsymbol{x}_k)\right)^+\right] \\
=&\mathbb{E}\left[\left(\mathbb{Y}(\tilde{\boldsymbol{x}}) - \min_{k=1,\dots,q} \mathbb{Y}(\boldsymbol{x}_k)\right)\sum_{k=1}^{q} \mathbb{1}\left\{\mathbb{Y}(\boldsymbol{x}_k) \le \mathbb{Y}(\tilde{\boldsymbol{x}}), \mathbb{Y}(\boldsymbol{x}_k) \le \mathbb{Y}(\boldsymbol{x}_j), \forall j \neq k\right\}\right] \\
=&\sum_{k=1}^{q} \mathbb{E}\left[(\mathbb{Y}(\tilde{\boldsymbol{x}}) - \mathbb{Y}(\boldsymbol{x}_k)) | \mathbb{Y}(\boldsymbol{x}_k) \le \mathbb{Y}(\tilde{\boldsymbol{x}}), \mathbb{Y}(\boldsymbol{x}_k) \le \mathbb{Y}(\boldsymbol{x}_j), \forall j \neq k\right] P\left(\mathbb{Y}(\boldsymbol{x}_k) \le \mathbb{Y}(\tilde{\boldsymbol{x}}), \mathbb{Y}(\boldsymbol{x}_k) \le \mathbb{Y}(\boldsymbol{x}_j), \forall j \neq k\right) \\
=&\sum_{k=1}^{q} \mathbb{E}\left[(\mathbb{Y}(\tilde{\boldsymbol{x}}) - \mathbb{Y}(\boldsymbol{x}_k)) | \mathbb{Y}(\boldsymbol{x}_k) - \mathbb{Y}(\tilde{\boldsymbol{x}}) \le 0, \mathbb{Y}(\boldsymbol{x}_k) - \mathbb{Y}(\boldsymbol{x}_j) \le 0, \forall j \neq k\right] \times \qquad (4) \\
&P\left(\mathbb{Y}(\boldsymbol{x}_k) - \mathbb{Y}(\tilde{\boldsymbol{x}}) \le 0, \mathbb{Y}(\boldsymbol{x}_k) - \mathbb{Y}(\boldsymbol{x}_j) \le 0, \forall j \neq k\right).
\end{aligned}
$$

To simplify (4), we introduce some additional notation: $\mathbb{W}^k_{\boldsymbol{x}} = \mathbb{Y}(\boldsymbol{x}_k) - \mathbb{Y}(\boldsymbol{x}), k = 1, 2, \dots, q$ and $\boldsymbol{x} \in \{\boldsymbol{x}\}_{j=1}^{q} \cup \tilde{\boldsymbol{x}}$. Also, let $\mathbb{W}^k = \left[\mathbb{W}^k_{\tilde{\boldsymbol{x}}}, \mathbb{W}^k_{\boldsymbol{x}_1}, \dots, \mathbb{W}^k_{\boldsymbol{x}_{k-1}}, \mathbb{W}^k_{\boldsymbol{x}_{k+1}}, \dots, \mathbb{W}^k_{\boldsymbol{x}_q}\right]$. Note that $\mathbb{W}^k \sim \mathrm{N}\left(\boldsymbol{D}^k\boldsymbol{M}, \boldsymbol{D}^k\boldsymbol{\Sigma}\boldsymbol{D}^{k\top}\right)$. Therefore, $q$-CEI$(\Omega;\tilde{\boldsymbol{x}}) = -\sum_{k=1}^{q} \mathbb{E}\left[\mathbb{W}^k_{\tilde{\boldsymbol{x}}} | \mathbb{W}^k \le \boldsymbol{0}\right] P(\mathbb{W}^k \le \boldsymbol{0})$. We then recognize that $P(\mathbb{W}^k \le \boldsymbol{0}) = \Phi_q\left(-\boldsymbol{\Sigma}^{k^{-1/2}}\boldsymbol{M}^k\right)$ and that we can apply Proposition 1 to $\mathbb{E}\left[\mathbb{W}^k_{\tilde{\boldsymbol{x}}} | \mathbb{W}^k \le \boldsymbol{0}\right]$ to achieve the closed-form expression stated in Theorem 1. $\square$

## 3.2 Building the $q$-CEI Candidate Set $\Omega$

Constructing a set that maximizes the $q$-CEI is a challenging combinatorial optimization problem with an expensive objective function evaluation and little structural information to exploit. A greedy stepwise maximization heuristic was proposed in Chevalier and Ginsbourger (2013) to maximize $q$-EI. In this heuristic, the set $\Omega$ begins as a singleton set composed of the solution with the largest marginal EI. Then, the solution that maximizes the 2-EI, keeping the first solution in $\Omega$, is added to $\Omega$. Solutions are sequentially added in this manner until $\Omega$ consists of $q$ solutions. To approximately optimize $q$-CEI, we use this strategy of sequentially maximizing $i$-CEI, $1 \le i \le q$ with a slight modification.

*We conjecture that individual solutions in a set $\Omega$ that have a large q-CEI will, themselves, tend to have large marginal CEI values.* For example, consider having to find the set $\Omega$ that maximizes 5-CEI out of a set of 1000 solutions. It is likely that the solutions in $\Omega$ will also appear among the solutions with the, say, 100 largest marginal CEI values. If not optimal, the candidate $\Omega$ that one can construct from the 100 solutions with large marginal CEI values may act as a reasonable stand-in. With this assumption, we proceed by using the marginal CEI values, which can be computed relatively efficiently due to work in Semelhago et al. (2017), to "screen" candidate solutions that may be included in $\Omega$. We refer to this subset as the *screening set*, and denote it by $\bar{\mathscr{S}}$. Our strategy is to limit the search for a $q$-CEI optimized set to $\bar{\mathscr{S}}$, and construct $\Omega$ in a greedy, stepwise fashion.

To compute $q$-CEI$(\Omega;\tilde{\boldsymbol{x}})$ according to Theorem 1, we require the conditional mean vector $\boldsymbol{M}_{\tilde{\Omega}}$ and the conditional covariance matrix $\boldsymbol{\Sigma}_{\tilde{\Omega}\tilde{\Omega}}$ associated with solutions in $\tilde{\Omega} = \Omega \cup \tilde{\boldsymbol{x}}$. We can easily derive these quantities by selecting appropriate rows and columns from the conditional mean vector and covariance matrix for the screening set, $\boldsymbol{M}_{\bar{\mathscr{S}}}$ and $\boldsymbol{\Sigma}_{\bar{\mathscr{S}}\bar{\mathscr{S}}}$, respectively. Algorithm 2 presents pseudocode for the greedy heuristic assuming $\bar{\mathscr{S}}$ is provided with the relevant statistics (conditional mean vector, conditional covariance matrix) for solutions in this set. For the $i$th solution added to $\Omega$, we search over all candidate solutions in $\bar{\mathscr{S}}$ that have not already been included, and construct a candidate $\Omega_i$. We then compute the $i$-CEI for this candidate $\Omega_i$ according to (3). Note that the inner loop has been denoted as **(par)for** to signify the potential to parallelize this search and construction of candidate $\Omega_i$s using the processors that would be available to simulate solutions in parallel, but would otherwise be idle. We note that rGMIA computes the entire posterior distribution of the search set, including the mean vector and covariance matrix. From this, we can compute and update the posterior distribution for the screening solutions (subset of the search set) and use this to compute $q$-CEI.

**Algorithm 2:** Greedy heuristic to optimize $q$-CEI for solutions in $\bar{\mathscr{S}}$ in rGMIA

```
  Data: $\tilde{\boldsymbol{x}}, \bar{\mathscr{S}}, \boldsymbol{M}_{\bar{\mathscr{S}}}, \boldsymbol{\Sigma}_{\bar{\mathscr{S}}\bar{\mathscr{S}}}, \boldsymbol{x}_{\bar{\mathscr{S}}}$
1 Let $\Omega = \{\boldsymbol{x}_{\mathrm{CEI}}\}$ and $\tilde{\Omega} = \Omega \cup \tilde{\boldsymbol{x}}$;
2 for $i = 2, \ldots, q$ do
3     (par)for $\boldsymbol{x} \in \bar{\mathscr{S}} \setminus \tilde{\Omega}$ do
4         Calculate and store $i$-CEI$(\Omega \cup \boldsymbol{x}; \tilde{\boldsymbol{x}})$;
5     end
6     Let $\boldsymbol{x}' = \arg\max_{\boldsymbol{x} \in \bar{\mathscr{S}} \setminus \tilde{\Omega}} i$-CEI$(\Omega \cup \boldsymbol{x}; \tilde{\boldsymbol{x}})$;
7     Let $\Omega \leftarrow \Omega \cup \boldsymbol{x}'$ and $\tilde{\Omega} = \Omega \cup \tilde{\boldsymbol{x}}$;
8 end
```

## 4 IMPLEMENTATION DETAILS

MATLAB remains a popular choice of software for OvS algorithms. Here, we describe the options available to practitioners and practical considerations in using the MATLAB implementation of GMIA and rGMIA (in Section 4.1) and describe our new OOP implementation and its advantages (in Section 4.2).

### 4.1 User Input and Interaction

The goal of OvS is to optimize system performance where the system is described by a simulation. For GMIA and rGMIA, the simulation takes the form of a MATLAB function, which takes as input a $d$-dimensional integer vector solution and the number of simulation replications to perform. The simulator itself can be implemented in MATLAB, or in another language with a MATLAB wrapper. In either case, it is then stored as an anonymous function, where other parameters describing the system (e.g., arrival rates in a queueing system) have been defined by the user. The user also specifies the hyperbox feasible region by its lower and upper bounds in a $d \times 2$ matrix.

In addition to the simulator, all user-defined parameters are stored in a MATLAB structure array, which fully specifies the DOvS problem to be solved. In this structure array, the user declares whether to use GMIA or rGMIA. This choice is dependent on the nature of the DOvS problem, including such factors as the size of the feasible region and the simulation runtime. These factors dictate how much time in a given iteration of GMIA is spent in computing CEI values versus simulating solutions. Unless the number of feasible solutions and dimension of the problem is small, we recommend using rGMIA.

If using rGMIA, additional parameters, such as search set size and the number of rapid search iterations between global search iterations must also be specified. If more than a single processor is available to simulate solutions in parallel, the number of processors available and the number of screening solutions to use should be declared. If the user has available the PARDISO package for computational linear algebra then GMIA and rGMIA can take advantage of it (Schenk and Gärtner 2004). Therefore, users can also specify by a flag whether to use PARDISO instead of MATLAB's built-in linear algebra libraries. Our implementation also makes use of the Statistics and Machine Learning Toolbox, as we use `normpdf` and `normcdf` to evaluate CEI and $q$-CEI of solutions, and the Parallel Computing Toolbox if one plans to simulate multiple solutions simultaneously.

Finally, to terminate in finite time (and not simulate every solution infinitely often), users must also specify whether they want to terminate with a fixed-precision or a fixed-budget guarantee. The former will only terminate when the optimality gap (estimated by CEI) of the best solution is below a user-defined threshold and is useful when time is not a constraint. The latter will terminate after a user-defined amount of time or number of iterations, when such a constraint is present.

The software can be accessed via a GitHub repository at Semelhago et al. (2021). The only dependency outside of MATLAB and its toolboxes is the PARDISO package (Schenk and Gärtner 2004) for fast computational linear algebra, but use of this is optional.

### 4.2 OOP Implementation

There are four classes defined in our OOP implementation. The `ExperimentalDesign` object contains the design points for the initial GMRF MLE parameters, associated simulation information and resulting MLEs themselves; `GlobalMetamodel` contains all simulation information and conditional statistics about the response at all solutions in the feasible region of the problem; `RapidMetamodel` contains simulation information and conditional statistics only for solutions in a constructed search set; and `Metamodel` is an abstract superclass from which `GlobalMetamodel` and `RapidMetamodel` inherit common traits.

An OOP implementation has many advantages. First, rather than releasing two distinct software packages, this implementation recognizes that GMIA and rGMIA are instances of the same high-level framework that uses GMRFs and the CEI criterion to solve DOvS problems. The metamodel of all feasible solutions (used in GMIA and rGMIA's global search) and the metamodel of solutions in a search set (used in rGMIA's rapid search) are both conceptually GMRF metamodels that can share many properties and methods when described in code. In an OOP framework, we can use inheritance to define an abstract `Metamodel` superclass from which `GlobalMetamodel` and `RapidMetamodel` subclasses inherit common properties (e.g., response sample means and variances) and methods (e.g., compute CEI values for solutions in metamodel object). In addition to improving readability and eliminating unnecessary duplication of code, a `Metamodel` superclass makes clear the connection between GMIA and rGMIA as part of the same framework.

MATLAB categorizes classes as one of two types: value and handle. The former results in an object that contains all the data specified by the class, while the latter simply references the object. We choose to implement the `Metamodel` class and its subclasses as handle classes. This is done so that any practitioner created functions that take as input a `Metamodel` object are not in danger of creating unnecessary copies of the data, which could be expensive in both memory and computation.

In addition, by using class files to implement all numeric computations, the resulting code that implements GMIA and rGMIA becomes more easily interpretable. By using descriptive method names, any practitioner can easily follow the code by referring to the pseudocode presented in Algorithm 1.

Furthermore, this implementation improves the modularity of the code. We made many design choices, but practitioners may wish to substitute alternative methods. For example, rather than using a Latin hypercube sampling experimental design to estimate GMRF parameters, one might wish to use the generalized method of moments from Song and Dong (2018). By introducing an `ExperimentalDesign` class that contains GMRF parameter information and the simulation replication information used in generating those parameters, both GMIA and rGMIA can take as input a single standardized object that can be created and stored independently from running GMIA and rGMIA. This improves reproducibility of results, since the initial conditions of experiments can be shared via an `ExperimentalDesign` object.

To give practitioners the freedom and flexibility to modify the code to suit their needs, we use encapsulation offered in MATLAB's OOP framework to make intended alterations easier to implement. This takes the form of only providing protected write access to properties. Thus, properties can only be modified within the class or subclasses, which reduces the chance of confusing cross references and protects object information from unwanted access.

To ease the debugging process, we also leverage a recently introduced feature (R2017a) in MATLAB's OOP framework to help with error handling: property validation functions. This feature allows us to impose restrictions on properties in class definitions by accepting a potential property value as an argument before throwing an error if the value does not meet the restriction. For example, we know that CEI, by definition, must be non-negative and real. However, when matrices become ill-conditioned, there is potential for an imaginary component to be introduced to calculations, resulting in erroneous complex CEI values. Property validation functions will throw an error allowing the user to debug such issues more easily.

To compute CEI values at each iteration, we require up-to-date simulation statistics such as the response sample means and sample variances of simulated solutions. However, it is well-known that updating these statistics, especially sample variances, in an online fashion can result in numerical instability. To combat this, Welford (1962) proposes updating the sum of squares of differences from the current sample mean response and then normalizing by the number of replications to obtain the sample variance. Hence, the `Metamodel` class stores the response sum and response sum of squares for each solution across simulation replications, rather than response sample mean and variance, which are designated as *dependent* properties, meaning that they are not stored in memory but are rather computed "on-the-fly" when called by any class method. Since a major premise of the GMIA/rGMIA is that few solutions will be simulated relative to the size of the feasible region, this on-the-fly computation is cheap.

## 5 EMPIRICAL RESULTS

There are two aspects important in evaluating the performance of $q$-CEI as a criterion compared to simply choosing the top $q$ solutions with the largest marginal CEI values: improvement in search quality and computational expense incurred by determining the set $\Omega$ that maximizes $q$-CEI through the heuristic outlined in Algorithm 2. These aspects are evaluated in Sections 5.1 and 5.2, respectively.

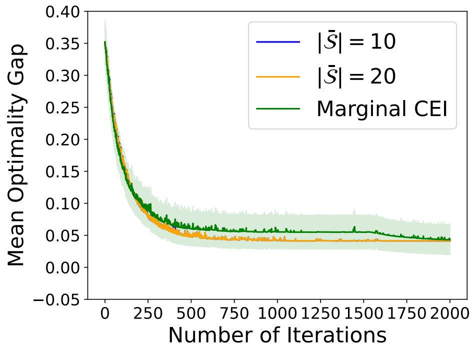
To evaluate the performance of $q$-CEI, we use a Griewank surface, modified to make the function range larger and ensure a global minimum distinguishable from other local minima; see Bingham and Surjanovic (2017) for a description of the original surface. The global minimum occurs at (0, 0) with a response value of 0. The domain of the surface is $[-5,5]\times[-5,5]$. To use this surface in a DOvS problem, we project it onto a square lattice of $401\times 401$ solutions and add independent $\mathrm{N}(0,10^{-4})$ simulation noise. This surface was used in Semelhago et al. (2020) to illustrate the power of rGMIA.

### 5.1 $q$-CEI vs Top $q$ Marginal CEI

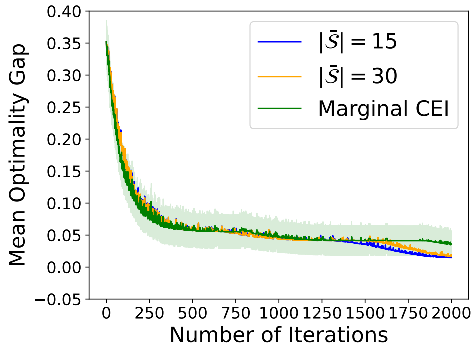
During rapid search, we compute the conditional mean vector and covariance matrix for all solutions in the search set which allows us to compute $q$-CEI for any given subset of solutions in the search set. Thus, we may choose the search set to be the screening set for Algorithm 2. We use a search set of 200 solutions and cycles of 9 rapid search iterations followed by a global search iteration. Results are aggregated across 50 macro-replications, setting different random number streams for each run. In simulating a solution, 10 replications are obtained if the solution has not been simulated before, and 2 additional replications are obtained on subsequent visits. MLEs of the GMRF parameters are estimated using a Latin hypercube sample of 20 feasible solutions, following the well-known rule of thumb of using $10d$ solutions, where $d$ is the dimension of the feasible region. Experiments are run using a high-performance computing cluster (HPCC) consisting of seven compute nodes, three with 20 cores each and four with 40 cores, and a head node with 20 cores; each node has 256GB of RAM and each core can process two threads. Experimental results were obtained using shared-memory parallelization and allocating a single thread for use by PARDISO.

We compare the performance of $q$-CEI to marginal CEI for $q=2$, 3, 4 and 5, using the greedy heuristic described in Algorithm 2 to construct the $q$-CEI optimized set using screening sets that are 5 and 10 times the size of $q$. In doing so, we illustrate the value $q$-CEI has in inducing more exploration in the optimization search. For problems with multiple local minima, such as our modified Griewank problem, this helps reduce the risk of "getting caught" in these local minima. We also show elsewhere that this exploration does not reduce performance in more "well-behaved" problems, where there is a single local (and global) minimum. We measure the mean optimality gap to evaluate performance.

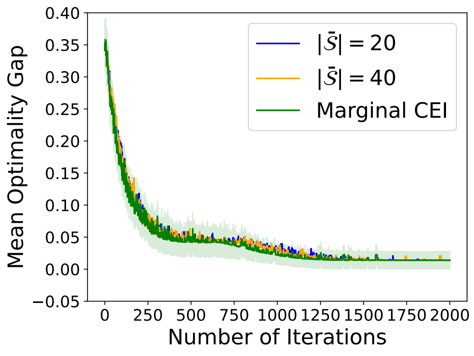
Figures 1a–1d depict the mean optimality gap versus number of iterations for 50 macro-replications of the modified Griewank problem. The green bands represent the standard error, across the 50 macro-replications, of the mean optimality gap where the top marginal CEI solutions are selected. These results indicate that $q$-CEI helps, but only conclusively when $q = 5$ and $|\bar{\mathscr{S}}| = 50$ due to the variability across macro-replications, illustrated by the large standard error. Note that the number of solutions simulated differs between the figures since at each iteration $q+1$ solutions are simulated. When this is accounted for, one can see the value in being able to simulate more solutions per iteration in exploring the feasible region. We see in Figure 1a that 6000 solutions is insufficient to find the optimal solution. Instead the plateau indicates many macro-replications getting caught in a local minimum. When $q = 3$ and $q = 4$ in Figures 1b and 1c, respectively, rGMIA gets caught in a plateau but eventually breaks out after about 6000 and 3750 solutions simulated, respectively. It is interesting to note that not only did more solutions being simulated per iteration result in faster per-iteration progress, but that additional solutions being simulated also resulted in higher quality choice of solutions being selected in subsequent iterations; hence "breaking out" of the plateau after fewer solutions simulated.



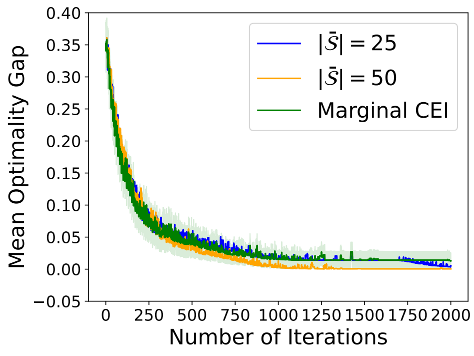
(a) $q = 2$



(b) $q = 3$



(c) $q = 4$



(d) $q = 5$

Figure 1: Mean optimality gap vs. number of iterations across 50 macro-replications for the modified Griewank problem with $q = 2,3,4,5$. The light green band represents the standard error of the estimated mean optimality gap by using marginal CEI guided search.

## 5.2 Timing of $q$-CEI Heuristic

Finding $\Omega$ becomes prohibitively expensive as $q$ and the number of candidate solutions being considered grows, thus the need for the greedy heuristic presented in Algorithm 2, and the use of screening solutions to reduce the number of candidate solutions. Nevertheless, with few processors to parallelize the `(par)for` loop in Algorithm 2, the computational time to find the $q$-CEI optimized set eclipses the time to find the $q$ solutions with the largest marginal CEI values, with the mean times being 0.0061s, 0.0062s, 0.0064s and 0.0056s for $q = 2,3,4,5$, respectively, for a search set of 200 solutions, estimated across 50 trials.

Table 1 illustrates the times to compute $q$-CEI for $q = 2, 3, 4$ and 5 using the $q+1$ processors that would be standing idle while $\Omega$ is being constructed. We varied $\bar{\mathscr{S}}$ from 10 to 200 in increments of 10 (i.e., $10, 20, 30, \ldots, 190, 200$). For each combination of $q$ and $\bar{\mathscr{S}}$, we ran 50 timed experiments. For each experiment, we randomly generated a conditional mean vector and covariance matrix of appropriate size and computed the CEI value for each solution. We then measured the wall clock time it took to construct each $q$-CEI optimized set using Algorithm 2. Parallelization of the `(par)for` loop in Algorithm 2 was implemented using the `parfor` function available in MATLAB's Parallel Computing Toolbox. Generally, for a given $q$, as $|\bar{\mathscr{S}}|$ increases, we expect the elapsed wall clock time to increase as well. Such an increase seems to be minimal for $q = 2$ and 3 based on the results in Table 1. However, as $q$ increases, we see that an increase in the number of candidate solutions results in a steeper increase in computation time, which is particularly noticeable when $q = 4$ and 5.

Based on these results and those in Section 5.1, $q$-CEI offers some improvement over simply choosing the top $q$ marginal CEI solutions but with a computational cost. Therefore, it is most valuable for very computationally expensive simulations when the total number of simulated solutions will be small.

Table 1: Mean times to compute $q$-CEI using Algorithm 2 for different screening set sizes across 50 trials, using $q+1$ processors. Standard errors of mean values are provided in parentheses.

| **Screening Set Size** ($\lvert\bar{\mathscr{S}}\rvert$) | $q=2$ | $q=3$ | $q=4$ | $q=5$ | **Screening Set Size** ($\lvert\bar{\mathscr{S}}\rvert$) | $q=2$ | $q=3$ | $q=4$ | $q=5$ |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 0.04 (0.01) | 0.07 (0.02) | 0.45 (0.02) | 1.52 (0.03) | 110 | 0.09 (0.00) | 0.24 (0.00) | 4.13 (0.10) | 21.24 (0.61) |
| 20 | 0.03 (0.00) | 0.08 (0.00) | 0.81 (0.02) | 3.55 (0.09) | 120 | 0.10 (0.00) | 0.25 (0.00) | 4.43 (0.12) | 23.13 (0.67) |
| 30 | 0.04 (0.00) | 0.10 (0.00) | 1.20 (0.03) | 5.63 (0.15) | 130 | 0.10 (0.00) | 0.27 (0.00) | 4.77 (0.12) | 24.95 (0.73) |
| 40 | 0.05 (0.00) | 0.12 (0.00) | 1.56 (0.04) | 7.39 (0.20) | 140 | 0.10 (0.00) | 0.29 (0.00) | 5.13 (0.13) | 29.10 (0.89) |
| 50 | 0.05 (0.00) | 0.15 (0.00) | 1.97 (0.05) | 9.39 (0.27) | 150 | 0.11 (0.00) | 0.30 (0.00) | 5.51 (0.14) | 32.33 (0.97) |
| 60 | 0.06 (0.00) | 0.15 (0.00) | 2.34 (0.06) | 11.18 (0.31) | 160 | 0.12 (0.00) | 0.32 (0.00) | 5.65 (0.15) | 34.61 (1.02) |
| 70 | 0.06 (0.00) | 0.17 (0.00) | 2.68 (0.07) | 13.25 (0.38) | 170 | 0.12 (0.00) | 0.33 (0.00) | 5.83 (0.15) | 35.60 (1.06) |
| 80 | 0.07 (0.00) | 0.18 (0.00) | 3.05 (0.07) | 15.64 (0.45) | 180 | 0.12 (0.00) | 0.35 (0.00) | 6.07 (0.16) | 36.38 (1.09) |
| 90 | 0.08 (0.00) | 0.20 (0.00) | 3.39 (0.08) | 17.57 (0.50) | 190 | 0.13 (0.00) | 0.38 (0.00) | 6.38 (0.16) | 38.05 (1.11) |
| 100 | 0.08 (0.00) | 0.22 (0.00) | 3.80 (0.09) | 19.41 (0.57) | 200 | 0.14 (0.00) | 0.39 (0.00) | 6.61 (0.17) | 39.32 (1.16) |

## 6 CONCLUSIONS

This paper presents a user-friendly, OOP implementation of GMIA and rGMIA as part of a state-of-the-art framework for DOvS problems and introduces $q$-CEI, a new search criterion suitable for the parallel simulation on multiple processor setting. There is evidence for $q$-CEI's ability to account for joint effects between solutions to yield better search, but additional time costs make it not always appropriate. However, simulating solutions with GMIA/rGMIA on multiple processors simultaneously is itself a novel contribution.

## ACKNOWLEDGMENTS

This work was completed prior to the affiliation of Mark Semelhago with Amazon.com and is supported by the National Science Foundation Grant Nos. DMS-1854562 and DMS-1854659.

## REFERENCES

Bingham, D. and Surjanovic, S. 2017. "Virtual Library of Simulation Experiments". https://www.sfu.ca/~ssurjano/.

Chevalier, C., and D. Ginsbourger. 2013. "Fast Computation of the Multi-points Expected Improvement with Applications in Batch Selection". In *Learning and Intelligent Optimization*, edited by G. Nicosia and P. Pardalos, 59–69. Berlin, Heidelberg: Springer Berlin Heidelberg.

Semelhago et al. 2021. "GMIA and rGMIA". https://github.com/mark-semelhago/GMIA_rGMIA.

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

Ginsbourger, David and Le Riche, Rodolphe and Carraro, Laurent 2008. "A Multi-points Criterion for Deterministic Parallel Global Optimization based on Gaussian Processes". https://hal.archives-ouvertes.fr/hal-00260579.

Ginsbourger, D., R. Le Riche, and L. Carraro. 2010. "Kriging is Well-suited to Parallelize Optimization". In *Computational Intelligence in Expensive Optimization Problems*, 131–162. New York, New York: Springer.

Hong, L. J., and B. L. Nelson. 2006. "Discrete Optimization via Simulation Using COMPASS". *Operations Research* 54(1):115–129.

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.

Rue, H., and L. Held. 2005. *Gaussian Markov Random Fields: Theory and Applications*. Boca Raton, Florida: CRC press.

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, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, 3809–3820. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Salemi, P., E. Song, B. L. Nelson, and J. Staum. 2019. "Gaussian Markov Random Fields for Discrete Optimization via Simulation: Framework and Algorithms". *Operations Research* 67(1):250–266.

Schenk, O., and K. Gärtner. 2004. "User Guide Version 6.0.0". *Journal of Future Generation Computer Systems* 20(3):475–487.

Semelhago, M., B. L. Nelson, E. Song, and A. Wächter. 2020. "Rapid Discrete Optimization via Simulation with Gaussian Markov Random Fields". *INFORMS Journal of Computing* 33(3):915–930.

Semelhago, M., B. L. Nelson, A. Wächter, and E. Song. 2017. "Computational Methods for Optimization via Simulation Using Gaussian Markov Random Fields". In *Proceedings of the 2017 Winter Simulation Conference*, edited by W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, 2080–2091. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Song, E., and Y. Dong. 2018. "Generalized Method of Moments Approach to Hyperparameter Estimation for Gaussian Markov Random Fields". In *Proceedings of the 2018 Winter Simulation Conference*, edited by M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, 1790–1801. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Tallis, G. M. 1961. "The Moment Generating Function of the Truncated Multi-normal Distribution". *Journal of the Royal Statistical Society: Series B (Methodological)* 23(1):223–229.

Wang, H., R. Pasupathy, and B. W. Schmeiser. 2013. "Integer-Ordered Simulation Optimization Using R-SPLINE: Retrospective Search with Piecewise-Linear Interpolation and Neighborhood Enumeration". *ACM Transactions on Modeling and Computer Simulation* 23(3):1–24.

Welford, B. P. 1962. "Note on a Method for Calculating Corrected Sums of Squares and Products". *Technometrics* 4(3):419–420.

Wu, J., and P. Frazier. 2017. "The Parallel Knowledge Gradient Method for Batch Bayesian Optimization". In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, edited by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, 3126–3134. Red Hook, New York: Curran Associates, Inc.

Xu, J., B. L. Nelson, and J. L. Hong. 2010. "Industrial Strength COMPASS: A Comprehensive Algorithm and Software for Optimization via Simulation". *ACM Transactions on Modeling and Computer Simulation* 20(1):1–29.

## AUTHOR BIOGRAPHIES

**MARK SEMELHAGO** is a Research Scientist at Amazon.com. He earned his PhD in the Department of Industrial Engineering and Management Sciences at Northwestern University. His research interests include simulation optimization, simulation methodology and computer experiments, with recent interest in inventory management applications. His e-mail address and website are markseme@amazon.com and https://mark-semelhago.github.io/, respectively.

**BARRY L. NELSON** is the Walter P. Murphy Professor in the Department of Industrial Engineering and Management Sciences at Northwestern University. 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 an author of *Foundations and Methods of Stochastic Simulation: A First Course, 2e* from Springer. His e-mail address and website are nelsonb@northwestern.edu and http://www.iems.northwestern.edu/~nelsonb/, respectively.

**EUNHYE SONG** is a Coca-Cola Foundation Early Career Assistant Professor in the Department of Industrial and Systems Engineering at Georgia Institute of Technology. Her research interests include simulation design of experiments, uncertainty and risk quantification, and simulation optimization. Her e-mail address is eus358@psu.edu and her website can be found at http://eunhyesong.info.

**ANDREAS WÄCHTER** is a 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.