European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/eor

Stochastics and Statistics

Dice and slice simulation optimization for high-dimensional discrete problems

Harun Avci [a,*], Barry L. Nelson [b], Eunhye Song [c], Andreas Wächter [b]

[a] Department of Economics, Finance, and Quantitative Analysis, Kennesaw State University, Kennesaw, GA, 30144, USA
[b] Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, 60208, USA
[c] H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, 30332, USA

ARTICLE INFO

Keywords:
Simulation
Large-scale discrete simulation optimization
Gaussian Markov random fields
Bayesian optimization

ABSTRACT

Although much progress has been made in simulation optimization, problems involving computationally expensive simulations having high-dimensional, *discrete* decision-variable spaces have been stubbornly resistant to solution. For this class of problems we propose Dice and Slice Simulation Optimization (DASSO). DASSO is a form of Bayesian optimization that represents the prior on the objective function implied by the simulation as a sum of low-dimensional Gaussian Markov random fields. This prior is consistent with the full-dimensional objective function, rather than assuming that it is actually separable. By working iteratively between posteriors on these low-dimensional "dice" and a full-dimensional "slice" of the decision-variable space, DASSO makes rapid progress with little algorithm overhead even on problems with more than a trillion feasible solutions. We achieve further computational savings by showing that we can find the best solution to simulate on each iteration without having to assess the potential of all solutions—as is traditionally done in Bayesian optimization—by identifying a small set of Pareto-optimal solutions in subsets of the dimensions. We prove that DASSO is asymptotically convergent to the optimal solution, while emphasizing that its most important feature is the ability to find good solutions quickly in problems beyond the capability of other methods.

1. Introduction

Simulation optimization (SO) is a technique for improving the performance of complex systems under uncertainty, where the design parameters of the simulation model represent the controllable decision variables of the system. Unlike deterministic optimization and stochastic programming problems, the objective function in SO has no explicit form and can only be estimated using simulation experiments. SO problems with discrete decision variables are known as discrete simulation optimization (DSO) problems and arise in many areas of operations research and management science.

For large-scale DSO problems, where only a small fraction of feasible solutions can reasonably be simulated, one can employ metaheuristics, adaptive random search, or inference-based algorithms. Metaheuristics typically provide no statistical performance guarantees. Adaptive random search algorithms, on the other hand, can be shown to achieve asymptotic convergence to the global or a local optimum. However, such convergence results have little practical meaning for large-scale problems if it is not possible to simulate even a modest fraction of the feasible solutions. See Fu et al. (2015) for a review of DSO algorithms.

Inference-based algorithms are closely related to Bayesian optimization (BO), a popular technique for optimizing black-box functions that is known for making rapid search progress. BO provides guidance for the search based on the conditional (posterior) distribution of a stochastic process representing the unknown objective function. A popular choice for the prior on the objective function is a Gaussian process (GP) due to its posterior being Gaussian if the simulation output distribution is also Gaussian. Although GPs with a continuous domain have been adapted to solve DSO problems (see, e.g., Quan et al. 2013, Sun et al. 2014, Xie et al. 2016), Salemi et al. (2019) demonstrate empirically that continuous-decision-variable covariance functions may fail dramatically when used for guidance and optimality-gap inference in discrete problems. They also show that using a Gaussian Markov random field (GMRF) prior, as an alternative to a continuous GP, provides much better search guidance and stopping inference for DSO. Our new methods start from the foundation of the Gaussian Markov Improvement Algorithm (GMIA) of Salemi et al. (2019) but are dramatically different.

A GMRF is a multivariate Gaussian random vector associated with an undirected graph where the nodes represent feasible solutions and the edges determine correlation structure. Unlike GPs with a continuous

* Corresponding author.
E-mail addresses: havci@kennesaw.edu (H. Avci), nelsonb@northwestern.edu (B.L. Nelson), esong32@gatech.edu (E. Song), andreas.waechter@northwestern.edu (A. Wächter).

https://doi.org/10.1016/j.ejor.2026.01.005
Received 22 April 2025; Accepted 5 January 2026
Available online 10 January 2026

domain, GMRFs can be naturally defined on a lattice, making it more suitable for DSO problems. The correlation structure is defined by a precision matrix, the inverse of the covariance matrix, where the nonzero elements correspond to edges in the defining, and ideally sparse, graph. The GMRF-based GMIA algorithm evaluates the complete expected improvement (CEI) of each feasible solution on each iteration to guide the search. CEI predicts each solution's improvement in the objective function relative to the current sample-best solution based on the posterior distribution of the GMRF (Salemi et al. 2019, Semelhago et al. 2021).

Computing the full posterior distribution at every iteration of GMIA requires inversion of the precision matrix, a computationally expensive calculation. Fortunately, computing the CEI of each feasible solution requires only the diagonal and a single column of the covariance matrix (i.e., the inverse of the precision matrix) of the posterior. While the elements of the required column can be obtained by a direct back-solve, extracting the diagonal elements is computationally more challenging. To address this, Semelhago et al. (2017) propose an efficient way to extract the diagonal elements by exploiting the sparse structure of the precision matrix. They further speed up GMIA by employing a divide-and-conquer strategy to restrict the search within a small promising subset of feasible solutions for several iterations (Semelhago et al. 2021). Alternatively, Li and Song (2020) apply the Sherman-Morrison-Woodbury formula recursively, updating only the necessary elements of the covariance matrix until it is no longer cheaper than inversion.

These computational improvements greatly extend the reach of GMIA-based DSO. Nevertheless, even these variants encounter a limit on the problem size they can attack, for instance when the decision-variable dimension is high, since the computational cost of calculating the posterior distribution increases at least quadratically in the number of feasible solutions. To reach significantly beyond this limit, we propose the Dice and Slice Simulation Optimization (DASSO) algorithm that decomposes the prior distribution into a rigorously justified additive form via a functional analysis of variance (FANOVA). In this decomposition, the first-order terms represent independent GMRFs and the higher-order terms form a random effect with a chosen first-order term. The decomposition reduces the problem dimension to facilitate efficient posterior updates, moving high-dimensional DSO problems that are too large to solve with current technology into the realm of possibility. Our numerical analysis reveals that DASSO can obtain good feasible solutions rapidly on problems with more than a trillion feasible solutions, far beyond the reach of any other algorithm.

Of course, no single SO algorithm is appropriate for all problems. Our particular interest is in stochastic DSO problems with integer-ordered decision variables, very large numbers of feasible solutions, and a simulation that itself is so computationally expensive that one only expects to simulate a tiny fraction of the solution space, even given substantial time and high-performance computing. For such problems rapid progress toward better and better solutions, iteration by iteration, is essential. Although we do prove that DASSO is "convergent," this is more of academic than practical interest for the class of problems on which we focus. Examples of such computationally prohibitive problems include the design of a new fuel injector production line described in Tongarlak et al. (2010), where 20 replications of a single feasible solution took 8 hours, and a "future mobility" problem that we worked on in collaboration with an automotive manufacturer, where each replication of a single feasible solution took a few hours. Computationally less expensive, yet still large-scale examples include the bike-sharing system modeled as a discrete-event simulation in Jian et al. (2016), and a multi-product inventory problem with an (s, S) policy for each product, where s and S are the reorder and order-up-to levels, respectively. We use this inventory problem for our numerical analysis and as a running example throughout the paper.

In summary, we propose the DASSO algorithm, a type of BO for solving large-scale, computationally expensive DSO problems whose feasible solutions are defined on a finite subset of a high-dimensional integer lattice. The key contributions are as follows: DASSO exploits an innovative representation that decomposes the prior on the objective function into an additive form, reducing the problem dimensionality to facilitate computationally efficient posterior updates, but without losing a full-dimensional representation as other additive decompositions do. Furthermore, DASSO achieves rapid search progress by identifying the best-CEI solution to simulate on each iteration while avoiding the computational overhead of actually computing the CEI of all solutions. In brief, DASSO makes large-scale DSO computationally efficient without sacrificing what makes BO so effective in small-scale problems, facilitating the solution of far larger problems than can be attacked by any competing BO algorithm.

The remainder of this article is organized as follows: Section 2 reviews the literature on high-dimensional BO and DSO. Section 3 briefly defines GMRFs and shows how they can be used to solve DSO problems. Section 4 explains how FANOVA justifies the prior distribution representation that underlies the DASSO algorithm. Empirical performance evaluations are in Section 5. Finally, conclusions are provided in Section 6. All proofs and many derivations are in the e-companion.

2. Literature review

In their review paper, Binois and Wycoff (2022) classify the solution tactics for high-dimensional BO into three categories based on the structural model assumptions made to deal with the curse of dimensionality. Worth noting is that much of this literature addresses machine-learning problems or deterministic computer experiments, rather than stochastic simulation.

Under the assumption that only a subset of the decision variables are active, the first approach is to reduce the problem dimension by removing the decision variables that have little or no impact on the objective function. By doing so, a lower-dimensional problem with only influential decision variables can be solved instead. Such a lower-dimensional problem can be obtained by initially performing variable selection, if no expert knowledge is available. However, variable selection itself is more computationally challenging as the dimension increases. Also, when all decision variables have similar influence on the objective function, no such reduction is possible. Even when it can be done, the reduced-dimension problem may still be relatively high dimensional depending on the actual number of influential decision variables.

Assuming the existence of a lower-dimensional active subspace, the second approach is to find a projection of the decision variables to obtain an active subspace, then apply BO. Since such an active subspace, if it exists, is unknown, the projection must be estimated. One can parameterize the projection scheme (e.g., a projection matrix) and estimate the parameters (see, e.g., Garnett et al. 2014, Tripathy et al. 2016), obtain the projection from a sensitivity analysis (see, e.g., Djolonga et al. 2013), or sample it randomly (see, e.g., Wang et al. 2016). Yet it is critical (especially, when such an active subspace does not exist) to figure out which feasible solution to simulate next in an optimization search because that decision involves projecting back to the full space, which is needed to parameterize the simulation. Such a projection back may be inefficient when a large part of the space is infeasible. Moreover, the computational savings from a projection-based approach relies heavily on the size of the active subspace's dimension (Mathesen et al. 2019).

Under the assumption of an additive structure for the objective function, the third approach is to decompose the objective function into a sum of functions, each with fewer decision variables. The motivation is to perform the optimization component-wise, making the search much more efficient, and thus addressing the computational burden of high-dimensional BO. Several papers in this stream assume that the objective function is a sum of independent low-dimensional functions with disjoint (separable) decision variable dimensions (see, e.g., Gardner et al. 2017, Kandasamy et al. 2015, Wang et al. 2018), while others allow possibly overlapping dimensions (see, e.g., Hoang et al. 2018, Rolland et al. 2018). They differ mainly in how they learn the additive structure. They all model the lower-dimensional functions as realizations of

independent GPs. This implies that the prior on the objective function values is a GP with an additive kernel. A potential downside of the additivity assumption is that it implies that the interactions among the subsets of decision variables (which we refer to as *groups*) are assumed to be negligible. Further, the covariance matrix of the additive GP may not be invertible due to linear relationships among the solutions, which is more likely to occur when the feasible solution space is discrete (Durrande et al. 2011).

Our DASSO methodology is superficially related to this last approach in terms of decomposing the objective function, but it does not enforce a separability approximation. Instead, it employs discrete FANOVA to include *all* high-order interactions among the groups by iteratively employing two steps: The "dice" step avoids the computational burden of the high-order interactions by forming a random-effect that combines the high-order terms with one of the lower-dimensional groups, and then selects a partial solution based on its posterior. Next, DASSO obtains a best-CEI solution from the posterior of the partial solution "slice," a posterior that has the same (low) dimension as the hold-out group and that fully accounts for all higher-order interactions. Details are in Section 4.

Although FANOVA has been used mostly as a basis for global sensitivity analysis, some studies (see, e.g., Ginsbourger et al. 2016, Muehlenstaedt et al. 2012, Ulaganathan et al. 2016) focus on high-dimensional GP modeling on a continuous domain and provide uncertainty measures over predictions or more computationally efficient GP models. Our aim is different in that we solve SO problems over a discrete domain.

To our knowledge, the only other inference-based algorithm that is designed for high-dimensional DSO problems is the projected Gaussian Markov improvement algorithm (pGMIA) of Li and Song (2024); see also Mes et al. (2011). pGMIA partitions the dimensions into two groups, called "region" and "solution" layers, and projects the latter onto the former. The region layer is represented as a graph, where the objective function value at a node is the average of objective function values at the feasible solutions projected onto that node. pGMIA alternates between choosing a node in the region-layer graph and deciding the next feasible solution to simulate from the solutions projected onto the chosen node. For both decisions, it models the objective function values as a GMRF and adopts CEI as the criterion to choose which feasible solution to simulate next. Li and Song (2024) also describe pGMIA+, a multi-layer extension of pGMIA, and both algorithms outperform the state-of-the-art high-dimensional BO algorithms to which they compare. Therefore, pGMIA represents the state of the art against which we compare in Section 5.

3. Discrete simulation optimization with GMRFs

Our problem to minimize $\boldsymbol{y}(\boldsymbol{x}) = \mathrm{E}[Y(\boldsymbol{x})]$ subject to $\boldsymbol{x} \in \mathcal{X}$, where $\mathcal{X}$ is a finite subset of the d-dimensional integer lattice treated as a collection of (column) vectors. The objective, $\boldsymbol{y}(\boldsymbol{x})$, can only be estimated via stochastic simulation. In particular, the output $Y_j(\boldsymbol{x}) = \boldsymbol{y}(\boldsymbol{x}) + \epsilon_j(\boldsymbol{x})$ can be observed for $\boldsymbol{x}$ on replication $j = 1, 2, \dots$, where $\{\epsilon_j(\boldsymbol{x})\}$ are independent and identically distributed with mean zero and finite (unknown) variance $\sigma^2(\boldsymbol{x})$ that may depend on $\boldsymbol{x}$. In the multi-product inventory problem, $\boldsymbol{x}$ denotes the vector of reorder and order-up-to values for each product, $Y(\boldsymbol{x})$ represents the average cost per period obtained from a single replication, and $\boldsymbol{y}(\boldsymbol{x})$ is the true expected average cost per period. We define GMRFs and explain how they have been used to solve the DSO problem in Sections 3.1–3.2, deferring our approach to Section 4.

3.1. Gaussian Markov random field (GMRF)

A GMRF is a multivariate Gaussian random vector $\mathbb{Y} = [\mathbb{Y}_1, \mathbb{Y}_2, \dots, \mathbb{Y}_n]^\top$ associated with an undirected labeled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of n nodes and $\mathcal{E}$ is the set of edges (Rue & Held 2005). In a DSO setting, each node represents a solution $\boldsymbol{x}_i$ associated with the corresponding element of $\mathbb{Y}$. Moreover, we impose the structure that solutions that differ by ± 1 in one coordinate are connected by edges. Letting $\boldsymbol{\mu}$ and $\boldsymbol{Q}$ be the mean vector and precision matrix of $\mathbb{Y}$, respectively, the probability density function of the GMRF can be written as $f(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{Q}) = (2\pi)^{-n/2}|\boldsymbol{Q}|^{1/2}\exp\left(-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu})^\top \boldsymbol{Q}(\boldsymbol{y}-\boldsymbol{\mu})\right)$. The precision matrix is the inverse of the covariance matrix and is positive-definite. The choice of $\mathcal{E}$ makes $\boldsymbol{Q}$ have no more than $(2d+1)n$ nonzero elements. The diagonal elements of $\boldsymbol{Q}$ are the conditional precisions: $Q_{ii} = \mathrm{Prec}(\mathbb{Y}_i \mid \mathbb{Y}_{\mathcal{V}\setminus\{i\}})$, where $\mathbb{Y}_S$ denotes the subvector of $\mathbb{Y}$ including only the nodes in any $S \subset \mathcal{V}$. The off-diagonal entries are proportional to conditional correlations; specifically, $Q_{ij} = -\mathrm{Corr}(\mathbb{Y}_i, \mathbb{Y}_j \mid \mathbb{Y}_{\mathcal{V}\setminus\{i,j\}})\sqrt{Q_{ii}Q_{jj}}$.

A GMRF $\mathbb{Y}$ is Markovian in the sense that it has the local Markov property: $\mathbb{Y}_i \perp \mathbb{Y}_{\mathcal{V}\setminus(\mathcal{N}(i)\cup\{i\})} \mid \mathbb{Y}_{\mathcal{N}(i)}$, where $\mathcal{N}(i) = \{j : \{i,j\} \in \mathcal{E}\}$ is the set of neighbors of node i in $\mathcal{G}$. This implies that the random variable $\mathbb{Y}_i$, conditional on the values of its neighbors, is independent of the values at non-neighboring nodes. Consequently, $Q_{ij} \neq 0$ if and only if $\{i,j\} \in \mathcal{E}$.

3.2. Gaussian Markov improvement algorithm (GMIA)

Since the vector of objective function values, denoted by $\boldsymbol{y} = [\boldsymbol{y}(\boldsymbol{x}_1), \boldsymbol{y}(\boldsymbol{x}_2), \dots, \boldsymbol{y}(\boldsymbol{x}_n)]^\top$, is unknown, it can be modeled as a realization of the GMRF $\mathbb{Y} = \left[\mathbb{Y}(\boldsymbol{x}_1), \mathbb{Y}(\boldsymbol{x}_2), \dots, \mathbb{Y}(\boldsymbol{x}_n)\right]^\top \sim \mathrm{N}\left(\boldsymbol{\mu}, \boldsymbol{Q}^{-1}\right)$ with mean vector $\boldsymbol{\mu}$ of size $n \times 1$ and precision matrix $\boldsymbol{Q}$ of size $n \times n$; this is the prior.

Suppose that only the "design points," a subset of the feasible solutions, have been simulated. Let $D \subseteq \mathcal{X}$ denote the current subset of design points, and partition $\mathcal{X}$ into the two disjoint sets D and $\mathcal{U} = \mathcal{X} \setminus D$. Notice that D is the set of feasible solutions that have been simulated, and $\mathcal{U}$ is the set of feasible solutions that have not (i.e., are unsimulated). Using these disjoint sets, the vector $\mathbb{Y}$ can be partitioned into $\mathbb{Y}_\mathcal{U}$ and $\mathbb{Y}_D$, which are the subvectors of $\mathbb{Y}$ including the solutions in $\mathcal{U}$ and D, respectively, that is, $\mathbb{Y} = \begin{pmatrix} \mathbb{Y}_\mathcal{U} \\ \mathbb{Y}_D \end{pmatrix}$. Similarly, the vector $\boldsymbol{\mu}$ can be partitioned into $\boldsymbol{\mu}_\mathcal{U}$ and $\boldsymbol{\mu}_D$.

The sample mean vector of simulated values at the design points, denoted by $\bar{\boldsymbol{Y}}_D$, can be represented as a realization of $\mathbb{Y}_D^\epsilon = \mathbb{Y}_D + \boldsymbol{\epsilon}$, with $\mathbb{Y}_D$ and $\boldsymbol{\epsilon}$ independent, and $\boldsymbol{\epsilon} \sim \mathrm{N}\left(\vec{\boldsymbol{0}}_{|D|}, \boldsymbol{\Sigma}^\epsilon\right)$, where $\vec{\boldsymbol{0}}_{|D|}$ is a $|D|$-dimensional vector of zeros and $\boldsymbol{\Sigma}^\epsilon$ is the intrinsic covariance matrix of the stochastic noise inherent to $\bar{\boldsymbol{Y}}_D$. When the design points are simulated independently, $\boldsymbol{\Sigma}^\epsilon$ is a diagonal matrix, whereas it is a dense matrix when the design points are simulated with common random numbers; due to the expense of inverting dense matrices we opt for independent simulations.

The conditional distribution of $\mathbb{Y}$ given $\mathbb{Y}_D^\epsilon = \bar{\boldsymbol{Y}}_D$ is

$$\mathrm{N}\left(\begin{pmatrix} \boldsymbol{\mu}_\mathcal{U} \\ \boldsymbol{\mu}_D \end{pmatrix} + \bar{\boldsymbol{Q}}^{-1}\begin{pmatrix} \vec{\boldsymbol{0}}_{|\mathcal{U}|} \\ [\boldsymbol{\Sigma}^\epsilon]^{-1}(\bar{\boldsymbol{Y}}_D - \boldsymbol{\mu}_D) \end{pmatrix}, \bar{\boldsymbol{Q}}^{-1}\right), \quad \text{where } \bar{\boldsymbol{Q}} = \boldsymbol{Q} + \begin{pmatrix} \boldsymbol{0}_{|\mathcal{U}|\times|\mathcal{U}|} & \boldsymbol{0}_{|\mathcal{U}|\times|D|} \\ \boldsymbol{0}_{|D|\times|\mathcal{U}|} & [\boldsymbol{\Sigma}^\epsilon]^{-1} \end{pmatrix} \tag{1}$$

is the conditional precision matrix and $\boldsymbol{0}_{a\times b}$ is the $a \times b$ matrix of zeros. The unsimulated solutions are first in the GMRF representation because we assume the solutions have been so ordered.

Let $\tilde{\boldsymbol{x}}$ denote the current sample-best solution (among the design points in D) based on sample means, i.e., $\tilde{\boldsymbol{x}} = \mathrm{argmin}_{\boldsymbol{x}\in D}\bar{\boldsymbol{Y}}_D(\boldsymbol{x})$, where $\bar{\boldsymbol{Y}}_D(\boldsymbol{x})$ is the element of $\bar{\boldsymbol{Y}}_D$ associated with $\boldsymbol{x}$. Then, the CEI of $\boldsymbol{x} \in \mathcal{X}$ relative to $\tilde{\boldsymbol{x}}$ is $\mathrm{CEI}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) = \mathrm{E}\left[\max\{\mathbb{Y}(\tilde{\boldsymbol{x}}) - \mathbb{Y}(\boldsymbol{x}), 0\} \mid \mathbb{Y}_D^\epsilon = \bar{\boldsymbol{Y}}_D\right]$. Let $\boldsymbol{m}(\boldsymbol{x})$ and $\boldsymbol{v}(\boldsymbol{x})$ denote the conditional mean and conditional variance of $\mathbb{Y}(\boldsymbol{x})$, respectively, and $\boldsymbol{c}(\tilde{\boldsymbol{x}}, \boldsymbol{x})$ denote the conditional covariance between $\mathbb{Y}(\tilde{\boldsymbol{x}})$ and $\mathbb{Y}(\boldsymbol{x})$. Then the conditional variance of the difference $\mathbb{Y}(\tilde{\boldsymbol{x}}) - \mathbb{Y}(\boldsymbol{x})$ is $\boldsymbol{v}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) = \boldsymbol{v}(\tilde{\boldsymbol{x}}) + \boldsymbol{v}(\boldsymbol{x}) - 2\boldsymbol{c}(\tilde{\boldsymbol{x}}, \boldsymbol{x})$. Further, the CEI of $\boldsymbol{x} \in \mathcal{X} \setminus \{\tilde{\boldsymbol{x}}\}$ can be expressed as

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

where ϕ and Φ are the standard normal probability density and cumulative distribution functions, respectively (Salemi et al. 2019). At each

iteration, GMIA simulates the sample-best solution, $\tilde{\boldsymbol{x}}$, and the solution with the largest CEI, then updates the posterior distribution and continues. CEI is an enhancement of the "expected improvement" criterion for optimizing deterministic black-box functions (Jones et al. 1998) that has been shown to be particularly effective in stochastic simulation (see also Chen and Ryzhov 2019). All else being equal, CEI$(\tilde{\boldsymbol{x}}, \boldsymbol{x})$ is decreasing in $\boldsymbol{m}(\boldsymbol{x})$ and increasing in $\boldsymbol{v}(\tilde{\boldsymbol{x}}, \boldsymbol{x})$, which is the property DASSO exploits to significantly improve computational efficiency as described in Section 4.6.

3.3. Intermezzo

A DSO algorithm for large-scale, expensive simulation optimization needs to be both search-effective and computationally feasible. The search-effectiveness of BO depends on the choice of GP prior, particularly its covariance structure. Hidden in the GMIA summary above is a sparse parameterization for $\boldsymbol{Q}$ that has been shown empirically to be exceptional for guiding the search in DSO problems (Li & Song 2020, 2024, Salemi et al. 2019, Semelhago et al. 2021, 2017) as well as computationally advantageous; see Section EC.5 in the e-companion. Clearly this formulation is something we do not want to lose.

However, even though the form of $\boldsymbol{Q}$ in GMIA is sparse, evaluating the posterior distribution, and even calculating the CEIs for every feasible solution on every iteration, becomes computationally prohibitive if the number of feasible solutions is (say) in the trillions. Therefore, GMIA and its variants are constrained by the size of the problem, particularly its dimension. To overcome this limitation, we propose the DASSO algorithm that has a search-effective yet computationally feasible covariance structure while avoiding the need to calculate the CEI of every feasible solution on each iteration, all without approximations.

4. Dice and slice simulation optimization (DASSO)

Recall that the aim is to minimize $\boldsymbol{y}(\boldsymbol{x})$ subject to $\boldsymbol{x} \in \mathcal{X}$, where $\mathcal{X}$ is a finite subset of the d-dimensional integer lattice, and the objective can only be estimated via stochastic simulation. The set of feasible solutions can be represented as $\mathcal{X} = \bigtimes_{i=1}^{d} \mathcal{X}_i$, where $\mathcal{X}_i$ is a finite subset of the one-dimensional integer lattice with $n_i = |\mathcal{X}_i|$. The number of feasible solutions is $n = \prod_{i=1}^{d} n_i$ which increases exponentially in d; thus, only a small fraction of feasible solutions can be simulated for high-dimensional problems with expensive simulations even if $|\mathcal{X}_i|$ is small. In the following sections, we show how DASSO tackles this problem.

4.1. Discrete FANOVA

The following Discrete FANOVA (DFA) representation of the objective function justifies the prior distribution at the heart of DASSO: Suppose we assign probabilities to each feasible solution for the purpose of measuring the global sensitivity in terms of variance of the output response with respect to solutions' component dimensions. Specifically, assume that the component dimensions of a feasible solution are independent and follow discrete uniform distributions. That is, letting $\mathbb{X} = [\mathbb{X}_1, \mathbb{X}_2, \dots, \mathbb{X}_d]^\top$ denote a random vector, we have $\mathrm{P}\{\mathbb{X} = \boldsymbol{x}\} = \prod_{i=1}^{d} \mathrm{P}\{\mathbb{X}_i = x_i\}$, $\forall \boldsymbol{x} = [x_1, x_2, \dots, x_d]^\top \in \mathcal{X}$, and $\mathrm{P}\{\mathbb{X}_i = x\} = 1/n_i$, $\forall x \in \mathcal{X}_i$, for each component dimension $\mathbb{X}_i$ of $\mathbb{X}$, where x_i denotes the ith component of $\boldsymbol{x}$. Therefore,

$$\mathrm{P}\{\mathbb{X} = \boldsymbol{x}\} = \prod_{i=1}^{d} \mathrm{P}\{\mathbb{X}_i = x_i\} = \frac{1}{\prod_{i=1}^{d} n_i} = \frac{1}{n}, \ \forall \boldsymbol{x} = [x_1, x_2, \dots, x_d]^\top \in \mathcal{X}. \tag{2}$$

Thus, all feasible solutions are equally likely, or equally weighted in a non-probabilistic sense.

For any subset $\mathfrak{u}$ of component indices $\mathfrak{d} = \{1, 2, \dots, d\}$, let $\boldsymbol{x}_\mathfrak{u} = (x_i)_{i \in \mathfrak{u}}$ denote the lower dimensional component of $\boldsymbol{x}$ associated with $\mathfrak{u}$. The number and set of lower dimensional components associated with $\mathfrak{u}$ are $n_\mathfrak{u} = \prod_{i \in \mathfrak{u}} n_i$ and $\mathcal{X}_\mathfrak{u} = \bigtimes_{i \in \mathfrak{u}} \mathcal{X}_i$, respectively. Also, let $-\mathfrak{u}$ denote the complementary set $\mathfrak{d} \setminus \mathfrak{u}$. In the DFA decomposition, we first define the grand mean to be $\boldsymbol{y}_\emptyset(\boldsymbol{x}) = \frac{1}{n} \sum_{\boldsymbol{x} \in \mathcal{X}} \boldsymbol{y}(\boldsymbol{x}) \triangleq \bar{y}, \ \forall \boldsymbol{x} \in \mathcal{X}$. Notice that $\boldsymbol{y}_\emptyset(\boldsymbol{x})$ is a constant that does not depend on any component dimensions of $\boldsymbol{x}$. Then, for $\mathfrak{u} \subseteq \mathfrak{d}$, the effect of $\boldsymbol{x}_\mathfrak{u}$ can be expressed as $\boldsymbol{y}_\mathfrak{u}(\boldsymbol{x}) = \frac{1}{n_{-\mathfrak{u}}} \sum_{\boldsymbol{x}_{-\mathfrak{u}} \in \mathcal{X}_{-\mathfrak{u}}} \big(\boldsymbol{y}(\boldsymbol{x}) - \sum_{\mathfrak{v} \subset \mathfrak{u}} \boldsymbol{y}_\mathfrak{v}(\boldsymbol{x})\big), \ \forall \boldsymbol{x} \in \mathcal{X}$. Notice that $\boldsymbol{y}_\mathfrak{u}(\boldsymbol{x})$ depends on $\boldsymbol{x}$ only through $\boldsymbol{x}_\mathfrak{u}$ as the effect of $\boldsymbol{x}_{-\mathfrak{u}}$ is averaged out. Using these effects, $\boldsymbol{y}(\boldsymbol{x})$ can be represented as a DFA decomposition: $\boldsymbol{y}(\boldsymbol{x}) = \sum_{\mathfrak{u} \subseteq \mathfrak{d}} \boldsymbol{y}_\mathfrak{u}(\boldsymbol{x})$. Under the probability measure in (2), we can show that these functions have certain properties.

Proposition 1. *The following properties hold if $\mathbb{X}$ follows Distribution (2):*

(a) $\mathrm{E}[\boldsymbol{y}(\mathbb{X})] = \bar{y}$,
(b) $\mathrm{E}\big[\boldsymbol{y}_\mathfrak{u}(\mathbb{X})\big] = 0$, $\forall \mathfrak{u} \subseteq \mathfrak{d}$ *with* $\mathfrak{u} \neq \emptyset$,
(c) $\mathrm{E}\big[\boldsymbol{y}_\mathfrak{u}(\mathbb{X})\boldsymbol{y}_\mathfrak{v}(\mathbb{X})\big] = 0$, $\forall \mathfrak{u}, \mathfrak{v} \subseteq \mathfrak{d}$ *with* $\mathfrak{u} \neq \mathfrak{v}$,
(d) $\mathrm{Var}[\boldsymbol{y}(\mathbb{X})] = \sum_{\mathfrak{u} \subseteq \mathfrak{d}} \mathrm{Var}\big[\boldsymbol{y}_\mathfrak{u}(\mathbb{X})\big]$.

Proposition 1 states that (a) the overall mean of $\boldsymbol{y}(\mathbb{X})$ is $\bar{y}$; (b) other than $\boldsymbol{y}_\emptyset(\boldsymbol{x})$, each $\boldsymbol{y}_\mathfrak{u}(\mathbb{X})$ in the DFA decomposition has zero mean; (c) $\boldsymbol{y}_\mathfrak{u}(\mathbb{X})$ and $\boldsymbol{y}_\mathfrak{v}(\mathbb{X})$ for $\mathfrak{u} \neq \mathfrak{v}$ are uncorrelated; and (d) the overall variance of $\boldsymbol{y}(\mathbb{X})$ decomposes into the sum of the variances of each individual function. Proposition 1 holds for any $\boldsymbol{y}(\cdot)$ regardless of its functional form, and it motivates the DASSO prior distribution on $\mathbb{Y}$.

4.2. Decomposition

Partition $\mathfrak{d}$ into $g \geq 1$ disjoint non-empty sets $\mathfrak{g}^{(1)}, \mathfrak{g}^{(2)}, \dots, \mathfrak{g}^{(g)}$, i.e., $\bigcup_{\rho \in \mathcal{G}} \mathfrak{g}^{(\rho)} = \mathfrak{d}$ and $\mathfrak{g}^{(\rho)} \cap \mathfrak{g}^{(\varrho)} = \emptyset$ for $\rho \neq \varrho$, where $\mathcal{G} = \{1, 2, \dots, g\}$. These sets represent "groups" indexed by the superscripts in parentheses. For instance, in the multi-product inventory problem, groups can represent subsets of products. The indexing of the groups is updated throughout the algorithm. Later we will single out group g for special treatment.

Throughout the paper, ρ and ϱ index groups. For each $\rho \in \mathcal{G}$, let $\mathcal{X}^{(\rho)}$ and $\boldsymbol{x}^{(\rho)}$ represent the corresponding components of $\mathcal{X}$ and $\boldsymbol{x}$, respectively, indexed by $\mathfrak{g}^{(\rho)}$, i.e., $\mathcal{X}^{(\rho)} = \bigtimes_{i \in \mathfrak{g}^{(\rho)}} \mathcal{X}_i$ and $\boldsymbol{x}^{(\rho)} = (x_i)_{i \in \mathfrak{g}^{(\rho)}}$. Notice that $\mathcal{X}^{(\rho)}$ is a finite subset of the $d^{(\rho)}$-dimensional integer lattice with $n^{(\rho)}$ distinct points, where $d^{(\rho)} = |\mathfrak{g}^{(\rho)}|$ and $n^{(\rho)} = |\mathcal{X}^{(\rho)}| = \prod_{i \in \mathfrak{g}^{(\rho)}} n_i$. Further, notice that $\sum_{\rho \in \mathcal{G}} d^{(\rho)} = d$ and $n = \prod_{\rho \in \mathcal{G}} n^{(\rho)}$. Fig. 1 illustrates a decomposition into two groups for a simple 3-dimensional example with $n = 12$ solutions.

For each group $\rho \in \mathcal{G}$, we define the *group function* $\boldsymbol{y}^{(\rho)}(\boldsymbol{x}) = \sum_{\mathfrak{u} \subseteq \mathfrak{g}^{(\rho)} \,:\, \mathfrak{u} \neq \emptyset} \boldsymbol{y}_\mathfrak{u}(\boldsymbol{x})$. That is, $\boldsymbol{y}^{(\rho)}(\boldsymbol{x})$ is the summation of all functions in the DFA decomposition indexed by a subset of $\mathfrak{g}^{(\rho)}$. Notice that $\boldsymbol{y}^{(\rho)}(\boldsymbol{x})$ is a function of only $\boldsymbol{x}^{(\rho)}$, and thus can be rewritten as $\boldsymbol{y}^{(\rho)}(\boldsymbol{x}^{(\rho)})$. Then, based on the DFA decomposition of $\boldsymbol{y}(\boldsymbol{x})$, we can write $\boldsymbol{y}(\boldsymbol{x}) = \beta_0 + \sum_{\rho \in \mathcal{G}} \boldsymbol{y}^{(\rho)}(\boldsymbol{x}^{(\rho)}) + \boldsymbol{y}^{(r)}(\boldsymbol{x})$, where $\beta_0 = \bar{y}$, and $\boldsymbol{y}^{(r)}(\boldsymbol{x})$ represents the remainder. Namely, $\boldsymbol{y}^{(r)}(\boldsymbol{x})$ is the summation of $\boldsymbol{y}_\mathfrak{u}(\boldsymbol{x})$ for $\mathfrak{u} \subseteq \mathfrak{d}$ such that $\mathfrak{u} \neq \emptyset$ and $\mathfrak{u} \not\subseteq \mathfrak{g}^{(\rho)}$ for all $\rho \in \mathcal{G}$. Thus, $\boldsymbol{y}^{(r)}(\boldsymbol{x})$ is a function of *all* component dimensions of $\boldsymbol{x}$ and captures the interactions across the groups.

Let $\boldsymbol{y}^{(\rho)} = \Big[\boldsymbol{y}^{(\rho)}(\boldsymbol{x}_1^{(\rho)}), \boldsymbol{y}^{(\rho)}(\boldsymbol{x}_2^{(\rho)}), \dots, \boldsymbol{y}^{(\rho)}(\boldsymbol{x}_{n^{(\rho)}}^{(\rho)})\Big]^\top$ be the vector of group ρ components of the objective function. Also, let $\boldsymbol{y}^{(r)}$ denote vector of remainder terms $[\boldsymbol{y}^{(r)}(\boldsymbol{x}_1), \boldsymbol{y}^{(r)}(\boldsymbol{x}_2), \dots, \boldsymbol{y}^{(r)}(\boldsymbol{x}_n)]^\top$. Using the $\boldsymbol{y}^{(\rho)}$'s and $\boldsymbol{y}^{(r)}$, the vector of objective function values at all solutions in $\mathcal{X}$ can be expressed as $\boldsymbol{y} = \beta_0 \vec{\mathbf{1}}_n + \sum_{\rho \in \mathcal{G}} \boldsymbol{T} \boldsymbol{y}^{(\rho)} + \boldsymbol{y}^{(r)}$, where $\vec{\mathbf{1}}_n$ is an n-dimensional vector of ones and $\boldsymbol{T}$ is the transformation matrix associated with group ρ. The (k, l)th element of $\boldsymbol{T}$ is 1 if $\boldsymbol{x}_l^{(\rho)}$ is the corresponding lower dimensional component of solution $\boldsymbol{x}_k$, and 0 otherwise. Notice that $\boldsymbol{T}$ is an $n \times n^{(\rho)}$ matrix containing only one nonzero element with value 1 in each row. To assist optimization we impose a prior distribution on $\boldsymbol{y}$.

Starting with the group function $\boldsymbol{y}^{(\rho)}$, we model it as a realization of the GMRF

$$\mathbb{Y}^{(\rho)} = \Big[\mathbb{Y}^{(\rho)}(\boldsymbol{x}_1^{(\rho)}), \mathbb{Y}^{(\rho)}(\boldsymbol{x}_2^{(\rho)}), \dots, \mathbb{Y}^{(\rho)}(\boldsymbol{x}_{n^{(\rho)}}^{(\rho)})\Big]^\top \sim \mathrm{N}\Big(\vec{\mathbf{0}}_{n^{(\rho)}}, [\boldsymbol{Q}^{(\rho)}]^{-1}\Big)$$

with precision matrix $\boldsymbol{Q}^{(\rho)}$ of size $n^{(\rho)} \times n^{(\rho)}$. The prior mean of $\mathbb{Y}^{(\rho)}$ is zero as a consequence of the DFA decomposition; see Property (b) of

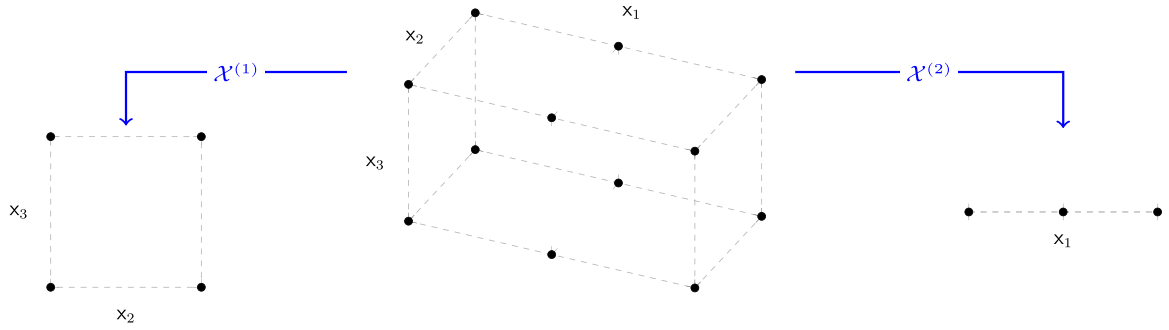


Fig. 1. Illustration of a decomposition into two groups with $g^{(1)} = \{2, 3\}$ and $g^{(2)} = \{1\}$.

Proposition 1. Similarly, the remainder $\boldsymbol{y}^{(r)}$ can be modeled as a realization of the GMRF $\mathbb{Y}^{(r)} \sim \mathrm{N}\left(\vec{\mathbf{0}}_n, [\boldsymbol{Q}^{(r)}]^{-1}\right)$ with precision matrix $\boldsymbol{Q}^{(r)}$ of size $n \times n$.

What sets our model apart from other additive approximations proposed for high-dimensional BO (cf. Kandasamy et al. 2015, Gardner et al. 2017, and Wang et al. 2018) is that we retain the term $\mathbb{Y}^{(r)}$, representing the inadequacy of a fully separable model. Therefore, we do not assume $\boldsymbol{y}(\cdot)$ separates into the sum of functions of groups, unlike other approaches. All of our GMRFs are mutually independent by Property (c) of Proposition 1 because uncorrelated implies independence for Gaussian processes.

In summary, as justified by the DFA decomposition, the vector $\boldsymbol{y}$ of objective function values is modeled as the realization of a constant plus a linear combination of mean-zero, independent GMRFs

$$\mathbb{Y} = \beta_0 \vec{\mathbf{1}}_n + \sum_{\rho \in \mathcal{G}} \boldsymbol{T} \mathbb{Y}^{(\rho)} + \mathbb{Y}^{(r)}. \tag{3}$$

The goal of defining the prior distribution on the DFA decomposition is to achieve significant computational benefits when obtaining the posterior of $\mathbb{Y}$ via the posteriors of the g lower-dimensional GMRFs. However, (3) does not reduce computation *unless* $\boldsymbol{Q}^{(r)}$ *is diagonal*, since it is full-dimensional ($n \times n$); diagonal implies no spatial correlation representing interaction. This is the first issue we address.

A second issue, whether $\boldsymbol{Q}^{(r)}$ is diagonal or not, is that if the values of its diagonal elements are very large, which may occur when the contribution of $\boldsymbol{y}^{(r)}(\boldsymbol{x})$ to the objective function value $\boldsymbol{y}(\boldsymbol{x})$ is negligible, then $\boldsymbol{y}(\boldsymbol{x})$ becomes closer to a purely separable function. Purely separable $\mathbb{Y}$ (without $\mathbb{Y}^{(r)}$) has linearly dependent elements due to the repetitive rows in the $\boldsymbol{T}$'s, leading to a singular covariance matrix for $\mathbb{Y}$. The singular covariance matrix may cause computational issues when updating the posterior distribution of $\mathbb{Y}$. More importantly, the linear dependence of groups causes inconsistencies between $\boldsymbol{y}$ and $\mathbb{Y}$ unless the former has the same additive structure as the latter. See Section EC.4 in the e-companion for a small example illustrating the linear dependence issue.

One of our key contributions is addressing these two issues in a way that preserves computational efficiency without losing a full-dimensional representation. To do this we divide and conquer.

Our first innovation is to obtain the posterior distributions of groups $1, 2, \ldots, g-1$ by modeling group g plus the remainder term as a simple *random effect*, leading to a computationally tractable posterior distribution for $\mathbb{Y}$. Specifically, we replace $\boldsymbol{T}^{(g)}\mathbb{Y}^{(g)} + \mathbb{Y}^{(r)}$ with a GMRF with prior distribution $\mathbb{W} \sim \mathrm{N}\left(\vec{\mathbf{0}}_n, \sigma_g^2 \mathbf{I}_n\right)$ with a positive but finite σ_g^2, where $\mathbf{I}_n$ is the $n \times n$ identity matrix. Therefore, the GMRF prior for $\boldsymbol{y}$ becomes

$$\mathbb{Y} = \beta_0 \vec{\mathbf{1}}_n + \sum_{\rho \in \mathcal{G}^{(-g)}} \boldsymbol{T} \mathbb{Y}^{(\rho)} + \mathbb{W}, \tag{4}$$

where $\mathcal{G}^{(-g)} = \mathcal{G} \setminus \{g\}$ eases notation. The identity of "group g" may change in each iteration of DASSO so that no group is restricted to be only a part of $\mathbb{W}$.

Both (3) and (4) model the same problem but with different prior distributions; (4) provides weaker inference on the component dimensions in group g, but is computationally tractable and free of the inconsistency issue. Moreover, updating the identity of g lets (4) learn all groups' effects equitably.

Under (4), the distribution of $\mathbb{Y}$ is a convolution of low-dimensional distributions and an easy-to-evaluate full-dimensional distribution. Moreover, for fixed $\mathcal{G}$, $\mathbb{Y}$ depends only on lower dimensional components $\boldsymbol{x}^{(-g)} = (x_i)_{i \in \{1,\ldots,d\} \setminus g^{(g)}}$, which we refer to as the "first $g-1$ components." Therefore, it infers the same posterior for any feasible solution with the same first $g-1$ components. Utilizing (4), DASSO performs a "dice" stage to determine the first $g-1$ components of the solution to simulate by maximizing the CEI defined over $\mathcal{X}^{(-g)} = \bigtimes_{i \in \{1,\ldots,d\} \setminus g^{(g)}} \mathcal{X}_i$.

Our second innovation is to further model the component dimensions in group g by a $n^{(g)}$-dimensional GMRF for all solutions with the same first $g-1$ components, as selected in the dice stage. DASSO then performs a "slice" stage to find the components in group g to simulate using the single-group GMRF.

In summary, DASSO alternates between using a simple random-effect GMRF to represent the complex remainder term when choosing values for the $g-1$ groups in the dice stage, and a detailed GMRF of the lower-dimensional slice of the remaining group with those values fixed. As we show in Section 5 this leads to rapid search progress with low computational overhead.

Remark: We have argued that the prior in (4) is ideal for DASSO, and the form of the prior leads to a computationally feasible posterior computation, as shown below. However, to be search-effective the prior must have appropriate parameters. The DASSO prior has $2 + \sum_{\rho=1}^{g}(d^{(\rho)} + 1)$ parameters in total, $d^{(\rho)} + 1$ for each non-remainder group, and two for the overall mean and the random-effect variance. In Section EC.5 in the e-companion, we describe how we estimate these parameters via maximum likelihood so that they are consistent with the decomposition in (4). In brief, we apply maximum likelihood estimation to the *differences* in the simulation outputs at carefully chosen *pairs* of design points so that through cancellation the difference of a pair reflects only the effect of a single group in (4). This accounts for all but two of the parameters, whose estimation is straightforward.

4.3. Bayesian inference for dice stage

In this section we derive the posterior distribution for the dice stage. Recall that $\mathcal{D} \subseteq \mathcal{X}$ is the set of feasible solutions that have been simulated, and $\mathcal{U} = \mathcal{X} \setminus \mathcal{D}$ is the set of feasible solutions that have not. Using these disjoint sets, the random-effect vector $\mathbb{W}$ can be partitioned into $\mathbb{W}_{\mathcal{U}}$ and $\mathbb{W}_{\mathcal{D}}$, which are the subvectors of $\mathbb{W}$ including the solutions in $\mathcal{U}$ and $\mathcal{D}$, respectively.

For each group $\rho \in \mathcal{G}$, let $\mathcal{D}^{(\rho)} \subseteq \mathcal{X}^{(\rho)}$ denote the set of lower-dimensional components of the solutions in $\mathcal{D}$ corresponding to group ρ, and partition $\mathcal{X}^{(\rho)}$ into the two disjoint sets $\mathcal{D}^{(\rho)}$ and $\mathcal{U}^{(\rho)} = \mathcal{X}^{(\rho)} \setminus \mathcal{D}^{(\rho)}$. Assuming that the elements of $\mathbb{Y}^{(\rho)}$ are reordered, we can partition $\mathbb{Y}^{(\rho)}$

and precision matrix $\boldsymbol{Q}^{(\rho)}$ as

$$\mathbb{Y}^{(\rho)} = \begin{pmatrix} \mathbb{Y}^{(\rho)}_{\mathcal{U}} \\ \mathbb{Y}^{(\rho)}_{D} \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \vec{\mathbf{0}}_{|\mathcal{U}^{(\rho)}|} \\ \vec{\mathbf{0}}_{|D^{(\rho)}|} \end{pmatrix}, \begin{pmatrix} \boldsymbol{Q}^{(\rho)}_{\mathcal{U}\mathcal{U}} & \boldsymbol{Q}^{(\rho)}_{\mathcal{U}D} \\ \boldsymbol{Q}^{(\rho)}_{D\mathcal{U}} & \boldsymbol{Q}^{(\rho)}_{DD} \end{pmatrix}^{-1} \right),$$

where $\mathbb{Y}^{(\rho)}_{\mathcal{U}}$ and $\mathbb{Y}^{(\rho)}_{D}$ are the subvectors of $\mathbb{Y}^{(\rho)}$ including the points in $D^{(\rho)}$ and $\mathcal{U}^{(\rho)}$, respectively. The covariance matrix of $\mathbb{Y}^{(\rho)}_{D}$, $\boldsymbol{\Sigma}^{(\rho)}_{DD} = \left[\boldsymbol{Q}^{(\rho)}_{DD} - \boldsymbol{Q}^{(\rho)}_{D\mathcal{U}}[\boldsymbol{Q}^{(\rho)}_{\mathcal{U}\mathcal{U}}]^{-1}\boldsymbol{Q}^{(\rho)}_{\mathcal{U}D}\right]^{-1}$, can be derived from the block matrix inversion formula. Similarly, transformation matrix $\boldsymbol{T}$ can also be partitioned into

$$\begin{pmatrix} \boldsymbol{T}^{(\rho)}_{\mathcal{U}\mathcal{U}} & \boldsymbol{T}^{(\rho)}_{\mathcal{U}D} \\ \mathbf{0}_{|D|\times|\mathcal{U}^{(\rho)}|} & \boldsymbol{T}^{(\rho)}_{DD} \end{pmatrix}.$$

The lower-left block is a zero matrix because $\boldsymbol{x}^{(\rho)} \in D^{(\rho)}$ if $\boldsymbol{x} \in D$ from the definition of $D^{(\rho)}$. Then, $\mathbb{Y}_D = \beta_0\vec{\mathbf{1}}_{|D|} + \sum_{\rho\in\mathcal{G}^{(-g)}} \boldsymbol{T}^{(\rho)}_{DD}\mathbb{Y}^{(\rho)}_{D} + \mathbb{W}_D$. The mean vector and covariance matrix of $\mathbb{Y}_D$ are $\boldsymbol{\mu}_D = \beta_0\vec{\mathbf{1}}_{|D|}$ and $\boldsymbol{\Sigma}_{DD} = \sum_{\rho\in\mathcal{G}^{(-g)}} \boldsymbol{T}^{(\rho)}_{DD}\boldsymbol{\Sigma}^{(\rho)}_{DD}[\boldsymbol{T}^{(\rho)}_{DD}]^{\top} + \sigma_g^2\mathbf{I}_{|D|}$, respectively.

Theorems 1 and 2 state the posterior distributions of $\mathbb{Y}^{(\rho)}$ for $\rho \in \mathcal{G}^{(-g)}$ and $\mathbb{W}$, respectively.

Theorem 1. *For $\rho \in \mathcal{G}^{(-g)}$, the conditional distribution of $\mathbb{Y}^{(\rho)}$ given $\mathbb{Y}^{\epsilon}_{D} = \bar{\boldsymbol{Y}}_D$ is normal with mean vector $\boldsymbol{m}^{(\rho)}$ and covariance matrix $\bar{\boldsymbol{\Sigma}}^{(\rho)} = [\bar{\boldsymbol{Q}}]^{-1}$, where*

$$\boldsymbol{m}^{(\rho)} = [\bar{\boldsymbol{Q}}]^{-1}\begin{pmatrix} \vec{\mathbf{0}}_{|\mathcal{U}^{(\rho)}|} \\ [\boldsymbol{T}^{(\rho)}_{DD}]^{\top}\boldsymbol{E}\left(\bar{\boldsymbol{Y}}_D - \boldsymbol{\mu}_D\right) \end{pmatrix},$$

$$\bar{\boldsymbol{Q}} = \boldsymbol{Q}^{(\rho)} + \begin{pmatrix} \mathbf{0}_{|\mathcal{U}^{(\rho)}|\times|\mathcal{U}^{(\rho)}|} & \mathbf{0}_{|\mathcal{U}^{(\rho)}|\times|D^{(\rho)}|} \\ \mathbf{0}_{|D^{(\rho)}|\times|\mathcal{U}^{(\rho)}|} & [\boldsymbol{T}^{(\rho)}_{DD}]^{\top}\boldsymbol{E}\boldsymbol{T}^{(\rho)}_{DD} \end{pmatrix},$$

and $\boldsymbol{E} = \left[\boldsymbol{\Sigma}_{DD} - \boldsymbol{T}^{(\rho)}_{DD}\boldsymbol{\Sigma}^{(\rho)}_{DD}[\boldsymbol{T}^{(\rho)}_{DD}]^{\top} + \boldsymbol{\Sigma}^{\epsilon}\right]^{-1}$.

Theorem 2. *The conditional distribution of $\mathbb{W}$ given $\mathbb{Y}^{\epsilon}_{D} = \bar{\boldsymbol{Y}}_D$ is normal with mean vector $\boldsymbol{m}^{(g)}$ and covariance matrix $\bar{\boldsymbol{\Sigma}}^{(g)} = [\bar{\boldsymbol{Q}}^{(g)}]^{-1}$, where*

$$\boldsymbol{m}^{(g)} = [\bar{\boldsymbol{Q}}^{(g)}]^{-1}\begin{pmatrix} \vec{\mathbf{0}}_{|\mathcal{U}|} \\ \boldsymbol{E}^{(g)}\left(\bar{\boldsymbol{Y}}_D - \boldsymbol{\mu}_D\right) \end{pmatrix}, \ \bar{\boldsymbol{Q}}^{(g)} = \frac{1}{\sigma_g^2}\mathbf{I}_n + \begin{pmatrix} \mathbf{0}_{|\mathcal{U}|\times|\mathcal{U}|} & \mathbf{0}_{|\mathcal{U}|\times|D|} \\ \mathbf{0}_{|D|\times|\mathcal{U}|} & \boldsymbol{E}^{(g)} \end{pmatrix},$$

and $\boldsymbol{E}^{(g)} = \left[\boldsymbol{\Sigma}_{DD} - \sigma_g^2\mathbf{I}_{|D|} + \boldsymbol{\Sigma}^{\epsilon}\right]^{-1}$.

Since $\bar{\boldsymbol{\Sigma}}^{(g)}$ is the inverse of a block-diagonal matrix, it has a block-diagonal structure. Although $\bar{\boldsymbol{\Sigma}}^{(g)}$ is an $n\times n$ matrix, its block-diagonal structure enables us to compute it by inverting only a $|D|\times|D|$ matrix. Corollary 1 below follows immediately from the fact that $\mathbb{Y} = \beta_0\vec{\mathbf{1}}_n + \sum_{\rho\in\mathcal{G}^{(-g)}} \boldsymbol{T}\mathbb{Y}^{(\rho)} + \mathbb{W}$.

Corollary 1. *The conditional distribution of $\mathbb{Y}$ given $\mathbb{Y}^{\epsilon}_{D} = \bar{\boldsymbol{Y}}_D$ is normal with the conditional mean vector and conditional covariance matrix*

$$\boldsymbol{m} = \beta_0\vec{\mathbf{1}}_n + \sum_{\rho\in\mathcal{G}^{(-g)}} \boldsymbol{T}\boldsymbol{m}^{(\rho)} + \boldsymbol{m}^{(g)} \text{ and } \bar{\boldsymbol{\Sigma}} = \sum_{\rho\in\mathcal{G}^{(-g)}} \boldsymbol{T}\bar{\boldsymbol{\Sigma}}^{(\rho)}[\boldsymbol{T}]^{\top} + \bar{\boldsymbol{\Sigma}}^{(g)}.$$

Let $\boldsymbol{v} = \mathrm{diag}\left(\bar{\boldsymbol{\Sigma}}\right)$ and $\boldsymbol{c} = \bar{\boldsymbol{\Sigma}}\mathbf{e}_{\tilde{\boldsymbol{x}}}$, where $\mathrm{diag}(\cdot)$ represents the diagonal of the corresponding matrix as a vector, and $\mathbf{e}_{\tilde{\boldsymbol{x}}}$ is the n-dimensional basis vector consisting of 0s and a single 1 in the position corresponding to $\tilde{\boldsymbol{x}}$. Then $\boldsymbol{v}$ is the vector of conditional variances of $\mathbb{Y}$, and $\boldsymbol{c}$ is the vector of conditional covariances between $\mathbb{Y}(\tilde{\boldsymbol{x}})$ and all components of $\mathbb{Y}$. For any $\boldsymbol{x}$, conditional mean $\boldsymbol{m}(\boldsymbol{x})$, conditional variance $\boldsymbol{v}(\boldsymbol{x})$ and conditional covariance $\boldsymbol{c}(\tilde{\boldsymbol{x}}, \boldsymbol{x})$ are the respective components of $\boldsymbol{m}$, $\boldsymbol{v}$ and $\boldsymbol{c}$ associated with $\boldsymbol{x}$.

For group $\rho \in \mathcal{G}^{(-g)}$, let $\boldsymbol{m}^{(\rho)}(\boldsymbol{x}^{(\rho)})$, $\boldsymbol{v}^{(\rho)}(\boldsymbol{x}^{(\rho)})$ and $\boldsymbol{c}^{(\rho)}(\tilde{\boldsymbol{x}}^{(\rho)}, \boldsymbol{x}^{(\rho)})$ be, respectively, the components of $\boldsymbol{m}^{(\rho)}$, $\boldsymbol{v}^{(\rho)}$ and $\boldsymbol{c}^{(\rho)}$, associated with lower dimensional component $\boldsymbol{x}^{(\rho)}$, where $\boldsymbol{v}^{(\rho)} = \mathrm{diag}\left(\bar{\boldsymbol{\Sigma}}^{(\rho)}\right)$ and $\boldsymbol{c}^{(\rho)} = \bar{\boldsymbol{\Sigma}}^{(\rho)}\mathbf{e}^{(\rho)}_{\tilde{\boldsymbol{x}}}$. Also, let $\boldsymbol{m}^{(g)}(\boldsymbol{x})$, $\boldsymbol{v}^{(g)}(\boldsymbol{x})$ and $\boldsymbol{c}^{(g)}(\tilde{\boldsymbol{x}}, \boldsymbol{x})$ be, respectively, the components of $\boldsymbol{m}^{(g)}$, $\boldsymbol{v}^{(g)}$ and $\boldsymbol{c}^{(g)}$, associated with solution $\boldsymbol{x}$, where $\boldsymbol{v}^{(g)} = \mathrm{diag}\left(\bar{\boldsymbol{\Sigma}}^{(g)}\right)$ and $\boldsymbol{c}^{(g)} = \bar{\boldsymbol{\Sigma}}^{(g)}\mathbf{e}_{\tilde{\boldsymbol{x}}}$. Due to the special structure of $\boldsymbol{T}$, Corollary 1 implies that

$$\boldsymbol{m}(\boldsymbol{x}) = \sum_{\rho\in\mathcal{G}^{(-g)}} \boldsymbol{m}^{(\rho)}(\boldsymbol{x}^{(\rho)}) + \boldsymbol{m}^{(g)}(\boldsymbol{x}), \ \boldsymbol{v}(\boldsymbol{x}) = \sum_{\rho\in\mathcal{G}^{(-g)}} \boldsymbol{v}^{(\rho)}(\boldsymbol{x}^{(\rho)}) + \boldsymbol{v}^{(g)}(\boldsymbol{x})$$

and

$$\boldsymbol{c}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) = \sum_{\rho\in\mathcal{G}^{(-g)}} \boldsymbol{c}^{(\rho)}(\tilde{\boldsymbol{x}}^{(\rho)}, \boldsymbol{x}^{(\rho)}) + \boldsymbol{c}^{(g)}(\tilde{\boldsymbol{x}}, \boldsymbol{x}).$$

Observe that $\boldsymbol{m}(\boldsymbol{x})$, $\boldsymbol{v}(\boldsymbol{x})$, and $\boldsymbol{c}(\tilde{\boldsymbol{x}}, \boldsymbol{x})$ are obtained from terms extracted from low-dimensional posterior distributions and an easy-to-evaluate full-dimensional posterior distribution. This is as opposed to a *computationally impossible-to-evaluate* full-dimensional distribution. And because we can choose any way to group the component decision variables we can effectively manage the burden of the posterior calculations. These computational savings enable DASSO to deal with high-dimensional DSO problems.

From the terms $\boldsymbol{m}(\boldsymbol{x})$, $\boldsymbol{v}(\boldsymbol{x})$, and $\boldsymbol{c}(\tilde{\boldsymbol{x}}, \boldsymbol{x})$, we can compute the CEI of $\boldsymbol{x}$. Recall that BO algorithms find good solutions rapidly because they are guided by evaluating—in our case simulating—the solution offering the most potential improvement based on inference from the posterior distribution, but this is a bottleneck if there are (say) trillions of solutions whose improvement must be assessed on each iteration. In Section 4.6, we make a significant computational improvement for the CEI calculation by exploiting the fact that $\boldsymbol{m}(\boldsymbol{x})$, $\boldsymbol{v}(\boldsymbol{x})$, and $\boldsymbol{c}(\tilde{\boldsymbol{x}}, \boldsymbol{x})$ are written in additive forms, and thus we can easily determine if there is another solution with a larger CEI than a group of solutions without fully computing their CEIs. This provides a substantial computational saving because it suffices to calculate the CEIs for only a relatively small number of feasible solutions to find a CEI-maximizing solution; see Section 5.3.

Expressing $\boldsymbol{m}(\boldsymbol{x})$, $\boldsymbol{v}(\boldsymbol{x})$, and $\boldsymbol{c}(\tilde{\boldsymbol{x}}, \boldsymbol{x})$ in additive forms also enables us to show that the CEIs of the feasible solutions in $\mathcal{U}$ with the same first $g-1$ components are identical; see Proposition 2 in Section 4.6. Therefore, there might be multiple solutions with the largest CEI. This is mainly a consequence of the random-effect group having a diagonal covariance matrix.

We think of the posteriors of the first $g-1$ groups as individual *dice* because the overall posterior includes their summation. Once we "roll the dice" (select their CEI maximizing components), we can evaluate a *slice* of the full-dimensional posterior that we address in the next section.

4.4. Bayesian inference for slice stage

Suppose that values of $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(g-1)}$, the first $g-1$ components, are chosen in the dice stage. For simplicity, let $\boldsymbol{z}$ denote these chosen components; notice that $\boldsymbol{z}$ is not full-dimensional, i.e., its dimension is $d - d^{(g)}$. Then, the remaining *slice* of $\mathbb{Y}$, denoted by $\mathbb{Y}(\cdot|\boldsymbol{z})$, is only a function of the last component, $\boldsymbol{x}^{(g)}$, and thus lower dimensional. In the inventory problem this corresponds to fixing the reorder and order-up-to values of some products with the remaining yet to be chosen. Thus, $\mathbb{Y}(\cdot|\boldsymbol{z}) = \boldsymbol{S}_{\boldsymbol{z}}\mathbb{Y}$ is a random vector of size $n^{(g)}\times 1$, where $\boldsymbol{S}_{\boldsymbol{z}}$ is the transformation matrix to fix $\boldsymbol{x}^{(-g)}$ to $\boldsymbol{z}$. We refer to the distribution of $\mathbb{Y}(\cdot|\boldsymbol{z})$ as a "slice" rather than a "conditional" because it is a subvector of $\mathbb{Y}$ and not conditioned on $\boldsymbol{z}$. The (k,l)th element of $\boldsymbol{S}_{\boldsymbol{z}}$ is 1 if solution $\boldsymbol{x}_l$ is composed of $\boldsymbol{x}^{(-g)} = \boldsymbol{z}$ and $\boldsymbol{x}^{(g)}_k$, and 0 otherwise. Notice that $\boldsymbol{S}_{\boldsymbol{z}}$ is a $n^{(g)}\times n$ matrix containing one nonzero element with value 1 in each row. Under (3), $\mathbb{Y}(\cdot|\boldsymbol{z}) = \beta_0\boldsymbol{S}_{\boldsymbol{z}}\vec{\mathbf{1}}_n + \sum_{\rho\in\mathcal{G}^{(-g)}} \boldsymbol{S}_{\boldsymbol{z}}\boldsymbol{T}\mathbb{Y}^{(\rho)} + \boldsymbol{S}_{\boldsymbol{z}}\boldsymbol{T}^{(g)}\mathbb{Y}^{(g)} + \boldsymbol{S}_{\boldsymbol{z}}\mathbb{Y}^{(r)}$. The special structures of the transformation matrices $\boldsymbol{S}_{\boldsymbol{z}}$ and $\boldsymbol{T}$ imply that $\boldsymbol{S}_{\boldsymbol{z}}\vec{\mathbf{1}}_n = \vec{\mathbf{1}}_{n^{(g)}}$, $\boldsymbol{S}_{\boldsymbol{z}}\boldsymbol{T}\mathbb{Y}^{(\rho)} = \mathbb{Y}^{(\rho)}(\boldsymbol{z}^{(\rho)})\vec{\mathbf{1}}_{n^{(g)}}$ for $\rho \in \mathcal{G}^{(-g)}$, and $\boldsymbol{S}_{\boldsymbol{z}}\boldsymbol{T}^{(g)} = \mathbf{I}_{n^{(g)}}$, where $\boldsymbol{z}^{(\rho)}$ represents the group ρ component of $\boldsymbol{z}$. Therefore,

$$\mathbb{Y}(\cdot|\boldsymbol{z}) = \left(\beta_0 + \sum_{\rho\in\mathcal{G}^{(-g)}} \mathbb{Y}^{(\rho)}(\boldsymbol{z}^{(\rho)})\right)\vec{\mathbf{1}}_{n^{(g)}} + \mathbb{Y}^{(g)} + \boldsymbol{S}_{\boldsymbol{z}}\mathbb{Y}^{(r)}.$$

Ignoring the remainder part, $\boldsymbol{S}_{\boldsymbol{z}}\mathbb{Y}^{(r)}$, for simplicity, we impose a GMRF prior with mean vector $\beta_{\boldsymbol{z}}\vec{\mathbf{1}}_{n^{(g)}}$ and precision matrix $\boldsymbol{Q}^{(g)}$ on $\mathbb{Y}(\cdot|\boldsymbol{z})$, where $\beta_{\boldsymbol{z}} = \beta_0 + \sum_{\rho\in\mathcal{G}^{(-g)}} \boldsymbol{y}^{(\rho)}(\boldsymbol{z}^{(\rho)})$. While $\mathbb{Y}(\cdot|\boldsymbol{z})$ could be modeled with any appropriate precision matrix, we use $\boldsymbol{Q}^{(g)}$ because of its convenience and simplicity; recall that the identity of g is updated over time and the parameters of $\boldsymbol{Q}^{(\rho)}$ are initially estimated for each group $\rho \in \mathcal{G}$.

Let $\mathcal{X}_{\boldsymbol{z}} = \{\boldsymbol{x} \in \mathcal{X} : \boldsymbol{x}^{(-g)} = \boldsymbol{z}\}$ denote the set of feasible solutions whose first $g-1$ components are $\boldsymbol{z}$; notice that $|\mathcal{X}_{\boldsymbol{z}}| = n^{(g)}$. Also, let

Algorithm 1 DASSO.

1: Initialize D by choosing a subset of feasible solutions. Simulate the solutions in D to estimate the parameters (i.e., $\boldsymbol{Q}^{(\rho)}$ and σ_ρ^2 for $\rho \in \mathcal{G}$, and β_0).
2: **while** stopping condition of the *dice stage* not satisfied **do**
3: Choose a group to be the last group g.
4: Re-estimate β_0 by using the simulation outputs of the solutions in D.
5: Using the posterior distribution of $\mathbb{Y}$, calculate the CEIs of the solutions to find $\acute{\boldsymbol{x}}$ with the largest CEI, i.e., $\acute{\boldsymbol{x}} \in \arg\max_{\boldsymbol{x}\in\mathcal{X}\setminus\{\tilde{\boldsymbol{x}}\}} \mathrm{CEI}(\tilde{\boldsymbol{x}}, \boldsymbol{x})$.
6: Restrict the set of candidate solutions to $\mathcal{X}_{\boldsymbol{z}} = \{\boldsymbol{x} \in \mathcal{X} : \boldsymbol{x}^{(-g)} = \boldsymbol{z}\}$, where $\boldsymbol{z} = \acute{\boldsymbol{x}}^{(-g)}$.
7: Simulate the current sample-best solution $\tilde{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\in D} \bar{\boldsymbol{Y}}_D(\boldsymbol{x})$.
8: Simulate some solutions in $\mathcal{X}_{\boldsymbol{z}}$ if $D_{\boldsymbol{z}} = \mathcal{X}_{\boldsymbol{z}} \cap D = \emptyset$ so that $\beta_{\boldsymbol{z}}$ can be estimated.
9: **while** stopping condition of the *slice stage* not satisfied **do**
10: For the solutions in $\mathcal{X}_{\boldsymbol{z}}$, let $\mathbb{Y}(\cdot|\boldsymbol{z}) \sim \mathrm{N}\left(\beta_{\boldsymbol{z}} \vec{\mathbf{1}}_{n^{(g)}}, [\boldsymbol{Q}^{(g)}]^{-1}\right)$ be a GMRF.
11: Estimate $\beta_{\boldsymbol{z}}$ by using the simulation outputs of the solutions in $D_{\boldsymbol{z}}$.
12: Let $\tilde{\boldsymbol{x}}|\boldsymbol{z} = \arg\min_{\boldsymbol{x}\in D_{\boldsymbol{z}}} \bar{\boldsymbol{Y}}_D(\boldsymbol{x})$ be the sample-best solution in $D_{\boldsymbol{z}}$.
13: Using the posterior distribution of $\mathbb{Y}(\cdot|\boldsymbol{z})$, calculate the CEIs of the solutions in $\mathcal{X}_{\boldsymbol{z}}$ to find the solution with the largest CEI, i.e., $\arg\max_{\boldsymbol{x}\in\mathcal{X}_{\boldsymbol{z}}} \mathrm{CEI}(\tilde{\boldsymbol{x}}|\boldsymbol{z}, \boldsymbol{x})$.
14: Simulate $\arg\max_{\boldsymbol{x}\in\mathcal{X}_{\boldsymbol{z}}} \mathrm{CEI}(\tilde{\boldsymbol{x}}|\boldsymbol{z}, \boldsymbol{x})$ and $\tilde{\boldsymbol{x}}|\boldsymbol{z}$.
15: **end while**
16: **end while**

$D_{\boldsymbol{z}} = \mathcal{X}_{\boldsymbol{z}} \cap D$ denote the set of design points (i.e., feasible solutions that have been simulated) whose first $g-1$ components are $\boldsymbol{z}$. Partitioning $\mathbb{Y}(\cdot|\boldsymbol{z})$ into two subvectors accordingly, the conditional distribution of $\mathbb{Y}(\cdot|\boldsymbol{z})$ given $\mathbb{Y}_D^c = \bar{\boldsymbol{Y}}_D$ can be obtained from (1).

4.5. The DASSO algorithm

DASSO is presented in Algorithm 1. At a high level, DASSO employs CEI to choose the values of the first $g-1$ components of $\boldsymbol{x}$ in the dice stage, selects the remaining components of group g in the slice stage, simulates the selected solutions and then repeats. A more detailed description follows.

Algorithm 1 first initializes the design set D by choosing a subset of feasible solutions. After simulating the solutions in D, the GMRF parameters of the prior are estimated. DASSO then starts the dice stage: the identity of the last group is chosen and the parameter β_0 is re-estimated accordingly by using the simulation outputs from the solutions in D. Re-estimation of β_0 is not necessary, but doing so helps the inference since the estimated value depends on the identity of the last group. Unlike the other GMRF parameters, estimation of the constant term is straightforward; see Section EC.5 in the e-companion. DASSO next finds a solution with the largest CEI, $\acute{\boldsymbol{x}} \in \arg\max_{\boldsymbol{x}\in\mathcal{X}\setminus\{\tilde{\boldsymbol{x}}\}} \mathrm{CEI}(\tilde{\boldsymbol{x}}, \boldsymbol{x})$, and fixes the first $g-1$ components to $\acute{\boldsymbol{x}}^{(-g)}$, i.e., $\boldsymbol{z} = \acute{\boldsymbol{x}}^{(-g)}$. The set of candidate solutions is restricted to $\mathcal{X}_{\boldsymbol{z}} = \{\boldsymbol{x} \in \mathcal{X} : \boldsymbol{x}^{(-g)} = \boldsymbol{z}\}$ to form a slice. Finally, the current sample-best solution, $\tilde{\boldsymbol{x}}$, is simulated. Fig. 2 illustrates the dice stage for a simple 3-dimensional example with $n = 64$ solutions.

In the slice stage, the parameter $\beta_{\boldsymbol{z}}$ is estimated by using the simulation outputs of the solutions in $D_{\boldsymbol{z}}$. If $D_{\boldsymbol{z}} = \mathcal{X}_{\boldsymbol{z}} \cap D = \emptyset$, i.e., if none of the solutions in $D_{\boldsymbol{z}}$ has been simulated before, then a set of solutions in $\mathcal{X}_{\boldsymbol{z}}$ is chosen to simulate before the estimation. Until the stopping criterion of the slice stage is satisfied, the algorithm simulates $\arg\max_{\boldsymbol{x}\in\mathcal{X}_{\boldsymbol{z}}} \mathrm{CEI}(\tilde{\boldsymbol{x}}|\boldsymbol{z}, \boldsymbol{x})$ and $\tilde{\boldsymbol{x}}|\boldsymbol{z} = \arg\min_{\boldsymbol{x}\in D_{\boldsymbol{z}}} \bar{\boldsymbol{Y}}_D(\boldsymbol{x})$, and updates the slice posterior distribution. Notice that the slice stage computes the CEI of each solution in $D_{\boldsymbol{z}}$ with respect to the sample best within $D_{\boldsymbol{z}}$, but this is a small set. Once the stopping criterion of the slice stage is satisfied, the algorithm returns to the dice stage unless its stopping criterion is also satisfied.

DASSO requires user inputs for the decomposition and some criteria in Steps 1, 2, 3, 8, and 9 as well as an acquisition function such as CEI to guide the search. Below, we discuss the choices adopted in our numerical experiments in Section 5. First, notice that the user-defined decomposition (i.e., the groups) remains the same throughout the algorithm. Although DASSO could allow the decomposition to be updated, the GMRF parameters would need to be estimated for each new decomposition. To estimate the parameters for the user-defined decomposition in Step 1, we employ a parameter estimation method described in Section EC.5 of the e-companion. This method simulates specially constructed pairs of design points. Thus, to estimate the parameters for a new decomposition, additional design points that do not directly guide the search would need to be simulated to complete the pairs, wasting simulation effort. Therefore, we do not update the decomposition in our numerical experiments. Nevertheless, we empirically evaluate the performance of DASSO with different fixed decompositions.

Algorithm 1 contains two stopping conditions, one for the dice stage in Step 2 and another for the slice stage in Step 9. For the former, we adopt a fixed computation budget in terms of total number of replications, i.e., the algorithm terminates after a certain number of replications. For the latter, we similarly adopt a fixed budget in a sense that the number of slice-stage iterations per dice stage is one; in other words, Steps 9 and 15 are removed from the algorithm. We use fixed-budget stopping conditions for the sake of fair comparison in our numerical experiments. An alternative is using CEI for stopping, as proposed in Salemi et al. (2019), where the algorithm terminates once the largest CEI value drops below a specified threshold.

To update the identity of the group g in Step 3, we can impose a discrete distribution with support $\mathcal{G}$ to randomly choose it. Alternatively this discrete distribution could adapt to some metrics measuring the impact of groups on the objective function values and changed from time to time during the search. For the sake of simplicity in our numerical experiments, we impose a discrete uniform distribution; that is, we choose group g randomly at each change. While intuitively it seems clear that each group should take turns as group g, the best way to assign it is an open question. Similarly, to choose some solutions from $\mathcal{X}_{\boldsymbol{z}}$ to simulate when $D_{\boldsymbol{z}} = \emptyset$ in Step 8, a discrete distribution with support $\mathcal{X}^{(g)}$ can be applied; we again adopt a discrete uniform distribution in our numerical experiments.

DASSO employs CEI as the acquisition function in both the dice and slice stages to choose the feasible solution to simulate next. Alternatively, acquisition functions such as the knowledge gradient (Frazier et al. 2009) and information-theoretic approaches (see, e.g., Hernández-Lobato et al. 2014, Hvarfner et al. 2022, Wang & Jegelka 2017) could be applied in the slice stage because the candidate solution set is relatively small. However, implementing them in the dice stage requires more careful consideration in large-scale problems. Specifically, identifying the solution that maximizes the acquisition function among a very large number of feasible solutions on each iteration by enumeration imposes too much computational overhead to be practical. An advantage of CEI, as described in Section 4.6, is that the structural properties of CEI support a strategy that identifies the best-CEI solution while only evaluating a small fraction of them.

Before providing the empirical evaluation to demonstrate the excellent finite-sample performance of DASSO in Section 5, we note that we establish the global convergence of DASSO under the very mild conditions and a small tweak to the algorithm; see Section EC.3 in the e-companion. We confess that this convergence result is strictly of academic interest, since we do not expect to approach anything like convergence in the class of problems we consider. The real test for DASSO and any competitor is whether it makes rapid improvement by exploiting limited simulation, unencumbered by algorithm overhead.

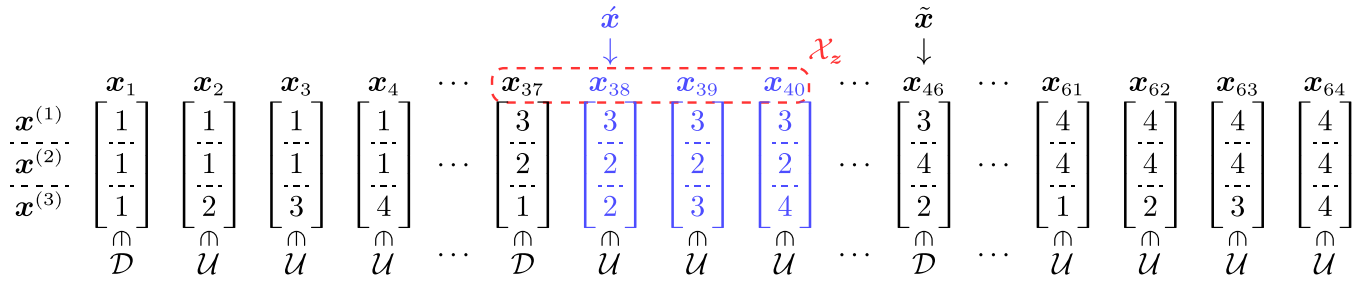


Fig. 2. Illustration of the dice stage on a solution space $\mathcal{X} = \{1,2,3,4\}^3$ with three 1-dimensional groups. Letting $g = 3$, suppose that solutions $\boldsymbol{x}_{38}$, $\boldsymbol{x}_{39}$ and $\boldsymbol{x}_{40}$ (in blue) have the largest CEI in the dice stage in Step 5. Then, the restricted set of candidate solutions is $\mathcal{X}_z = \{\boldsymbol{x}_{37}, \boldsymbol{x}_{38}, \boldsymbol{x}_{39}, \boldsymbol{x}_{40}\}$ (indicated with red dashed line). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4.6. Computational efficiency of dice stage

Although we have focused on the computational expense of the posterior distribution update, all expected-improvement type algorithms suffer the additional computational burden of needing to evaluate the potential of *every* feasible solution on *every* iteration. Choosing a solution heuristically or focusing on only a small subset of feasible solutions risks making a poor decision, which is fatal when simulations are so expensive. Fortunately, DASSO facilitates an elegant solution to this problem.

To make a decision in the dice stage, the CEI of each solution needs to be computed; see Step 5 of Algorithm 1. Aiming to reduce the bottleneck, we derive the following two relationships between the CEIs of solutions that have not been simulated:

Proposition 2. *For two solutions $\boldsymbol{x}$ and $\check{\boldsymbol{x}}$ in $\mathcal{U}$: (R1)* $\mathrm{CEI}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) = \mathrm{CEI}(\tilde{\boldsymbol{x}}, \check{\boldsymbol{x}})$ *if* $\boldsymbol{x}^{(\rho)} = \check{\boldsymbol{x}}^{(\rho)}$ *for all* $\rho \in \mathcal{G}^{(-g)}$. *(R2)* $\mathrm{CEI}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) \le \mathrm{CEI}(\tilde{\boldsymbol{x}}, \check{\boldsymbol{x}})$ *if for some group* $\rho \in \mathcal{G}^{(-g)}$, *all of the following conditions hold:*

(C2.1) $\boldsymbol{x}^{(\varrho)} = \check{\boldsymbol{x}}^{(\varrho)}$ *for all* $\varrho \in \mathcal{G} \setminus \{\rho, g\}$,
(C2.2) $\boldsymbol{m}^{(\rho)}(\boldsymbol{x}^{(\rho)}) \ge \boldsymbol{m}^{(\rho)}(\check{\boldsymbol{x}}^{(\rho)})$, *and*
(C2.3) $\boldsymbol{v}^{(\rho)}(\tilde{\boldsymbol{x}}^{(\rho)}, \boldsymbol{x}^{(\rho)}) \le \boldsymbol{v}^{(\rho)}(\tilde{\boldsymbol{x}}^{(\rho)}, \check{\boldsymbol{x}}^{(\rho)})$, *where* $\boldsymbol{v}^{(\rho)}(\tilde{\boldsymbol{x}}^{(\rho)}, \cdot) = \boldsymbol{v}^{(\rho)}(\tilde{\boldsymbol{x}}^{(\rho)}) + \boldsymbol{v}^{(\rho)}(\cdot) - 2c^{(\rho)}(\tilde{\boldsymbol{x}}^{(\rho)}, \cdot)$.

Furthermore, for two solutions $\boldsymbol{x}$ and $\check{\boldsymbol{x}}$ in $\mathcal{D}$: (R3) $\mathrm{CEI}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) \le \mathrm{CEI}(\tilde{\boldsymbol{x}}, \check{\boldsymbol{x}})$ *if* $\boldsymbol{x}^{(\rho)} = \check{\boldsymbol{x}}^{(\rho)}$ *for all* $\rho \in \mathcal{G}^{(-g)}$, $\boldsymbol{m}^{(g)}(\boldsymbol{x}) \ge \boldsymbol{m}^{(g)}(\check{\boldsymbol{x}})$, *and* $\boldsymbol{v}^{(g)}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) \le \boldsymbol{v}^{(g)}(\tilde{\boldsymbol{x}}, \check{\boldsymbol{x}})$, *where* $\boldsymbol{v}^{(g)}(\tilde{\boldsymbol{x}}, \cdot) = \boldsymbol{v}^{(g)}(\tilde{\boldsymbol{x}}) + \boldsymbol{v}^{(g)}(\cdot) - 2c^{(g)}(\tilde{\boldsymbol{x}}, \cdot)$.

Relation (R1) states that the CEIs of the feasible solutions in $\mathcal{U}$ (unsimulated) with the same first $g-1$ components are identical. If a solution $\boldsymbol{x}$ is dominated by some other solution $\check{\boldsymbol{x}}$ in terms of CEI, then $\boldsymbol{x}$ can be excluded from the candidate solutions to simulate in the next iteration. Recall that $\mathrm{CEI}(\tilde{\boldsymbol{x}}, \cdot)$ decreases in $\boldsymbol{m}(\cdot)$ and increases in $\boldsymbol{v}(\tilde{\boldsymbol{x}}, \cdot)$. From this property, Relation (R2) stipulates that the dominance relationship between CEIs at solutions $\boldsymbol{x}$ and $\check{\boldsymbol{x}}$ can be established without actually computing them by identifying the Pareto-efficient points for each group based on conditional means and variances. Therefore, we can identify solutions in $\mathcal{U}$ whose CEIs cannot be the largest if one of their lower dimensional components is dominated in mean and variance; in the inventory problem this corresponds to reorder and order-up-to points that cannot have the largest expected improvement on the next iteration. Since these relations enable us to compare the CEIs of two solutions in $\mathcal{U}$ without actually calculating them, the computational burden can be *drastically* reduced, especially when the number of feasible solutions is large. Relation (R3) similarly stipulates the dominance relationship between CEIs at design points (i.e., simulated solutions in $\mathcal{D}$) with the same first $g-1$ components. Specifically, we can identify solutions in $\mathcal{D}$ that cannot be the largest if their random-effect component is dominated in mean and variance. Relation (R3) does not reduce the computational burden as much as Relation (R2) does since the cardinality of $\mathcal{D}$ is much smaller than $\mathcal{U}$, but it could help in a DSO problem for which a larger number of distinct solutions can be simulated.

To further elaborate on (R2), let

$$\mathcal{F}^{(\rho)}(\boldsymbol{x}^{(\rho)}) = \big\{\check{\boldsymbol{x}}^{(\rho)} \in \mathcal{X}^{(\rho)} \setminus \{\boldsymbol{x}^{(\rho)}\} : \boldsymbol{m}^{(\rho)}(\boldsymbol{x}^{(\rho)}) \ge \boldsymbol{m}^{(\rho)}(\check{\boldsymbol{x}}^{(\rho)}) \text{ and } \boldsymbol{v}^{(\rho)}(\tilde{\boldsymbol{x}}^{(\rho)}, \boldsymbol{x}^{(\rho)}) \le \boldsymbol{v}^{(\rho)}(\tilde{\boldsymbol{x}}^{(\rho)}, \check{\boldsymbol{x}}^{(\rho)})\big\}$$

denote the set of points that Pareto-dominate $\boldsymbol{x}^{(\rho)} \in \mathcal{X}^{(\rho)}$ for $\rho \in \mathcal{G}^{(-g)}$. Moreover, $\mathrm{CEI}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) \le \mathrm{CEI}(\tilde{\boldsymbol{x}}, \check{\boldsymbol{x}})$ for solutions $\boldsymbol{x}$ and $\check{\boldsymbol{x}}$ in $\mathcal{U}$ with $\boldsymbol{x}^{(\varrho)} = \check{\boldsymbol{x}}^{(\varrho)}$ for all $\varrho \in \mathcal{G} \setminus \{\rho, g\}$ and $\check{\boldsymbol{x}}^{(\rho)} \in \mathcal{F}^{(\rho)}(\boldsymbol{x}^{(\rho)})$. We define a point $\boldsymbol{x}^{(\rho)} \in \mathcal{X}^{(\rho)}$ as Pareto-efficient in group $\rho \in \mathcal{G}^{(-g)}$ if $\mathcal{F}^{(\rho)}(\boldsymbol{x}^{(\rho)}) = \emptyset$, i.e., if it is not Pareto-dominated by any other point. Then, the Pareto frontier, which consists of the Pareto-efficient points, of group $\rho \in \mathcal{G}^{(-g)}$ is $\mathcal{F}^{(\rho)} = \big\{\boldsymbol{x}^{(\rho)} \in \mathcal{X}^{(\rho)} : \mathcal{F}^{(\rho)}(\boldsymbol{x}^{(\rho)}) = \emptyset\big\}$. We use these Pareto frontiers to construct $\mathcal{F} = \big\{\boldsymbol{x} \in \mathcal{U} : \boldsymbol{x}^{(\rho)} \in \mathcal{F}^{(\rho)},\ \forall \rho \in \mathcal{G}^{(-g)}\big\}$, which is the set of solutions in $\mathcal{U}$ with Pareto-efficient points for all lower dimensional components except that of the last group. As a result of Proposition 2, the following corollary presents the conditions under which it suffices to calculate only the CEIs of solutions in $\mathcal{F}$ to identify the solution with the largest CEI, in $\mathcal{U}$.

Corollary 2. *Let* $\acute{\boldsymbol{x}} \in \arg\max_{\boldsymbol{x} \in \mathcal{U} \setminus \mathcal{F}} \mathrm{CEI}(\tilde{\boldsymbol{x}}, \boldsymbol{x})$. *If there exist some solution* $\check{\boldsymbol{x}} \in \mathcal{U}$ *such that* $\acute{\boldsymbol{x}}^{(\varrho)} = \check{\boldsymbol{x}}^{(\varrho)}$ *for all* $\varrho \in \mathcal{G} \setminus \{\rho, g\}$ *and* $\check{\boldsymbol{x}}^{(\rho)} \in \mathcal{F}^{(\rho)}(\acute{\boldsymbol{x}}^{(\rho)})$ *for some group* $\rho \in \mathcal{G}^{(-g)}$, *then*

$$\max_{\boldsymbol{x} \in \mathcal{U} \setminus \mathcal{F}} \mathrm{CEI}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) = \mathrm{CEI}(\tilde{\boldsymbol{x}}, \acute{\boldsymbol{x}}) \le \mathrm{CEI}(\tilde{\boldsymbol{x}}, \check{\boldsymbol{x}}) \le \max_{\boldsymbol{x} \in \mathcal{F}} \mathrm{CEI}(\tilde{\boldsymbol{x}}, \boldsymbol{x}).$$

Furthermore, $\check{\boldsymbol{x}} \in \mathcal{F}$ *because* $\acute{\boldsymbol{x}} \notin \arg\max_{\boldsymbol{x} \in \mathcal{U} \setminus \mathcal{F}} \mathrm{CEI}(\tilde{\boldsymbol{x}}, \boldsymbol{x})$, *otherwise.*

When the number of feasible solutions is large, the condition in Corollary 2 holds almost all the time because considering the large size of $\mathcal{U}$, for all $\acute{\boldsymbol{x}} \in \arg\max_{\boldsymbol{x} \in \mathcal{U} \setminus \mathcal{F}} \mathrm{CEI}(\tilde{\boldsymbol{x}}, \boldsymbol{x})$, it is very unlikely to simulate every solution $\check{\boldsymbol{x}}$ with $\acute{\boldsymbol{x}}^{(\varrho)} = \check{\boldsymbol{x}}^{(\varrho)}$ for all $\varrho \in \mathcal{G} \setminus \{\rho, g\}$ and $\check{\boldsymbol{x}}^{(\rho)} \in \mathcal{F}^{(\rho)}(\acute{\boldsymbol{x}}^{(\rho)})$, for some group $\rho \in \mathcal{G}^{(-g)}$. Therefore, Corollary 2 implies that it suffices to calculate the CEIs of solutions in $\mathcal{D} \cup \mathcal{F}$ to find a solution with the largest CEI in $\mathcal{X}$ because for some solution with the largest CEI in $\mathcal{X} \setminus (\mathcal{D} \cup \mathcal{F}) = \mathcal{U} \setminus \mathcal{F}$, there exists a solution in $\mathcal{F}$ with a larger CEI. Thus, a solution with the largest CEI in $\mathcal{F}$ has a relatively large—typically the largest—CEI in $\mathcal{U}$. Further, following Relation (R1), the CEIs of solutions in $\mathcal{F} \subseteq \mathcal{U}$ with the same first $g-1$ components are identical. Therefore, it is sufficient to calculate the CEIs of solutions in $\mathcal{D} \cup \check{\mathcal{F}}$ to obtain promising first $g-1$ components, where $\check{\mathcal{F}} \subseteq \mathcal{F}$ includes only one solution for each first $g-1$ components. Since $\mathcal{D} \cup \check{\mathcal{F}}$ is much smaller than $\mathcal{X}$, the reduction in computational overhead is huge, especially when the number of feasible solutions is very large; computational analysis conducted in Section 5 shows this tremendous benefit.

Despite the significant computational savings from using Pareto frontiers, DASSO stills encounters a limit on the problem size it can handle because the size of $\check{\mathcal{F}}$ increases exponentially in the number of groups. To extend DASSO beyond this limit, a heuristic approach can keep the number of CEI calculations computationally feasible. In particular, $\mathcal{F}$ can be alternatively constructed as $\big\{\boldsymbol{x} \in \mathcal{U} : \boldsymbol{x}^{(\rho)} \in \acute{\mathcal{F}}^{(\rho)},\ \forall \rho \in \mathcal{G}^{(-g)}\big\}$, where $\acute{\mathcal{F}}^{(\rho)}$ is some subset of $\mathcal{X}^{(\rho)}$ chosen heuristically at a desired size. We employ this approach in our numerical experiments in Section 5 when $\big\{\boldsymbol{x} \in \mathcal{U} : \boldsymbol{x}^{(\rho)} \in \mathcal{F}^{(\rho)},\ \forall \rho \in \mathcal{G}^{(-g)}\big\}$ is too large to handle directly.

Computing CEIs requires the diagonal elements of $[\bar{\boldsymbol{Q}}]^{-1}$, the column of $[\bar{\boldsymbol{Q}}]^{-1}$ corresponding to $\tilde{\boldsymbol{x}}$, and $\boldsymbol{m}^{(\rho)}$, for each $\rho \in \mathcal{G}$. Instead of inverting $\bar{\boldsymbol{Q}}$ to obtain these elements, we use the block matrix inversion

formula (Lemma EC.2 in Section EC.1 in the e-companion) to compute the elements more efficiently. Letting $\check{\boldsymbol{\Sigma}}_{DD} = \boldsymbol{\Sigma}_{DD} + \boldsymbol{\Sigma}^{\epsilon}$, we have

$$[\bar{\boldsymbol{Q}}]^{-1} = \begin{pmatrix} [\boldsymbol{Q}^{(\rho)}_{\mathcal{U}\mathcal{U}}]^{-1} + [\boldsymbol{Q}^{(\rho)}_{\mathcal{U}\mathcal{U}}]^{-1}\boldsymbol{Q}^{(\rho)}_{\mathcal{U}D}\boldsymbol{S}\boldsymbol{Q}^{(\rho)}_{D\mathcal{U}}[\boldsymbol{Q}^{(\rho)}_{\mathcal{U}\mathcal{U}}]^{-1} & -[\boldsymbol{Q}^{(\rho)}_{\mathcal{U}\mathcal{U}}]^{-1}\boldsymbol{Q}^{(\rho)}_{\mathcal{U}D}\boldsymbol{S} \\ -\boldsymbol{S}\boldsymbol{Q}^{(\rho)}_{D\mathcal{U}}[\boldsymbol{Q}^{(\rho)}_{\mathcal{U}\mathcal{U}}]^{-1} & \boldsymbol{S} \end{pmatrix},$$

and thus

$$\boldsymbol{m}^{(\rho)} = \begin{pmatrix} -[\boldsymbol{Q}^{(\rho)}_{\mathcal{U}\mathcal{U}}]^{-1}\boldsymbol{Q}^{(\rho)}_{\mathcal{U}D}\boldsymbol{S}[\boldsymbol{T}^{(\rho)}_{DD}]^{\top}\boldsymbol{E}\left(\bar{\boldsymbol{Y}}_D - \boldsymbol{\mu}_D\right) \\ \boldsymbol{S}[\boldsymbol{T}^{(\rho)}_{DD}]^{\top}\boldsymbol{E}\left(\bar{\boldsymbol{Y}}_D - \boldsymbol{\mu}_D\right) \end{pmatrix},$$

where $\boldsymbol{S} = \boldsymbol{\Sigma}^{(\rho)}_{DD} - \boldsymbol{\Sigma}^{(\rho)}_{DD}[\boldsymbol{T}^{(\rho)}_{DD}]^{\top}\check{\boldsymbol{\Sigma}}^{-1}_{DD}\boldsymbol{T}^{(\rho)}_{DD}\boldsymbol{\Sigma}^{(\rho)}_{DD}$ is the inverse of the Schur complement of $\boldsymbol{Q}^{(\rho)}_{\mathcal{U}\mathcal{U}}$ in $[\bar{\boldsymbol{Q}}]^{-1}$, for $\rho \in \mathcal{G}^{(-g)}$. Also, we have

$$[\bar{\boldsymbol{Q}}^{(g)}]^{-1} = \begin{pmatrix} \sigma_g^2\mathbf{I}_{|\mathcal{U}|} & \mathbf{0}_{|\mathcal{U}|\times|D|} \\ \mathbf{0}_{|D|\times|\mathcal{U}|} & \boldsymbol{S}^{(g)} \end{pmatrix}, \text{ and thus } \boldsymbol{m}^{(g)} = \begin{pmatrix} \vec{\mathbf{0}}_{|\mathcal{U}|} \\ \boldsymbol{S}^{(g)}\boldsymbol{E}^{(g)}\left(\bar{\boldsymbol{Y}}_D - \boldsymbol{\mu}_D\right) \end{pmatrix},$$

where $\boldsymbol{S}^{(g)} = \sigma_g^2\mathbf{I}_{|D|} - \left(\sigma_g^2\right)^2\check{\boldsymbol{\Sigma}}^{-1}_{DD}$ is the inverse of the Schur complement of $\boldsymbol{Q}^{(g)}_{\mathcal{U}\mathcal{U}}$ in $[\bar{\boldsymbol{Q}}^{(g)}]^{-1}$, making $\boldsymbol{S}^{(g)}\boldsymbol{E}^{(g)} = \mathbf{I}_{|D|} - \frac{1}{\sigma_g^2}\boldsymbol{S}^{(g)}$. Further, to compute $\boldsymbol{E}$ more efficiently for $\rho \in \mathcal{G}^{(-g)}$, we use the Woodbury matrix identity (Lemma EC.1 in Section EC.1 in the e-companion):

$$\boldsymbol{E} = \check{\boldsymbol{\Sigma}}^{-1}_{DD} + \check{\boldsymbol{\Sigma}}^{-1}_{DD}\boldsymbol{T}^{(\rho)}_{DD}\left([\boldsymbol{\Sigma}^{(\rho)}_{DD}]^{-1} - [\boldsymbol{T}^{(\rho)}_{DD}]^{\top}\check{\boldsymbol{\Sigma}}^{-1}_{DD}\boldsymbol{T}^{(\rho)}_{DD}\right)^{-1}[\boldsymbol{T}^{(\rho)}_{DD}]^{\top}\check{\boldsymbol{\Sigma}}^{-1}_{DD}.$$

This is more efficient than inverting $\boldsymbol{\Sigma}_{DD} - \boldsymbol{T}^{(\rho)}_{DD}\boldsymbol{\Sigma}^{(\rho)}_{DD}[\boldsymbol{T}^{(\rho)}_{DD}]^{\top} + \boldsymbol{\Sigma}^{\epsilon}$ of size $|D| \times |D|$ for each $\rho \in \mathcal{G}^{(-g)}$ because it requires inverting two matrices of size $|D^{(\rho)}| \times |D^{(\rho)}|$ for each $\rho \in \mathcal{G}^{(-g)}$ in addition to inverting $\check{\boldsymbol{\Sigma}}_{DD}$ of size $|D| \times |D|$ only once. Since $|D^{(\rho)}|$ is bounded by $n^{(\rho)}$ and $|D|$, and the increase on $|D^{(\rho)}|$ is more slowly than the increase on $|D|$ as the algorithm iterates, the benefit is greater in the later stages.

5. Numerical experiments

In this section, we empirically evaluate the performance of DASSO, investigate its sensitivity to properties of the objective function, and explore the importance of the chosen decomposition. A key feature of the DASSO prior is that it is consistent even with non-separable objective functions, but one might expect that a separable objective function, if one knows the proper separation, would be favorable.

For the numerical experiments we consider four examples: The first two examples are optimization of the high-dimensional Zakharov and Styblinski-Tang test functions with added stochastic noise. The third is a multi-product inventory problem which can be considered as a practical DSO problem with knowledge about a natural decomposition. The last is a stylized problem allowing control of how separable the objective function is. Recall that lack of separability—i.e., the contribution of the remainder term in the decomposition—is addressed in the slice stage, so these experiments assess the effectiveness of slicing. While these problems are all high-dimensional with large numbers of feasible solutions, they are not computationally expensive simulations. This allows us to run many long macroreplications that show convergence behaviors, which we would not expect to do in practice. Therefore, our computation-time analysis is effectively an assessment of algorithm overhead, not simulation time.

In all experiments the parameters of the GMRF are estimated from s initial design points chosen by Latin hypercube sampling and sg additional design points selected to augment these s initial design points; this is the minimum number of additional points needed to parameterize the g groups plus remainder prior. See Section EC.5 in the e-companion for details of parameter estimation. All $s(g+1)$ design points are simulated for r_0 replications. However, for better comparison in different settings, the DASSO search is initialized from only the s initial design points, which are common with the same simulation outputs for all settings. Each time a solution is selected, it is simulated r_d replications if it has been simulated previously and r_u replications otherwise. For the experiments, $s = 100$ and $r_0 = r_d = r_u = 10$ in Section 5.1, $s = 200$ and $r_0 = r_d = r_u = 10$ in Section 5.2, and $s = 15$, $r_0 = 20$, $r_d = 4$, and $r_u = 10$ in Sections 5.3 and 5.4. These are largely arbitrary choices and there is likely room for a more data-driven selections, a topic we reserve for future work. All computations are executed on a desktop computer with a Windows 10 operating system, a 2.9 GHz Intel Core i7 CPU, 32 GB of RAM, 8 cores and 16 logical processors.

5.1. Well-known test functions

The k-dimensional Zakharov function $\boldsymbol{y}(x_1, x_2, \ldots, x_k) = \sum_{i=1}^{k} x_i^2 + \left(\sum_{i=1}^{k} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{k} 0.5ix_i\right)^4$ is often used as a test case in BO as it presents a very challenging problem. The function is minimized at $\boldsymbol{x} = (0, 0, \ldots, 0)^{\top}$. Letting $\mathcal{X} = \{-2, -1, 0, 1, 2\}^{10}$ be the set of feasible solutions, the 10-dimensional Zakharov function takes values between $[0, 9.15 \times 10^6]$ and has around 9.8 million feasible solutions. To make the problem stochastic, we add zero-mean normally distributed noise with variance 1.8^2 to objective function values. We also considered different levels of stochastic noise with variances 2.6^2, 3.9^2, and 5.2^2, but chose not to exhibit the results because they show similar performance.

The k-dimensional Styblinski-Tang function $\boldsymbol{y}(x_1, x_2, \ldots, x_k) = \frac{1}{20}\sum_{i=1}^{k}(x_i^4 - 16x_i^2 + 5x_i)$ is another standard test case. Letting $\mathcal{X} = \{-6, -3, 0, 3, 6\}^{10}$ be the set of feasible solutions, the 10-dimensional Styblinski-Tang function takes values between $[-39, 375]$ and is minimized at $\boldsymbol{x} = (-3, -3, \ldots, -3)^{\top}$. We add zero-mean normally distributed noise with variance 3^2 to objective function values.

We use these two problems to compare the performance of DASSO to pGMIA proposed by Li and Song (2024), the current state-of-the-art for high-dimensional DSO problems, and Bounce (**B**ayesian **o**ptimization **u**sing **in**creasingly high-dimensional **c**ombinatorial and continuous **e**mbeddings) proposed by Papenmeier et al. (2023), a high-dimensional BO algorithm optimizing over combinatorial, continuous, or mixed spaces.

pGMIA was shown empirically to outperform the multi-resolution GMIA of Salemi et al. (2019) and four state-of-the-art high-dimensional BO algorithms: Random EMbedding Bayesian Optimization (Wang et al. 2016), Sparse Axis-Aligned Subspace Bayesian Optimization (Eriksson & Jankowiak 2021), and High-Dimensional Bayesian Optimization and High-Dimensional Batch Bayesian Optimization (Wang et al. 2017). Thus, comparing DASSO with pGMIA indirectly assesses DASSO against these algorithms.

Unlike DASSO, which decomposes the prior distribution into an additive form, pGMIA batches the dimensions into two layers and hierarchically optimizes each layer by projecting one layer onto the other. We note that pGMIA is implemented in MATLAB while DASSO is implemented in Python. Since pGMIA with the random projection criterion (pGMIA-R) is observed to empirically outperform other benchmarks, we compare DASSO to that version. In their numerical experiments, Li and Song (2024) set the numbers of region- and solution-layer dimensions to 5 for pGMIA. The closest equivalence for DASSO is a decomposition consisting of two 5-dimensional groups. Let $\mathcal{G}_5$ denote such a decomposition with the sets of component indices being $\mathrm{g}^{(1)} = \{1, 2, 3, 4, 5\}$ and $\mathrm{g}^{(2)} = \{6, 7, 8, 9, 10\}$. This assignment of component indices to groups is arbitrary, and when we tried other assignments we observed no difference in performance.

Papenmeier et al. (2023) demonstrate that Bounce empirically outperforms five state-of-the-art algorithms designed for combinatorial, continuous, or mixed input domains: BO with Dictionaries (Deshwal et al. 2023), CAtegorical Spaces, or Mixed, OPtimisatiOn with Local-trust-regIons & TAilored Non-parametric (Wan et al. 2021), COMbinatorial Bayesian Optimization (Oh et al. 2019), Sequential Model-based Algorithm Configuration (Hutter et al. 2011), and Random Decomposition Upper-Confidence Bound (Ziomek & Ammar 2023). To address the computational burden of high-dimensional BO, Bounce defines a GP surrogate in an iteratively refined lower-dimensional subspace called the target space, and maximizes the acquisition function within promising regions of the target space by using a trust-region-based method. Bounce is also implemented in Python.

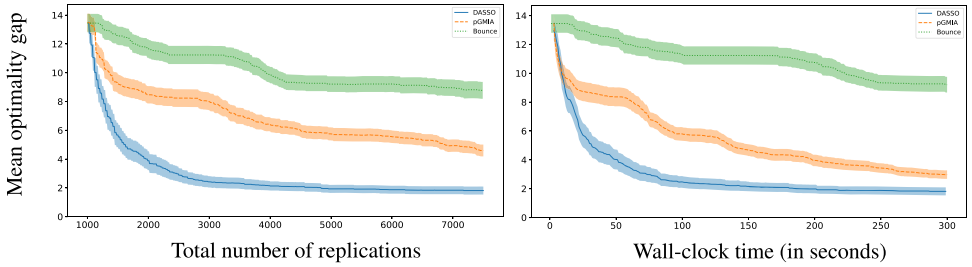


Fig. 3. Mean optimality gap of the Zakharov function vs. total number of replications and wall-clock time across 100 macro-replications for DASSO (blue solid line) with decomposition $\mathcal{G}_5$, pGMIA (orange dashed line), and Bounce (green dotted line). The shaded area around each curve shows point-wise ± 2 standard error of the average. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

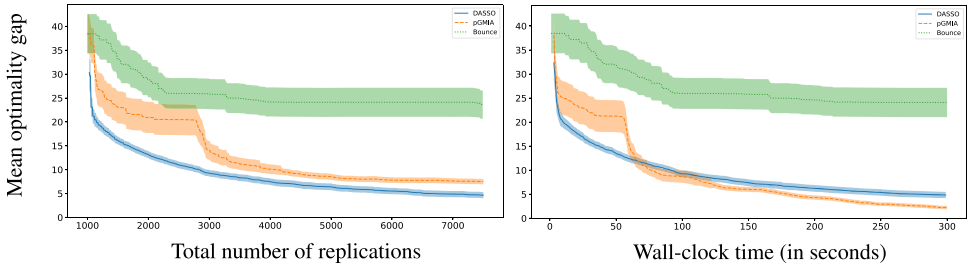


Fig. 4. Mean optimality gap of the Styblinski-Tang function vs. total number of replications and wall-clock time across 100 macro-replications for DASSO (blue solid line) with decomposition $\mathcal{G}_5$, pGMIA (orange dashed line), and Bounce (green dotted line). The shaded area around each curve shows point-wise ± 2 standard error of the average. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Bounce is not designed to handle stochastic output, unlike DASSO and pGMIA that are specifically created to be effective on such problems. Therefore, whenever a solution is queried by Bounce, we run 10 replications and treat their average as deterministic response. To align Bounce as closely as possible to DASSO with decomposition $\mathcal{G}_5$, we set its initial target dimensionality to 5.

Applications of large-scale, high-dimensional DSO typically exist between two extremes: Simulations that execute so slowly that even substantial optimization algorithm overhead is negligible relative to simulation time, and simulations that execute so fast that algorithm overhead is the bottleneck to computational feasibility. Tongarlak et al. (2010) is an example of the former (20 replications of a single feasible solution took 8 hours), and the Zakharov and Styblinski-Tang functions are examples of the latter. DASSO is designed to be effective when the number of feasible solutions or replications to simulate is computationally limited (i.e., the simulation is very slow), while imposing much less algorithm overhead than other methods.

To see this, Figs. 3 and 4 show the mean optimality gap of the 10-dimensional Zakharov and Styblinski-Tang functions, respectively, as the distance from the optimal value versus the total number of simulation replications (left) and versus the wall-clock time (right), averaged over 100 macro-replications of DASSO, pGMIA, and Bounce. When each simulation replication takes significant time it is more critical to make rapid improvement in the objective value with fewer replications. On the other hand, when each simulation replication is very fast, the computational overhead of the algorithm matters. The figures show that DASSO outperforms pGMIA and Bounce in terms of progress per replication for both Zakharov and Styblinski-Tang. In terms of progress per second, DASSO is the best for Zakarov and slightly lags pGMIA for Styblinski-Tang after a faster start. Since the experimental settings are the same, the difference in performance between these test functions can be attributed to different objective functions. For our target applications of DASSO the initial rapid decrease in the optimality gap is what we desire; we do not expect to achieve anything like convergence when the simulation is high-dimensional and computationally expensive.

5.2. A higher-dimensional test function

To evaluate the performance of DASSO in problems of significantly higher dimension, we compare it to pGMIA+ (Li & Song 2024), a multi-layer extension of pGMIA for higher-dimensional problems. We test the 100-dimensional Zakharov function with $\mathcal{X} = \{-5, -4, \dots, 5\}^{100}$ which takes values between $[0, 2.54 \times 10^{16}]$. To make the problem stochastic, we add zero-mean normally distributed noise with variance 1.8^2 to objective function values.

In numerical experiments for pGMIA+, Li and Song (2024) set the numbers of dimensions in the first two layers to 3 leaving the 94 dimensions to the last. The sampling decisions in the first two layers are made by maximizing the CEI while the coordinates in the remaining 94 dimensions are selected randomly. The closest correspondence for DASSO is a decomposition consisting of thirty-two 3-dimensional groups and two 2-dimensional groups. Let $\mathcal{G}_3$ denote such a decomposition with the sets of component indices being $\mathrm{g}^{(i)} = \{3i-2, 3i-1, 3i\}$ for $i = 1, 2, \dots, 32$, $\mathrm{g}^{(33)} = \{97, 98\}$, and $\mathrm{g}^{(34)} = \{99, 100\}$. Furthermore, since the problem size makes the exact CEI maximization computationally infeasible in each dice stage, $\mathcal{F}$ is heuristically constructed as $\left\{\boldsymbol{x} \in \mathcal{U} : \boldsymbol{x}^{(\rho)} \in \acute{\mathcal{F}}^{(\rho)}, \forall \rho \in \mathcal{G}^{(-\mathrm{g})}\right\}$, where $\acute{\mathcal{F}}^{(\rho)}$ is the Pareto frontier $\mathcal{F}^{(\rho)}$ for two randomly chosen ρs in $\mathcal{G}^{(-\mathrm{g})}$ and a single randomly chosen point from $\mathcal{X}^{(\rho)}$ for the remaining thirty-one groups. This is a heuristic modification with the goal to demonstrate feasibility for DASSO to tackle even higher-dimensional problems.

Fig. 5 shows the mean optimality gap of the 100-dimensional Zakharov function as the distance from the optimal value versus the total number of simulation replications (left) and versus the wall-clock time (right), averaged over 100 macro-replications of DASSO and pGMIA+. In terms of progress per replication in this example, DASSO and pGMIA+ outperform each other at different points while in the later iterations, pGMIA+ appears to have smaller optimality gap on average (although statistically indistinguishable from DASSO) with smaller run-to-run variability. Considering the range of the function, 2.54×10^{16}, the optimality gap (approximately 1000) after 7500 replications is

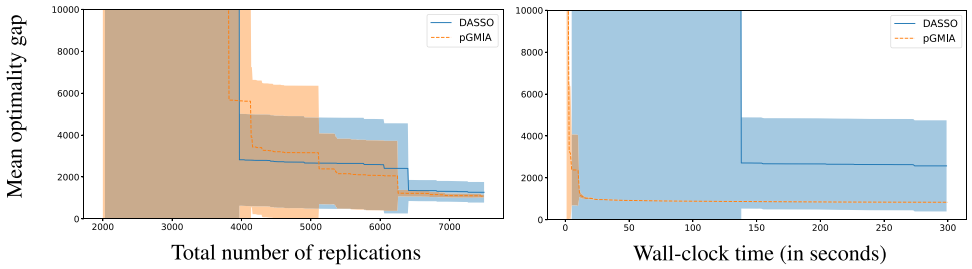


Fig. 5. Mean optimality gap of the 100-dimensional Zakharov function vs. total number of replications and wall-clock time across 100 macro-replications for DASSO (blue solid line) with decomposition $\mathcal{G}_3$ and pGMIA+ (orange dashed line). The shaded area around each curve shows point-wise ±2 standard error of the average. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

negligible for both algorithms. In terms of progress per second, the large number of groups in DASSO causes computational overhead, making the algorithm iterate slower than pGMIA+. If *simulation* overhead was more substantial, rather than negligible as it is for simulating the Zakharov function, the difference would be less.

5.3. Inventory problem

Consider a multi-product inventory problem where each product follows an (s, S) policy, i.e., once the inventory level of product ρ falls below its reorder point, s^ρ, an order is given to bring the inventory level up to S^ρ. This problem is a variant of the single-product version in Salemi et al. (2019), where the product is subject to periodic demand that follows a Poisson distribution and the optimal policy is $(s, S-s) = (18, 35)$ under the objective of minimizing the expected average cost per period over a fixed planning horizon. Similarly, in our version, demand for each product follows a common Poisson distribution and the objective is to minimize the expected value of cost defined as $\boldsymbol{y}(\boldsymbol{x}) = \sum_{\rho\in\mathcal{G}} \boldsymbol{y}^{(\rho)}(\boldsymbol{x}^{(\rho)}) + \boldsymbol{y}^{(r)}(\boldsymbol{x})$, where $\boldsymbol{y}^{(\rho)}(\boldsymbol{x}^{(\rho)})$ represents the expected average cost per period for product ρ and $\boldsymbol{y}^{(r)}(\boldsymbol{x})$ is a deterministic interaction term that depends on how much feasible solution $\boldsymbol{x}$ deviates from the optimal solution. In other words, we induce an interaction so that the problem is not separable. Therefore, the optimal policy for each product is $(s^\rho, S^\rho - s^\rho) = (18, 35)$, and the interaction term is $\boldsymbol{y}^{(r)}(\boldsymbol{x}) = \prod_{\rho\in\mathcal{G}} \sqrt{(s^\rho - 18)^2 + (S^\rho - s^\rho - 35)^2}$. To treat the feasible region as a hyperbox defined on the integer lattice, we let $\boldsymbol{x}^{(\rho)} = (s^\rho, S^\rho - s^\rho)$ for $\rho \in \mathcal{G}$ and $\boldsymbol{x} = \{(s^\rho, S^\rho - s^\rho)\}_{\rho\in\mathcal{G}}$.

The natural decomposition for this multi-product inventory problem is that each group represents a product, and thus is 2-dimensional, one dimension is for s^ρ and the other is for $S^\rho - s^\rho$. We consider a 5-product (i.e., 10-dimensional) problem where the feasible region is defined by constraints $10 \le s^\rho \le 34$ and $20 \le S^\rho - s^\rho \le 44$, which leads to $n = 25^{10} = 95{,}367{,}431{,}640{,}625$ feasible solutions. In the experiments, we set the number of macro-replications to 30 and the total number of replications to 7500, i.e., DASSO terminates after obtaining this number of simulation outputs.

Fig. 6(a) depicts the mean optimality gap as a percentage versus total number of replications for 30 macro-replications of the inventory problem with the natural decomposition. Before DASSO starts performing dice and slice stages, the mean optimality gap of the sample-best solution among the first $s = 15$ initial design points is around 85 %; recall that these design points are chosen by Latin hypercube sample and simulated for 20 replications. Clearly DASSO obtains rapid improvement: it only takes exploring around 145 (or 38) design points and simulating 2500 (or 650) replications in total to have the mean optimality gap below 1.5 % (or 5 %). Considering the very large scale of the problem, with more than 95 trillion feasible solutions, the design points explored by the algorithm to obtain such small optimality gaps is an amazingly tiny portion of the feasible solution set.

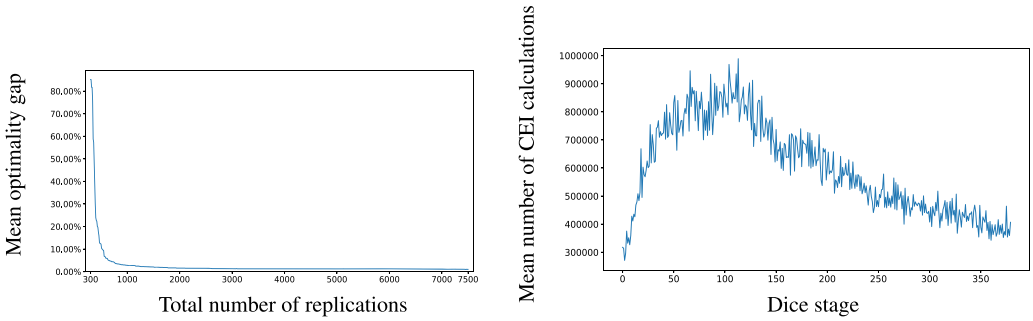
Recall from Section 4.6 that rather than performing CEI calculation for all 95 trillion feasible solutions at each dice stage, it is sufficient to calculate the CEIs of solutions from the Pareto set $\mathcal{D} \cup \check{\mathcal{F}}$. To illustrate how much smaller $|\mathcal{D} \cup \check{\mathcal{F}}|$ is than the number of feasible solutions $n = |\mathcal{X}|$, Fig. 6(b) shows the mean number of CEI calculations performed, i.e., $|\mathcal{D} \cup \check{\mathcal{F}}|$, at each dice stage averaged over 30 macro-replications of the inventory problem with the natural decomposition. Although the mean number of CEI calculations varies from one dice stage to another, it is no greater than 989 thousand, which is around one hundred-millionth of the number of feasible solutions, i.e., $|\mathcal{D} \cup \check{\mathcal{F}}| \le 10^{-8} n$. Without such a reduction in the number of CEI calculations it would be computationally impossible to solve a problem of this scale.

The mean total CPU times spent to perform three different tasks—simulation execution, slice stage, and dice stage—averaged over 30 macro-replications of the inventory problem with the natural decomposition are as follows; recall that the algorithm stops after simulating 7500 replications. It takes around 347 CPU seconds (5.8 CPU minutes) on average to run a single macro-replication: 251 seconds for the dice stage, 79 seconds for the slice stage, and 17 seconds for the simulation executions. The dice and slice stage times are independent of the simulation time, which is abnormally small in this constructed example. In other words, even if the simulation took orders of magnitude longer the DASSO algorithm overhead would be the same. These results indicate that DASSO is not only effective but also efficient to solve very large-scale problems.

The results above are for the natural decomposition, where each group represents a product. To explore the importance of decomposition, in addition to the natural one, we consider 10 alternative decompositions with five 2-dimensional groups; notice that the decompositions are the same size. These additional decompositions are different from the natural one in a way that at least one group represents either $(s^\rho, S^{\rho(\varrho)} - s^{\rho(\varrho)})$, or $(s^\rho, s^{\rho(\varrho)})$, or $(S^\rho - s^\rho, S^{\rho(\varrho)} - s^{\rho(\varrho)})$, for some $\rho \ne \varrho$. Fig. 7 illustrates the mean optimality gap as a percentage versus total number of replications for 30 macro-replications of the inventory problem with various decompositions. The results indicate that the performance of DASSO with some alternative decompositions is statistically indistinguishable from that achieved with the natural one, suggesting that an alternative decomposition can be used when there is no knowledge about a natural decomposition. In the next section, we further explore the importance of decomposition.

5.4. Controlled test function

Consider the objective function $\boldsymbol{y}(\boldsymbol{x}) = (1-\alpha)\sum_{\rho\in\mathcal{G}} \boldsymbol{y}^{(\rho)}(\boldsymbol{x}^{(\rho)}) + \alpha\lambda\boldsymbol{y}^{(r)}(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{X}$, where $\mathcal{X} = \{\underline{x}, \underline{x}+1, \dots, \overline{x}-1, \overline{x}\}^d$ for some integers $\underline{x}$ and $\overline{x}$ and dimension d. The role of $\alpha \in [0, 1]$ is to control how close to separable the objective function is, or equivalently, how much the remainder matters. Let each $\boldsymbol{y}^{(\rho)}(\cdot)$ for $\rho \in \mathcal{G}$ and $\boldsymbol{y}^{(r)}(\cdot)$ be an inverted multivariate normal density function with a shift, generically defined as $f(x_1, x_2, \dots, x_k) = -\gamma_1 \exp\left\{-\gamma_2 \sum_{i=1}^k i x_i^2\right\} + \gamma_1$, where $\gamma_1 = 1000$ and $\gamma_2 = 0.001$; notice that $k = d^{(\rho)}$ for each $\rho \in \mathcal{G}$ and $k = d$ for the remainder term. The function is minimized at $x_i = 0$ for $i = 1, 2, \dots, k$, and thus, $\boldsymbol{y}(\boldsymbol{x})$ is also minimized at $\boldsymbol{x} = (0, 0, \dots, 0)^\top$. The



(a) Effectiveness (b) Efficiency

Fig. 6. (a) Mean optimality gap vs. total number of replications and (b) mean number of CEI calculations at each dice stage across 30 macro-replications for the natural decomposition in the inventory problem.

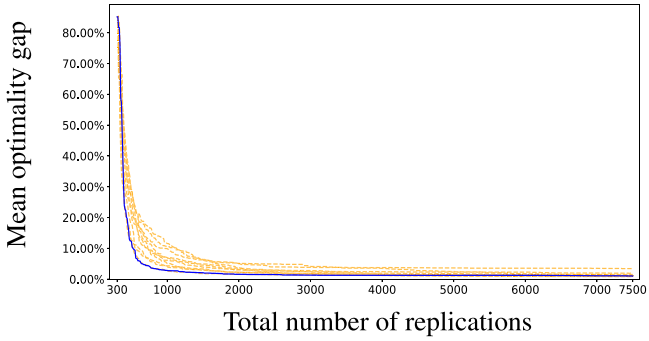


Fig. 7. Mean optimality gap vs. total number of replications across 30 macro-replications for 11 different decompositions in the inventory problem. The blue solid line represents the natural decomposition while each orange dashed line represent an alternative decomposition. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

shifting term makes the optimal objective function value 0 regardless of the value of α. To have the same range of $\boldsymbol{y}(\cdot)$ for any α, we set $\lambda = \sum_{\rho \in \mathcal{G}} \boldsymbol{y}^{(\rho)}(\boldsymbol{x}^{(\rho)\star})/\boldsymbol{y}^{(r)}(\boldsymbol{x}^{\star})$ with solution $\boldsymbol{x}^{\star}$ that maximizes the objective function. We consider a 12-dimensional problem with $d^{(\rho)} = 2$ for $\rho \in \mathcal{G} = \{1, 2, \ldots, 6\}$, where the sets of component indices are $\mathrm{g}^{(1)} = \{1, 2\}$, $\mathrm{g}^{(2)} = \{3, 4\}$, $\mathrm{g}^{(3)} = \{5, 6\}$, $\mathrm{g}^{(4)} = \{7, 8\}$, $\mathrm{g}^{(5)} = \{9, 10\}$, and $\mathrm{g}^{(6)} = \{11, 12\}$; recall that $\boldsymbol{x}^{(\rho)} = (x_i)_{i \in \mathrm{g}^{(\rho)}}$. Letting $\underline{x} = -2$ and $\overline{x} = 2$, the objective function values range between $[0, 71.57]$, and the total number of feasible solutions is $n = 5^{12} = 244{,}140{,}625$. To make the problem stochastic, a zero-mean normally distributed noise with variance 3^2 is added to the objective function values.

Slightly abusing notation, let $\mathcal{G}_2$ denote the decomposition that consists of $\mathrm{g}^{(1)} = \{1, 2\}$, $\mathrm{g}^{(2)} = \{3, 4\}$, $\mathrm{g}^{(3)} = \{5, 6\}$, $\mathrm{g}^{(4)} = \{7, 8\}$, $\mathrm{g}^{(5)} = \{9, 10\}$, and $\mathrm{g}^{(6)} = \{11, 12\}$; we use the subscript to indicate that each group is 2-dimensional. We treat $\mathcal{G}_2$ as a natural decomposition since it defines the objective function. We also consider three alternative decompositions: $\mathcal{G}_1$ consists of $\mathrm{g}^{(\rho)} = \{\rho\}$ for $\rho = 1, 2, \ldots, 12$; $\mathcal{G}_3$ consists of $\mathrm{g}^{(1)} = \{1, 2, 3\}$, $\mathrm{g}^{(2)} = \{4, 5, 6\}$, $\mathrm{g}^{(3)} = \{7, 8, 9\}$, and $\mathrm{g}^{(4)} = \{10, 11, 12\}$; and finally $\mathcal{G}_4$ consists of $\mathrm{g}^{(1)} = \{1, 2, 3, 4\}$, $\mathrm{g}^{(2)} = \{5, 6, 7, 8\}$, and $\mathrm{g}^{(3)} = \{9, 10, 11, 12\}$. $\mathcal{G}_4$ is also a natural decomposition because there is no interaction among groups.

Fig. 8 depicts the mean optimality gap as a distance from the optimal value versus total number of replications averaged over 50 macro-replications of the stylized problem for $\alpha \in \{0, 0.5, 1\}$ with decompositions $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$, and $\mathcal{G}_4$. Although $\mathcal{G}_4$ shows rapid improvement in the early stages of the algorithm, it fails to close the gap as fast as the other decompositions. On the other hand, $\mathcal{G}_1$ performs the best. Even for $\alpha = 0$, where the objective function value is fully separable, the natural decompositions $\mathcal{G}_2$ and $\mathcal{G}_4$ do not show a better performance than $\mathcal{G}_1$. For $\alpha = 1$, where the objective function value is not even approximately separable, DASSO still performs well with a good choice of decomposition, such as $\mathcal{G}_1$, $\mathcal{G}_2$ or $\mathcal{G}_3$.

Table 1
Mean CPU times (in seconds) spent for performing three different tasks across 50 macro-replications for 4 different decompositions in the stylized problem with $\alpha = 0$. The percentages in parentheses represent the proportion of each task to total time.

Task	$\mathcal{G}_1$	$\mathcal{G}_2$	$\mathcal{G}_3$	$\mathcal{G}_4$
Simulation	1.5 (0.2 %)	2.3 (4.5 %)	2.5 (3.9 %)	4.2 (2.5 %)
Slice stage	4.6 (0.7 %)	10.3 (20.0 %)	15.2 (24.0 %)	80.8 (47.7 %)
Dice stage	698.7 (99.1 %)	38.9 (75.5 %)	45.7 (72.1 %)	84.5 (49.8 %)
All	704.8 (100 %)	51.5 (100 %)	63.4 (100 %)	169.5 (100 %)

We note that the discussion above ignores the computation time overhead of DASSO and focuses on the total number of replications. One might expect that decompositions with smaller-size groups are computationally faster, and thus $\mathcal{G}_1$ should be the ideal decomposition. However, this is not the case based on Table 1 because $\mathcal{G}_1$ is much slower than the other decompositions. The table exhibits the mean CPU times spent to perform three different tasks—simulation execution, slice stage and dice stage—as well as the total time across 50 macro-replications for decompositions $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$, and $\mathcal{G}_4$ in the stylized problem with $\alpha = 0$; recall that the algorithm stops after simulating 7500 replications.

The time spent to perform the slice stages increases as the size of the groups increases, whereas that of the dice stages does not show the same trend. This is mainly because $\mathcal{G}_1$ requires many more CEI calculations: the mean number of CEI calculations in each dice stage varies between 72 thousand to 5 million for $\mathcal{G}_1$, 3.7 thousand to 49 thousand for $\mathcal{G}_2$, 1.3 thousand to 5.4 thousand for $\mathcal{G}_3$, and 103 to 1.2 thousand for $\mathcal{G}_4$. This result also illustrates the importance of the reduction of the number of CEI calculations.

To further investigate the importance of decomposition, we consider two more decompositions with the same size as $\mathcal{G}_2$, i.e., they also consists

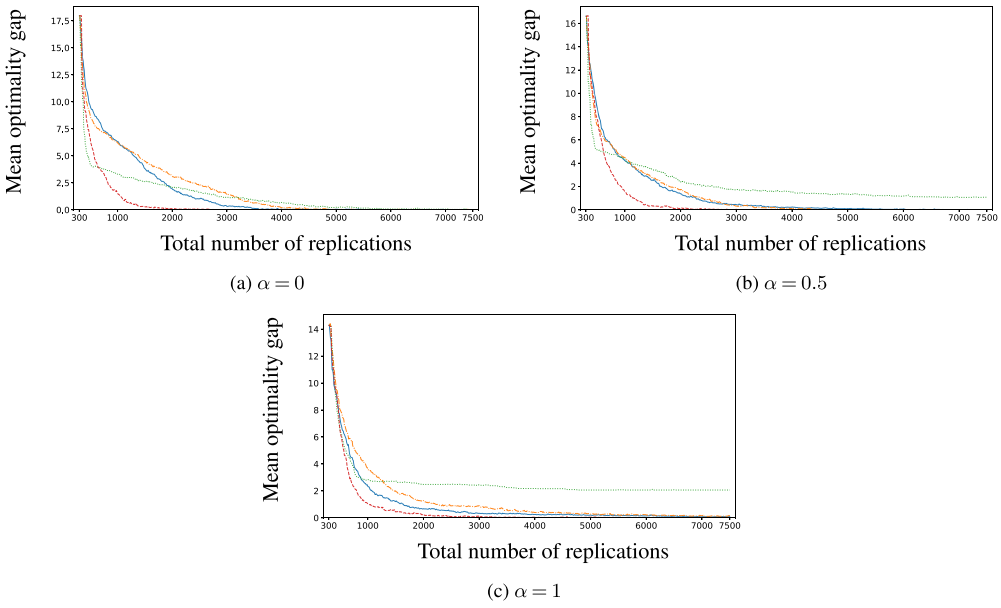


Fig. 8. Mean optimality gap vs. total number of replications across 50 macro-replications for 4 different decompositions in the stylized problem with various α values. The red dashed, blue solid, orange dash-dotted, and green dotted lines represent $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$, and $\mathcal{G}_4$, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

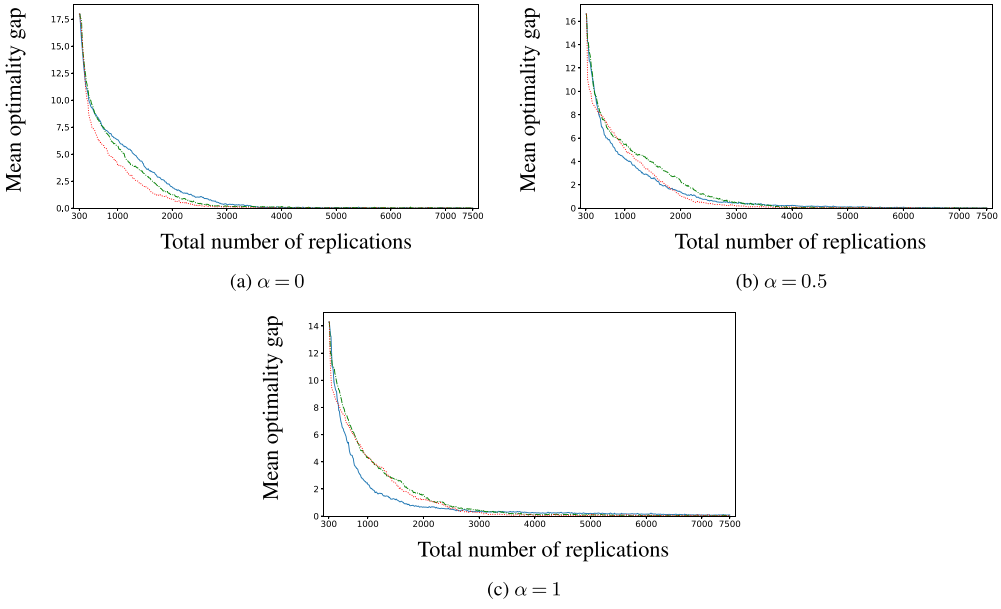


Fig. 9. Mean optimality gap vs. total number of replications across 50 macro-replications for 3 different decompositions in the stylized problem with various α values. The blue solid, green dash-dotted, and red dotted lines represent $\mathcal{G}_2$, $\hat{\mathcal{G}}_2$, and $\tilde{\mathcal{G}}_2$, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

of six 2-dimensional groups. The sets of component indices for the first one, denoted by $\hat{\mathcal{G}}_2$, are $g^{(1)} = \{1, 10\}$, $g^{(2)} = \{5, 8\}$, $g^{(3)} = \{2, 7\}$, $g^{(4)} = \{4, 11\}$, $g^{(5)} = \{3, 9\}$, and $g^{(6)} = \{6, 12\}$. The sets of component indices for the second one, denoted by $\tilde{\mathcal{G}}_2$, are $g^{(1)} = \{1, 7\}$, $g^{(2)} = \{2, 6\}$, $g^{(3)} = \{3, 8\}$, $g^{(4)} = \{4, 10\}$, $g^{(5)} = \{5, 12\}$, and $g^{(6)} = \{9, 11\}$. Fig. 9 depicts the mean optimality gap as a distance from the optimal value versus total number of replications for 50 macro-replications of the stylized problem for $\alpha \in \{0, 0.5, 1\}$ with decompositions $\mathcal{G}_2$, $\hat{\mathcal{G}}_2$, and $\tilde{\mathcal{G}}_2$.

All decompositions work well. For $\alpha = 0$, where the objective function values are fully separable, the natural decomposition $\mathcal{G}_2$ is outperformed by $\tilde{\mathcal{G}}_2$, which is not a natural decomposition. On the other hand, for $\alpha = 1$, where the objective function values are not separable at all, the behavior is the opposite, that is, $\mathcal{G}_2$ outperforms $\tilde{\mathcal{G}}_2$. These results suggest that the natural decomposition is not necessarily the best one, aligned with the results in Section 5.3, but DASSO is robust to the choice.

6. Conclusion

In this paper, we proposed DASSO to tackle high-dimensional DSO problems. DASSO decomposes the prior distribution into an additive form, reducing the problem dimensionality to facilitate efficient posterior updates. This decomposition makes the search much more efficient and computationally possible by avoiding the CEI calculation for each feasible solution: we showed that it is sufficient to calculate the CEIs of only small fraction of solutions at a dice stage. Our numerical results

revealed the effectiveness and efficiencies of DASSO on very large-scale problems: it can obtain rapid improvement on a problem with more than a trillion feasible solutions within a couple of minutes of algorithm overhead. Furthermore, it empirically outperforms the state-of-the-art high-dimensional DSO algorithm pGMIA and BO algorithm Bounce.

Future research includes improving the performance of DASSO via an adaptive stopping condition for the slice stage, inventing a principled way to update the identity of the last group, and exploiting parallel computing: it is easy to parallelize replications for simulating the sample-best and best-CEI solutions and computation of the posterior distributions and Pareto frontiers for each group can also be parallelized. Attaining greater benefit from fast computational linear algebra is also relevant future work. Lastly, although we show that DASSO is capable of tackling a 100-dimensional problem, sampling decisions in the dice stage can be improved from the heuristic we demonstrated here.

CRediT authorship contribution statement

Harun Avci: Writing – original draft, Visualization, Validation, Software, Methodology, Formal analysis, Conceptualization; **Barry L. Nelson:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Conceptualization; **Eunhye Song:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Conceptualization; **Andreas Wächter:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Conceptualization.

Declarations of competing interest

None.

Acknowledgment

This research was partially supported by National Science Foundation Grants CMMI-2206973, CMMI-2045400 and CMMI-2417616.

Supplementary material

Supplementary material associated with this article can be found in the online version at 10.1016/j.ejor.2026.01.005.

References

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

Chen, Y., & Ryzhov, I. O. (2019). Complete expected improvement converges to an optimal budget allocation. *Advances in Applied Probability*, *51*(1), 209–235.

Deshwal, A., Ament, S., Balandat, M., Bakshy, E., Doppa, J. R., & Eriksson, D. (2023). Bayesian optimization over high-dimensional combinatorial spaces via dictionary-based embeddings. In *International conference on artificial intelligence and statistics* (pp. 7021–7039). PMLR.

Djolonga, J., Krause, A., & Cevher, V. (2013). High-dimensional Gaussian process bandits. In *Advances in neural information processing systems* (pp. 1025–1033).

Durrande, N., Ginsbourger, D., & Roustant, O. (2011). Additive kernels for Gaussian process modeling. arXiv preprint arXiv:1103.4023.

Eriksson, D., & Jankowiak, M. (2021). High-dimensional Bayesian optimization with sparse axis-aligned subspaces. In *Uncertainty in artificial intelligence* (pp. 493–503). PMLR.

Frazier, P., Powell, W., & Dayanik, S. (2009). The knowledge-gradient policy for correlated normal beliefs. *INFORMS Journal on Computing*, *21*(4), 599–613.

Fu, M. C. et al. (2015). Handbook of simulation optimization (vol. 216). Springer.

Gardner, J., Guo, C., Weinberger, K., Garnett, R., & Grosse, R. (2017). Discovering and exploiting additive structure for Bayesian optimization. In *Proceedings of the 20th international conference on artificial intelligence and statistics* (pp. 1311–1319). PMLR.

Garnett, R., Osborne, M. A., & Hennig, P. (2014). Active learning of linear embeddings for Gaussian processes. In *Proceedings of the thirtieth conference on uncertainty in artificial intelligence* (pp. 230–239).

Ginsbourger, D., Roustant, O., Schuhmacher, D., Durrande, N., & Lenz, N. (2016). On ANOVA decompositions of kernels and Gaussian random field paths. In *Monte Carlo and Quasi-Monte Carlo methods* (pp. 315–330). Springer.

Hernández-Lobato, J. M., Hoffman, M. W., & Ghahramani, Z. (2014). Predictive entropy search for efficient global optimization of black-box functions. *Advances in Neural Information Processing Systems*, *27*, 918–926.

Hoang, T. N., Hoang, Q. M., Ouyang, R., & Low, K. H. (2018). Decentralized high-dimensional Bayesian optimization with factor graphs. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 3231–3238).

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *Learning and intelligent optimization: 5th international conference, LION 5, Rome, Italy, January 17–21, 2011. selected papers 5* (pp. 507–523). Springer.

Hvarfner, C., Hutter, F., & Nardi, L. (2022). Joint entropy search for maximally-informed Bayesian optimization. *Advances in Neural Information Processing Systems*, *35*, 11494–11506.

Jian, N., Freund, D., Wiberg, H. M., & Henderson, S. G. (2016). Simulation optimization for a large-scale bike-sharing system. In *Proceedings of the winter simulation conference* (pp. 602–613). IEEE.

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, *13*(4), 455–492.

Kandasamy, K., Schneider, J., & Póczos, B. (2015). High dimensional Bayesian optimisation and bandits via additive models. In *International conference on machine learning* (pp. 295–304). PMLR.

Li, X., & Song, E. (2020). Smart linear algebraic operations for efficient Gaussian Markov improvement algorithm. In *Proceedings of the winter simulation conference* (pp. 2887–2898). IEEE.

Li, X., & Song, E. (2024). Projected Gaussian Markov improvement algorithm for high-dimensional discrete optimization via simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, *34*(3), 1–29.

Mathesen, L., Chandrasekar, K. K., Li, X., Pedrielli, G., & Candan, K. S. (2019). Subspace communication driven search for high dimensional optimization. In *Proceedings of the winter simulation conference* (pp. 3528–3539). IEEE.

Mes, M. R. K., Powell, W. B., & Frazier, P. I. (2011). Hierarchical knowledge gradient for sequential sampling. *Journal of Machine Learning Research*, *12*(10), 2931–2974.

Muehlenstaedt, T., Roustant, O., Carraro, L., & Kuhnt, S. (2012). Data-driven kriging models based on FANOVA-decomposition. *Statistics and Computing*, *22*, 723–738.

Oh, C., Tomczak, J., Gavves, E., & Welling, M. (2019). Combinatorial Bayesian optimization using the graph cartesian product. *Advances in Neural Information Processing Systems*, *32*, 2891–2901.

Papenmeier, L., Nardi, L., & Poloczek, M. (2023). Bounce: Reliable high-dimensional Bayesian optimization for combinatorial and mixed spaces. *Advances in Neural Information Processing Systems*, *36*, 1764–1793.

Quan, N., Yin, J., Ng, S. H., & Lee, L. H. (2013). Simulation optimization via kriging: A sequential search using expected improvement with computing budget constraints. *IIE Transactions*, *45*(7), 763–780.

Rolland, P., Scarlett, J., Bogunovic, I., & Cevher, V. (2018). High-dimensional Bayesian optimization via additive models with overlapping groups. In *Proceedings of the twenty-first international conference on artificial intelligence and statistics* (pp. 298–307).

Rue, H., & Held, L. (2005). Gaussian Markov random fields: Theory and applications. Chapman and Hall/CRC.

Salemi, P. L., Song, E., Nelson, B. L., & Staum, J. (2019). Gaussian Markov random fields for discrete optimization via simulation: Framework and algorithms. *Operations Research*, *67*(1), 250–266.

Semelhago, M., Nelson, B. L., Song, E., & Wächter, A. (2021). Rapid discrete optimization via simulation with Gaussian Markov random fields. *INFORMS Journal on Computing*, *33*(3), 915–930.

Semelhago, M., Nelson, B. L., Wächter, A., & Song, E. (2017). Computational methods for optimization via simulation using Gaussian Markov random fields. In *Proceedings of the winter simulation conference* (pp. 2080–2091). IEEE.

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

Tongarlak, M. H., Ankenman, B., Nelson, B. L., Borne, L., & Wolfe, K. (2010). Using simulation early in the design of a fuel injector production line. *Interfaces*, *40*(2), 105–117.

Tripathy, R., Bilionis, I., & Gonzalez, M. (2016). Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation. *Journal of Computational Physics*, *321*, 191–223.

Ulaganathan, S., Couckuyt, I., Dhaene, T., Degroote, J., & Laermans, E. (2016). High dimensional kriging metamodelling utilising gradient information. *Applied Mathematical Modelling*, *40*(9–10), 5256–5270.

Wan, X., Nguyen, V., Ha, H., Ru, B., Lu, C., & Osborne, M. A. (2021). Think global and act local: Bayesian optimisation over high-dimensional categorical and mixed search spaces. In *International conference on machine learning* (pp. 10663–10674). PMLR.

Wang, Z., Gehring, C., Kohli, P., & Jegelka, S. (2018). Batched large-scale Bayesian optimization in high-dimensional spaces. In *Proceedings of the twenty-first international conference on artificial intelligence and statistics* (pp. 745–754). PMLR.

Wang, Z., Hutter, F., Zoghi, M., Matheson, D., & De Feitas, N. (2016). Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, *55*, 361–387.

Wang, Z., & Jegelka, S. (2017). Max-value entropy search for efficient Bayesian optimization. In *International conference on machine learning* (pp. 3627–3635). PMLR.

Wang, Z., Li, C., Jegelka, S., & Kohli, P. (2017). Batched high-dimensional Bayesian optimization via structural kernel learning. In *Proceedings of the 34th international conference on machine learning* (pp. 3656–3664). PMLR.

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

Ziomek, J. K., & Ammar, H. B. (2023). Are random decompositions all we need in high dimensional Bayesian optimisation? In *International conference on machine learning* (pp. 43347–43368). PMLR.