BO4IO: A Bayesian optimization approach to inverse optimization with uncertainty quantification

Yen-An Lu[1], Wei-Shou Hu[1], Joel A. Paulson [*2], and Qi Zhang [†1]

[1]*Department of Chemical Engineering and Materials Science, University of Minnesota, Minneapolis, MN 55455, USA*
[2]*Department of Chemical and Biomolecular Engineering, The Ohio State University, Columbus, OH 43210, USA*

Abstract

Data-driven inverse optimization (IO) aims to estimate unknown parameters in an optimization model from observed decisions. The IO problem is commonly formulated as a large-scale bilevel program that is notoriously difficult to solve. We propose a derivative-free optimization approach based on Bayesian optimization, BO4IO, to solve general IO problems. The main advantages of BO4IO are two-fold: (i) it circumvents the need of complex reformulations or specialized algorithms and can hence enable computational tractability even when the underlying optimization problem is nonconvex or involves discrete variables, and (ii) it allows approximations of the profile likelihood, which provide uncertainty quantification on the IO parameter estimates. Our extensive computational results demonstrate the efficacy and robustness of BO4IO to estimate unknown parameters from small and noisy datasets. In addition, the proposed profile likelihood analysis effectively provides good approximations of the confidence intervals on the parameter estimates and assesses the identifiability of the unknown parameters.

1 Introduction

Inverse optimization (IO) is an emerging method for learning unknown decision-making processes (Chan et al., 2023). Following the principle of optimality (Schoemaker, 1991), the fundamental idea is to use mathematical optimization as a model for decision-making, where decisions can be viewed as the optimal or near-optimal solutions to an underlying optimization problem. Given observed decisions made by an agent, the goal of IO is to learn the unknown optimization model that best represents the agent's decision-making process. A key advantage of the IO approach is that it can directly consider constraints, which allows us to leverage all the modeling flexibility of mathematical programming, incorporate domain knowledge, and hence obtain inherently interpretable decision-making models (Gupta and Zhang, 2023); this makes it distinct

[*]Corresponding author (paulson.82@osu.edu)
[†]Corresponding author (qizh@umn.edu)

from common black-box machine learning methods such as deep learning, where neural networks are trained as optimization proxies to map inputs to decisions (Sun et al., 2018; Karg and Lucia, 2020).

The concept of IO was first introduced in an inverse shortest-path problem, where travel costs in a network are determined based on a given route taken by a user (Burton and Toint, 1992). Since then, IO has been applied to problems across a wide variety of fields, including personalized treatment planning in healthcare (Chan et al., 2014; Ghate, 2020; Ajayi et al., 2022), network design in transportation and power systems (Bertsimas et al., 2015; Zhang et al., 2018; Fernández-Blanco et al., 2021), risk preference learning in portfolio optimization (Li, 2021; Yu et al., 2023), and inference of cellular decision making in biological systems (Burgard and Maranas, 2003; Uygun et al., 2007; Zhao et al., 2016).

Early IO research focused on the development of solution approaches for various classes of deterministic IO problems (IOPs) where it is assumed that observed decisions are exact globally optimal solutions to the unknown forward optimization problem (FOP). Ahuja and Orlin (2001) proposed a general solution method for estimating the cost coefficients of a linear program. Others have proposed extensions that address conic (Iyengar and Kang, 2005; Zhang and Xu, 2010), discrete (Schaefer, 2009; Wang, 2009), and nonlinear convex FOPs (Zhang and Zhang, 2010), as well as constraint estimation (Chan and Kaw, 2020; Ghobadi and Mahmoudzadeh, 2021). In recent years, the focus has shifted toward data-driven IO, where parameters are estimated based on a set of noisy observations collected under different experiment (input) conditions (Mohajerin Esfahani et al., 2018). As such, data-driven IO can address real-world problems where the observed decisions are rarely perfectly optimal given a hypothesized structure of the FOP.

A data-driven IOP is typically formulated as a bilevel program, featuring as many lower-level problems as there are observations. Each lower-level problem represents an instance of the FOP with the corresponding model inputs and observed decisions. The upper-level problem aims to estimate model parameters that align the FOP solutions with the observed decisions. IOPs are notoriously difficult to solve as their size grows with the number of observations. Common solution approaches involve single-level reformulations based on replacing the lower-level problems with their optimality conditions (Keshavarz et al., 2011; Aswani et al., 2018) or cutting plane methods (Wang, 2009). Recent works have also considered tailored decomposition methods for solving IOPs with high-dimensional FOPs and large datasets (Gupta and Zhang, 2022, 2023). In general, existing exact solution methods for IOPs cannot be applied to all classes of IOPs, and designing such algorithms becomes particularly challenging when the FOP is nonconvex.

In this work, we depart from the existing literature and take a derivative-free optimization (DFO) approach to solve general IOPs, where we treat the evaluation of the loss function as a black box. DFO is commonly applied to optimization problems where the full derivative information is not easily accessible (Rios and Sahinidis, 2013). In the field of bilevel or multilevel optimization, local or global sampled-based algorithms, e.g. evolutionary algorithms, have long been used as metaheuristic solution methods (Talbi, 2013; Sinha et al., 2017). In the DFO framework developed by Beykal et al. (2020), a bilevel program is solved as a single-level grey-box optimization problem, where at each sampling point, the lower-level problem is solved to generate input-output information that the grey-box solver can leverage to suggest new solution candidates and iteratively optimize the upper-level objective. In this work, we propose to use Bayesian

optimization (BO), a model-based DFO method designed to find the optimum of an expensive-to-evaluate objective function within a relatively small number of iterations (Frazier, 2018). BO leverages Bayesian statistics and surrogate modeling to update a prior distribution for the objective function as new function evaluations are performed, and it provides a means of balancing exploration and exploitation in sampling the next points. BO has become very popular in recent years and has proven to be an effective method for, for example, hyperparameter tuning in deep learning (Snoek et al., 2012; Shahriari et al., 2015) and design of experiments (Greenhill et al., 2020; Frazier and Wang, 2016). However, only recently, it has also been used to tackle bilevel programs (Kieffer et al., 2017; Dogan and Prestwich, 2023) as well as multilevel optimization problems that arise in feasibility analysis and robust optimization (Kudva et al., 2022, 2024).

In our proposed BO framework for solving general IOPs, which we call BO4IO, we approximate the loss function of the IOP with a non-parametric probabilistic surrogate using Gaussian process (GP) regression (Williams and Rasmussen, 2006). It converges to the optimal parameter estimates that provide the minimum decision loss by iteratively selecting candidate solutions through the minimization of the acquisition function. Here, the key advantage of BO4IO is that, at each iteration, it only requires the evaluation of the loss function with the current parameter estimates; this can be achieved by directly solving the FOP for every data point, which circumvents the need for a complex reformulation or special algorithm. Consequently, BO4IO remains applicable even when the FOP is nonlinear and nonconvex, and it can consider both continuous and discrete decision variables. This approach also naturally decomposes the problem in a sense that the FOP can be solved independently for each data point, which allows parallelization and hence enhances the scalability of the algorithm with respect to the number of data points.

BO4IO also enables us to address another major challenge in IO that is often overlooked, namely the quantification of uncertainty with respect to the estimated parameters. In IO, there are often multiple parameter estimates that lead to the same loss; these "equally good" estimates form the so-called inverse-feasible set. The size of the inverse-feasible set can be viewed as a measure of uncertainty in any point estimate drawn from that set; the larger the inverse-feasible set, the less confident one can be about the estimate. In the case of linear programs where the cost vector is unknown, the inverse-feasible set takes the form of a polyhedral cone; Gupta and Zhang (2022) use this insight to develop an adaptive sampling method that tries to sample points that can help further reduce the size of the inverse-feasible set. However, that method is specific to linear programs and cannot be directly extended to general IOPs. A common approach in parameter estimation is to derive confidence intervals for the point estimates. Asymptotic confidence intervals can be computed using a Fisher information matrix (Sobel, 1982), but it is only exact when the estimated model is linear (Joshi et al., 2006). In contrast, sample-based confidence intervals derived using the profile likelihood function also apply to the nonlinear case (Neale and Miller, 1997; Raue et al., 2009). We find that in BO4IO, we can use the posterior of the GP surrogate to approximate the profile likelihood, which provides uncertainty quantification and insights into the identifiability of given model parameters.

To assess the efficacy of the proposed BO approach for various classes of FOPs, we perform computational case studies wherein we learn cellular objectives in flux balance analysis of metabolic networks as well as market requirements in standard and generalized pooling problems. The computational experiments demonstrate that BO4IO can efficiently identify estimates

of the unknown model parameters with a relatively small number of iterations. It proves to be especially beneficial in cases where the size of the FOP is large relative to the number of unknown parameters. In addition, using the approximate profile likelihood, we can assess how the identifiability of certain parameters changes with the number and quality of observations. To the best of our knowledge, this is the first DFO method specifically designed to solve IOPs as well as the first approach that provides a direct measure of uncertainty on the IO parameter estimates.

The remainder of this paper is structured as follows. In Section 2, we present a general formulation of an IOP. We provide the necessary background on GPs and BO and introduce our BO4IO algorithm in Section 3. In Section 4, we introduce the profile likelihood and derive its approximation based on the GP surrogate in BO4IO. Finally, we demonstrate the performance of BO4IO in three numerical case studies in Section 5 and conclude in Section 6.

2 The inverse optimization problem

In IO, we assume that a decision-making process can be modeled as an optimization problem, also referred to as the *forward* optimization problem, formulated in the following general form:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x, u; \theta) \\ \text{subject to} \quad & g(x, u; \theta) \leq 0, \end{aligned} \tag{FOP}$$

where $x \in \mathbb{R}^n$ is the vector of decision variables, $u \in \mathbb{R}^m$ denotes the vector of contextual inputs that describe the system conditions, and f and g represent the objective and constraint functions, respectively, that are parameterized by model parameters $\theta \in \Theta$, where Θ is some compact set in $\mathbb{R}^d$. In a data-driven IO setting, θ are unknown, but decisions $\{x_i\}_{i \in \mathcal{I}}$ can be collected under varying conditions $\{u_i\}_{i \in \mathcal{I}}$, where $\mathcal{I}$ denotes the set of observations. The observations are assumed to be noisy, which could be due to, for example, measurement errors, suboptimal decisions, or model mismatch. The goal of IO is to estimate θ such that the solutions of (FOP) best fit the observed decisions, which gives rise to the following bilevel optimization problem:

$$\begin{aligned} \underset{\hat{\theta} \in \Theta,\, \hat{x}}{\text{minimize}} \quad & l(\hat{\theta}) := \sum_{i \in \mathcal{I}} (x_i - \hat{x}_i(\hat{\theta}))^\top W (x_i - \hat{x}_i(\hat{\theta})) \\ \text{subject to} \quad & \hat{x}_i(\hat{\theta}) \in \underset{\tilde{x} \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ f(\tilde{x}, u_i; \hat{\theta}) : g(\tilde{x}, u_i; \hat{\theta}) \leq 0 \right\} \quad \forall\, i \in \mathcal{I}, \end{aligned} \tag{IOP}$$

where the objective of the upper-level problem is to obtain parameter estimates, denoted by $\hat{\theta}$, such that the (empirical) decision loss $l : \Theta \to \mathbb{R}_+$, a function that quantifies the agreement between the model predictions ($\hat{x}_i$) and observations (x_i), is minimized. For simplicity, we focus on the case where l is defined as a weighted sum of squared residuals, where $W \in \mathbb{S}^n_+$ denotes the positive definite matrix of weighting factors; this loss function can be easily replaced with a more general likelihood or posterior loss function (see, e.g., (Goodfellow et al., 2016, Chapter 5)). In the $|\mathcal{I}|$ lower-level problems of (IOP), one for each observation, $\hat{x}_i$ is constrained to be an optimal solution to the corresponding optimization problem parameterized by $\hat{\theta}$; as such, $\hat{x}_i$ is a function of $\hat{\theta}$.

3 The BO4IO framework

In this section, we provide a brief overview of derivative-free optimization and Gaussian process regression, and show how they are applied to solve (IOP) in the proposed Bayesian optimization for inverse optimization (BO4IO) framework.

3.1 Derivative-free inverse optimization

We propose to solve general IOPs using a DFO approach where we treat (IOP) as the following black-box optimization problem:

$$\underset{\hat{\theta}\in\Theta}{\text{minimize}} \quad l(\hat{\theta}).$$

DFO methods aim to sequentially optimize the target objective within a designated search space, where the query points are suggested based on past function evaluations. At each iteration, the objective function, which in our case is the loss function l, is evaluated for a given $\hat{\theta}$ by solving the $|\mathcal{I}|$ individual FOPs to global optimality. Specifically, for every $i \in \mathcal{I}$, we solve (FOP) with $u = u_i$ and $\theta = \hat{\theta}$ to obtain $\hat{x}_i(\hat{\theta})$ with which we can then compute $l(\hat{\theta}) = \sum_{i\in\mathcal{I}}(x_i - \hat{x}_i(\hat{\theta}))^\top W(x_i - \hat{x}_i(\hat{\theta}))$. Note that each FOP can be solved independently; hence, they can be solved in parallel to further reduce the computation time.

Even though parallel computing can be applied, each evaluation of the loss function can still be expensive if the underlying FOP is difficult to solve; hence, it is important to select a DFO algorithm that requires as few evaluations as possible. DFO methods can be roughly categorized into direct and model-based methods (Rios and Sahinidis, 2013). While direct algorithms explicitly determine a search direction from previous function evaluations, model-based methods use these function evaluations to construct a surrogate model that approximates the target function and determine the next sampling point using that surrogate (Jones et al., 1998; Huyer and Neumaier, 2008). In this work, we choose Bayesian optimization (BO) to be our preferred model-based DFO method (Mockus, 1994; Frazier, 2018). In BO, a probabilistic surrogate model is constructed where Bayes' rule is used to compute the surrogate posterior distribution based on the selected prior and past function evaluations. BO uses the posterior to quantify the uncertainty in the estimate and forms a utility (acquisition) function that balances the exploration-exploitation trade-off in the search process. A wide variety of probabilistic surrogate models have been used in BO, e.g., Gaussian processes (Mockus, 1994), random forests (Hutter et al., 2011), tree-structured Parzen estimators (Bergstra et al., 2011), and deep neural networks (Snoek et al., 2015). In this work, we use GP as the probabilistic surrogate due to its nonparametric feature and the analytical expression of the posterior; however, it should be noted that the proposed framework can readily consider other surrogate models.

3.2 Gaussian process regression

Based on the standard GP detailed in Williams and Rasmussen (2006), we build a GP model that approximates the loss function $l : \Theta \to \mathbb{R}_+$. The approximation is obtained by assuming that the function values $l(\hat{\theta})$ at different $\hat{\theta}$ values are random variables of which any finite subset forms a joint Gaussian distribution. A GP model can thus be interpreted as an infinite collection of these

random variables, describing a distribution over the function space.

We let $\hat{l}(\hat{\theta}) \sim \mathcal{GP}(\mu(\hat{\theta}), \kappa(\hat{\theta}, \hat{\theta}'))$ denote the GP surrogate of the true loss function l. The GP model is fully specified by a prior mean function $\mu(\hat{\theta})$ and a covariance (kernel) function $\kappa(\hat{\theta}, \hat{\theta}')$ that is the covariance of the function values $\hat{l}(\hat{\theta})$ and $\hat{l}(\hat{\theta}')$ for any $\hat{\theta}$, $\hat{\theta}' \in \Theta$. Without loss of generality, we assume that $\mu(\hat{\theta}) = 0$, which can be achieved by normalizing the output data when fitting the surrogate. The kernel function encodes the smoothness and rate of change in the target function (see, e.g., Williams and Rasmussen (2006, Chapter 4)). We focus on the stationary covariance function from the Matern class, which is a popular kernel choice in many GP applications. Let $\mathcal{D}_t = \{(\hat{\theta}_i, l(\hat{\theta}_i)\}_{i=1}^t$ denote a set of t past evaluations. Conditioned on the evaluations in $\mathcal{D}_t$, the predicted posterior distribution of the function $\hat{l}$ at a future input $\hat{\theta}_{t+1}$ remains Gaussian with the following posterior mean $\mu_t(\hat{\theta}_{t+1})$ and covariance $\sigma_t^2(\hat{\theta}_{t+1})$:

$$\mu_t(\hat{\theta}_{t+1}) = \boldsymbol{\kappa}_t^\top(\hat{\theta}_{t+1})\boldsymbol{K}_t^{-1}\boldsymbol{l}_t$$
$$\sigma_t^2(\hat{\theta}_{t+1}) = \kappa(\hat{\theta}_{t+1}, \hat{\theta}_{t+1}) - \boldsymbol{\kappa}_t^\top(\hat{\theta}_{t+1})\boldsymbol{K}_t^{-1}\boldsymbol{\kappa}_t(\hat{\theta}_{t+1}),$$

where $\boldsymbol{l}_t = [l(\hat{\theta}_1), ..., l(\hat{\theta}_t)]^\top$ is the vector of t past loss function evaluations, $\boldsymbol{K}_t \in \mathbb{R}^{t\times t}$ is the covariance matrix with entries $[\boldsymbol{K}_t]_{i,j} = \kappa(\hat{\theta}_i, \hat{\theta}_j)$ for all $i, j \in \{1, ..., t\}$, and $\boldsymbol{\kappa}(\hat{\theta}_{t+1})$ is the vector of covariance values between the new input $\hat{\theta}_{t+1}$ and the evaluated inputs $\{\hat{\theta}_i\}_{i=1}^t$. In practice, the kernel function contains unknown hyperparameters, such as smoothness and length-scale parameters, which are calibrated to the dataset $\mathcal{D}_t$ using maximum likelihood estimation.

3.3 The BO4IO algorithm

A pseudocode for the proposed BO4IO algorithm is shown in Algorithm 1, which aims to minimize the decision loss $l(\hat{\theta})$ of (IOP) over the parameter domain $\hat{\theta} \in \Theta$ by sequentially querying $\hat{\theta}$ based on the predicted posterior distribution of the GP surrogate. At the tth iteration, the GP model is updated using the past t evaluations ($\mathcal{D}_t$). An acquisition function that encodes the exploration-exploitation trade-off is constructed based on the predicted GP posterior and is optimized to query the next sampling point $\hat{\theta}_{t+1}$. The true loss function value $l(\hat{\theta}_{t+1})$ is then evaluated by solving $|\mathcal{I}|$ FOPs. The dataset $\mathcal{D}_t$ is then appended with the new evaluation $(\hat{\theta}_{t+1}, l(\hat{\theta}_{t+1}))$ to create a concatenated dataset $\mathcal{D}_{t+1}$. The new dataset $\mathcal{D}_{t+1}$ is then used to update the GP model, and the same process is repeated until the maximum number of iterations is reached. Note that other termination criteria, such as an acceptable level for the decision loss or a maximum number of consecutive iterations without improvement in the loss value, can also be considered if needed.

A wide variety of acquisition functions have been developed in BO; here, we use the lower confidence bound (LCB) acquisition function (Auer, 2002) due to its simplicity and nice theoretical properties. The acquisition function is minimized to obtain the next sampling point, i.e.

$$\hat{\theta}_{t+1} \in \underset{\hat{\theta}\in\Theta}{\operatorname{argmin}}\ \mu_t(\hat{\theta}) - \beta_t^{1/2}\sigma_t(\hat{\theta}), \tag{1}$$

where $\beta_t > 0$ is an exploration factor. When β_t is defined as an iteration-dependent parameter, some theoretical guarantees can be derived for the convergence to a global optimal solution (Srinivas et al., 2009); however, the resulting search process may become very slow with the adaptive approach. Instead, one often considers a constant β_t value that bounds the failure probability per

iteration. For example, a value of $\beta_t = 4$ has demonstrated good results in practical applications (Berkenkamp et al., 2019).

Algorithm 1: BO4IO: Bayesian optimization for inverse optimization

Input: parameter domain Θ;
kernel for GP prior $\kappa(\hat{\theta}, \hat{\theta}')$;
exploration parameter β;
set of observations $\{(u_i, x_i)\}_{i=1}^{|\mathcal{I}|}$;
FOPs with inputs $\{u_i\}_{i=1}^{|\mathcal{I}|}$;
initial evaluation dataset $\mathcal{D}_0$;
maximum number of iterations T.

Generate GP model using $\mathcal{D}_0$
for $t = 1, 2, ..., T$ **do**
  $\hat{\theta}_{t+1} \in \operatorname{argmin}_{\hat{\theta} \in \Theta} \mu_t(\hat{\theta}) - \beta \sigma_t(\hat{\theta})$, see (1)
  Solve the $|\mathcal{I}|$ FOPs with $\hat{\theta}_{t+1}$ and compute $l(\hat{\theta}_{t+1})$
  $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(\hat{\theta}_{t+1}, l(\hat{\theta}_{t+1}))\}$
  Update GP model using $\mathcal{D}_{t+1}$
end for
return optimal solution $\hat{\theta}_T^* \in \operatorname{argmin}_{\hat{\theta}_t \in \mathcal{D}_T} \{l(\hat{\theta}_t)\}_{t=1}^T$

4 Uncertainty quantification using profile likelihood

In this section, we propose a method to quantify the uncertainty in the parameter estimates from BO4IO using the profile likelihood (PL) method. We first review uncertainty quantification in general parameter estimation problems and discuss how the confidence interval (CI) of each parameter estimate can be derived using the PL method. We then propose approximations of the PLs and CIs based on the GP surrogate in BO4IO, which can be used to derive optimistic and worst-case CIs of parameter estimates and provide insights into the identifiability of the unknown parameters in IO.

4.1 General approach for parameter estimation and uncertainty quantification

It is widely known that the weighted sum of the residuals function $l(\theta)$ defined in (IOP) is directly related to the likelihood function under the assumption of independent and identically distributed (i.i.d.) Gaussian observation noise. In particular, if $x_i = \hat{x}_i(\hat{\theta}) + \epsilon_i$ with $\epsilon_i \sim \mathcal{N}(0, \sigma_i^2)$, then

$$l(\hat{\theta}) = c - 2\log(L(\hat{\theta})),$$

where $L(\hat{\theta}) = p(\{x_i\}_{i \in \mathcal{I}} | \{u_i\}_{i \in \mathcal{I}}, \hat{\theta})$ is the likelihood function, c is a constant, and $l(\hat{\theta})$ is given in (IOP) with $W = \operatorname{diag}(\sigma_1, \ldots, \sigma_{|\mathcal{I}|})$. Let $\hat{\theta}^* \in \operatorname{argmin}_{\hat{\theta} \in \Theta} l(\hat{\theta})$ denote the maximum likelihood estimate (MLE) of the unknown parameters θ. Finite sample confidence intervals, which signify

that the true value is within a set with at least some probability $1-\alpha$, can be derived as follows

$$\mathrm{CI}_{\hat{\theta}^*} = \{\hat{\theta} \in \Theta : l(\hat{\theta}) - l(\hat{\theta}^*) < \Delta_\alpha\}, \tag{2}$$

where Δ_α denotes the α-level quantile of the standard χ^2 distribution with degrees of freedom equal to 1 for pointwise and d for simultaneous CIs, respectively.

An important challenge in many parameter estimation problems is determining the *identifiability* of the parameters to be estimated in the model (Bellman and Åström, 1970; Cobelli and Distefano III, 1980). We can think of identifiability in terms of the size of the CI. If the CI is small, we have found a narrow range in which the true parameter exists and so we can say that the model is fully identifiable (and we can consequently trust the found estimates $\hat{\theta}^*$). On the other hand, if the CI is extended in any direction, we are unable to pinpoint an accurate estimate of the parameter; hence, the parameter is non-identifiable. As discussed in Wieland et al. (2021), there are two types of non-identifiabilities that can emerge in a model. The first is *structural non-identifiability*, which is the result of an effectively redundant model parametrization. Structurally non-identifiable parameters result in non-unique MLE estimates $\hat{\theta}^*$ such that they cannot be resolved by gathering more data or improving the quality of the existing data. The second is *practical non-identifiability*, which refers to a parameter that is structurally identifiable (in the sense that it has a unique MLE estimate) but $l(\hat{\theta}) - l(\hat{\theta}^*) < \Delta_\alpha$ remains below the threshold Δ_α for a large portion of Θ. One may be able to make practically non-identifiable parameters identifiable by collecting additional and/or higher-quality data to reduce the CI. One can use model-based design of experiments (DOE) methods (Balsa-Canto et al., 2008; Bandara et al., 2009; Kreutz and Timmer, 2009) to systematically select new experimental trials that are most informative for reducing the CI. We plan to study the DOE problem for (IOP) in more detail in future work.

4.2 The profile likelihood method

The PL method is an effective strategy for differentiating between the different identifiability categories, even in high-dimensional spaces (Raue et al., 2009). The approach assesses the identifiability of a parameter of interest $\hat{\theta}_k$ by profiling the maximum likelihood (i.e. minimum loss). The profile likelihood $\mathrm{PL}(\hat{\theta}_k)$ for a given value of $\hat{\theta}_k$ is defined as

$$\mathrm{PL}(\hat{\theta}_k) = \min_{\hat{\theta}_{k' \neq k}} \; l(\hat{\theta}). \tag{3}$$

Note that $\mathrm{PL}(\hat{\theta}_k)$ is obtained by fixing $\hat{\theta}_k$ while re-optimizing the rest of the parameters. Based on the definition given in (2), the CI of each parameter estimate θ_k^* can be computed as follows:

$$\mathrm{CI}_{\hat{\theta}_k^*} = \{\hat{\theta}_k \mid \mathrm{PL}(\hat{\theta}_k) - l(\hat{\theta}^*) \leq \Delta_\alpha\}, \tag{4}$$

where $l(\hat{\theta}^*)$ is the decision loss with the optimal parameter estimates of (IOP). It should be noted that the re-optimization during the PL evaluation provides useful information about the nonlinear relationships between the parameters, which can be further leveraged in model reduction (Maiwald et al., 2016).

4.3 Approximation of profile likelihood and confidence intervals in BO4IO

Similar to (IOP), solving (3) exactly is challenging for general FOPs; as a result, we do not have access to the exact PL values and CIs of the parameter estimates. Instead, we approximate them using the GP posterior obtained in BO4IO. We first define the upper and lower confidence bounds on the true loss function $l(\hat{\theta})$ at the tth iteration as follows:

$$\hat{l}_t^{\text{UCB}}(\hat{\theta}) = \mu_t(\hat{\theta}) + \rho_t^{1/2}\sigma_t(\hat{\theta})$$
$$\hat{l}_t^{\text{LCB}}(\hat{\theta}) = \mu_t(\hat{\theta}) - \rho_t^{1/2}\sigma_t(\hat{\theta}),$$

where ρ_t is a parameter denoting the confidence levels. The confidence bounds on the PL of a parameter of interest $\hat{\theta}_k$ at the tth iteration can be further derived as

$$\widehat{\text{PL}}_t^{\text{UCB}}(\hat{\theta}_k) = \min_{\hat{\theta}_{k'\neq k}} \ \hat{l}_t^{\text{UCB}}(\hat{\theta}) \tag{5}$$

$$\widehat{\text{PL}}_t^{\text{LCB}}(\hat{\theta}_k) = \min_{\hat{\theta}_{k'\neq k}} \ \hat{l}_t^{\text{LCB}}(\hat{\theta}), \tag{6}$$

where $\widehat{\text{PL}}_t^{\text{UCB}}(\hat{\theta}_k)$ and $\widehat{\text{PL}}_t^{\text{LCB}}(\hat{\theta}_k)$ denote the upper and lower confidence bounds on the PL, respectively. In addition, the optimistic estimate of the minimum decision loss can be determined by $\hat{l}_t^{*\text{LCB}} = \min l_t^{\text{LCB}}(\hat{\theta})$, whereas a pessimistic estimate of the minimum decision loss is the current best-found loss $l_t^* = \min\{l(\hat{\theta}_i), ..., l(\hat{\theta}_t)\}$. The outer-approximation (OA) and inner-approximation (IA) of the CI on each best parameter estimate $\hat{\theta}_k^*$ at the tth iteration can hence be written as

$$\text{OA-CI}_{t,\hat{\theta}_k^*} = \{\hat{\theta}_k | \widehat{\text{PL}}_t^{\text{LCB}}(\hat{\theta}_k) - l_t^* \leq \Delta_\alpha\} \tag{7}$$

$$\text{IA-CI}_{t,\hat{\theta}_k^*} = \{\hat{\theta}_k | \widehat{\text{PL}}_t^{\text{UCB}}(\hat{\theta}_k) - \hat{l}_t^{*\text{LCB}} \leq \Delta_\alpha\}. \tag{8}$$

Since $l_t^* \geq l^* \geq \hat{l}_t^{*\text{LCB}}$ and $\widehat{\text{PL}}_t^{\text{UCB}}(\hat{\theta}_k) \geq \text{PL}(\hat{\theta}_k) \geq \widehat{\text{PL}}_t^{\text{LCB}}(\hat{\theta}_k)$, we have $\text{OA-CI}_{t,\hat{\theta}_k^*} \subseteq \text{CI}_{\hat{\theta}_k^*} \subseteq \text{IA-CI}_{t,\hat{\theta}_k^*}$. In other words, $\text{OA-CI}_{t,\hat{\theta}_k^*}$ provide a worst-case estimate of the confidence level, whereas $\text{IA-CI}_{t,\hat{\theta}_k^*}$ is an optimistic estimate. An illustrative example of the approximate and true CIs are shown in Figure 1. Notably, we are primarily interested in the worst-case estimate of the CI; hence, we only report $\text{OA-CI}_{t,\hat{\theta}_k^*}$ in our case studies. The pseudocode for the algorithm used to determine $\text{OA-CI}_{t,\hat{\theta}_k^*}$ is shown in Algorithm 2.

5 Computational case studies

We apply the proposed BO4IO algorithm to three computational case studies covering various classes of FOPs. In the first case study, we estimate the optimal linear combinations of multiple cellular objectives in the flux balance analysis, which is formulated as a convex nonlinear program (NLP). In the second case study, we learn the product demands in the standard pooling problem, which is formulated as a nonconvex NLP. Finally, we extend the second study to a generalized pooling problem that contains mixed-integer decisions and estimate the product quality constraints. Using synthetic data, we evaluate the performance of the algorithm and apply the

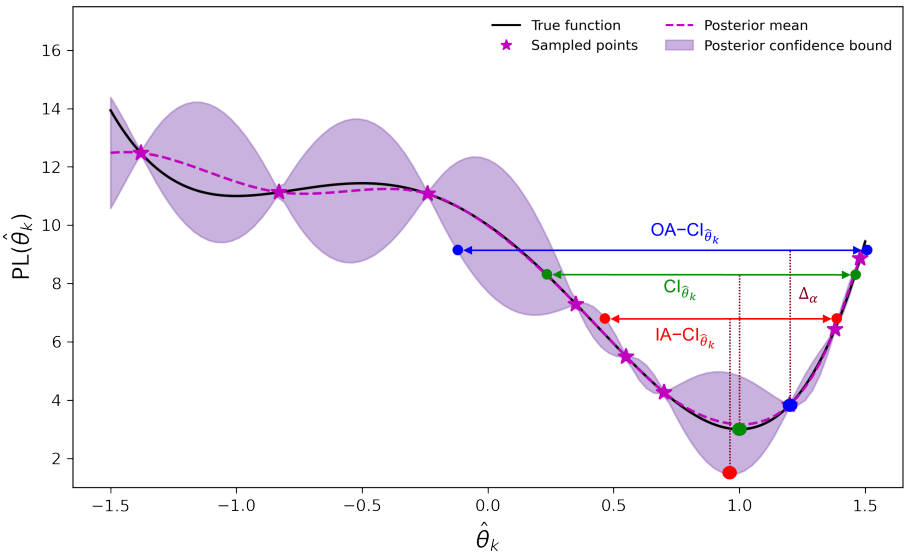

Figure 1: Illustrative example of outer-approximate (OA-CI, blue), original (CI, green), and inner-approximate (IA-CI, red) confidence intervals of a parameter of interest ($\hat{\theta}_k$) based on the proposed PL analysis.

Algorithm 2: Outer-approximation of confidence intervals in BO4IO

Input: parameter of interest, $\hat{\theta}_k$;
posterior mean and covariance function of GP $\mu_t(\hat{\theta})$ and $\sigma_t^2(\hat{\theta})$;
confidence parameter ρ;
the α-quantile of χ^2 distribution Δ_α;
the best decision loss of t past evaluations l_t^*;
the step size for $\hat{\theta}_k$ $\delta_{\hat{\theta}_k}$;
the sampled PL range $[\hat{\theta}_k^{\mathrm{LB}}, \hat{\theta}_k^{\mathrm{UB}}]$

$\mathcal{S} \leftarrow \emptyset$ and $\hat{\theta}_k \leftarrow \hat{\theta}_k^{\mathrm{LB}}$
while $\hat{\theta}_k \leq \hat{\theta}_k^{\mathrm{UB}}$ **do**
  $\widehat{\mathrm{PL}}_t^{\mathrm{LCB}}(\hat{\theta}_k) \leftarrow \min_{\hat{\theta}_{k' \neq k}} \mu_t(\hat{\theta}) - \rho\sigma_t(\hat{\theta})$, see (6)
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{(\hat{\theta}_k, \widehat{\mathrm{PL}}_t^{\mathrm{LCB}}(\hat{\theta}_k))\}$
  $\hat{\theta}_k = \hat{\theta}_k + \delta_{\hat{\theta}_k}$
end while
OA-CI$_{t,\hat{\theta}_k^*} \leftarrow \{\hat{\theta}_k | \widehat{\mathrm{PL}}_t^{\mathrm{LCB}}(\hat{\theta}_k) - \hat{l}_t^* \leq \Delta_\alpha\}$ based on $\mathcal{S}$
return OA-CI$_{t,\hat{\theta}_k^*}$

approximate PL method to assess the confidence intervals and identifiability of the IO parameter estimates. For each specific case, 10 random instances were considered to provide reliable computational statistics. The algorithm was implemented in Python 3.8 using Pyomo (Bynum et al., 2021) as the optimization modeling environment and Gurobi v10.0.1 (Gurobi Optimization, LLC, 2023) as the optimization solver. The BO framework was built under BoTorch (Balandat et al., 2020) where GPyTorch (Gardner et al., 2018) was used to construct the GP models. All codes were run on the Agate cluster of the Minnesota Supercomputing Institute (MSI) using a node allocated with 32 GB of RAM and 64 cores. The implementation of all case studies can be found in our GitHub repository at https://github.com/ddolab/BO4IO.

5.1 Case study 1: Learning cellular objectives in flux balance analysis

Flux balance analysis (FBA) is a method for simulating the flux distributions in the metabolic network of a cell at steady state (Orth et al., 2010b). Metabolic flux is one of the critical determinants of cellular status under different physiological and environmental conditions; the FBA method serves as a powerful tool for the optimization of biosynthesis in microbial industries (Pardelha et al., 2012; Choon et al., 2014) and the identification of drug targets in human diseases (Lee et al., 2009; Nilsson and Nielsen, 2017). The original FBA problem is formulated as a linear program by assuming that the cells try to optimize a cellular objective subject to the reaction stoichiometry of cell metabolism. An extension with l_1 or l_2 regularization is often used to avoid degeneracy in the FBA solutions (Bonarius et al., 1996; Lewis et al., 2010). Moreover, a variation that considers multi-objective optimization via a weighted combination of common objectives has proven to better describe the fluxes in some organisms (Lee et al., 2004; Schuetz et al., 2012; García Sánchez et al., 2012); however, the weights are often arbitrarily assigned (Nagrath et al., 2010). The goal of this IOP is to learn the weight vector in multi-objective FBA from the synthetic datasets. The FBA problem is formulated as the following quadratic FOP with l_2-regularization:

$$\underset{\boldsymbol{v}}{\text{minimize}} \quad \sum_{k\in\mathcal{R}^{\text{obj}}} \theta_k v_k + \lambda \sum_{k\in\mathcal{R}} v_k^2 \tag{9a}$$

$$\text{subject to} \quad \sum_{k\in\mathcal{R}} s_{jk} v_k = 0 \quad \forall j \in \mathcal{M} \tag{9b}$$

$$L_k \le v_k \le U_k \quad \forall k \in \mathcal{R}, \tag{9c}$$

where $\mathcal{M}$ and $\mathcal{R}$ denote the sets of metabolites and metabolic reactions, respectively. Here, the reaction fluxes are represented by v. The first term of the objective function (9a) is a linear combination of common cellular objectives in the set $\mathcal{R}^{\text{obj}}$ with unknown weight parameters θ. The second term of (9a) is the l_2-regularization scaled by a factor λ. Constraints (9b) represent the material balances for all metabolites, where s_{jk} denotes the stoichiometric coefficient of metabolite j in reaction k. Constraints (9c) set lower and upper bounds, respectively denoted by L and U, on the metabolic fluxes.

We consider the core *E. coli* model (ID: e_coli_core) from the BiGG database (Orth et al., 2010a; King et al., 2016), where the metabolic network specification ($|\mathcal{M}|$, $|\mathcal{R}|$) is $(72, 95)$. The true cellular objective is assumed to be an unknown linear combination of up to five common cellular objectives in $\mathcal{R}^{\text{obj}}$, including maximization of biomass formation, maximization of ATP genera-

tion, minimization of CO_2, minimization of nutrient uptake, and minimization of redox potential (Savinell and Palsson, 1992; Schuetz et al., 2007; García Sánchez and Torres Sáez, 2014) with an unknown weight vector θ. In each random instance of the IOP, we first generate the ground-truth parameters θ from a Dirichlet distribution, where the weight vector is scaled by assuming the summation of all weights to be one. The same equality constraint is considered when minimizing the acquisition function; hence, the number of unknown parameters is $d = |\mathcal{R}^{\text{obj}}| - 1$. Next, using the same θ vector, we collect the optimal flux vectors v_i^* for each observation i by solving the FOP under varying input conditions $\{u_i\}_{i\in\mathcal{I}} = \{(L_i, U_i)\}_{i\in\mathcal{I}}$, where the flux bound vectors L_i, $U_i \in \mathbb{R}^{|\mathcal{R}|}$ are sampled from $L_{ik}, U_{ik} \sim \mathcal{U}(10, 100)$ for each $k \in \mathcal{R}$. We then generate the noisy data by first standardizing each optimal flux v_{ik}^* across all observations and perturb the standardized values such that $\bar{v}_{ik} = \bar{v}_{ik}^* + \epsilon_{ik}$ for all $i \in \mathcal{I}$ and $k \in \mathcal{R}$, where $\bar{v}_{ik}^*$ denotes the standardized optimal flux for reaction k and $\epsilon_{ik} \sim \mathcal{N}(0, \sigma^2)$.

5.1.1 Computational performance under varying dimensionality

We run 10 random instances with 50 training data points, i.e. $|\mathcal{I}| = 50$, under varying dimensionality of unknown parameters ($d = 1 \sim 4$) at a low noise level of $\sigma = 0.01$. Separate testing datasets with the same $|\mathcal{I}|$ number of data points are generated to test the model prediction. The convergence profiles for the training and testing decision errors as well as the parameter error over 250 BO iterations are compared under a noise level of $\sigma = 0.1$, as shown in Figure 2. One can see that while the speed of convergence decreases as the dimensionality increases, the algorithm effectively estimates accurate FOPs as all conditions achieve low training errors within 100 BO iterations. The estimated FOPs generalize well to the testing datasets as the training and testing errors share nearly identical convergence rates and reach the same level after 100 iterations. The ground-truth parameters are recovered in all random instances under different dimensionality, where low parameter errors ($\sim 10^{-4}$) are achieved after 200 iterations.

We also record the computation times for solving the 2- and 4-dimensional problems with and without solving the FOPs with 10 parallel workers in a machine on MSI. Table 1 reports the total times as well as the breakdown into the BO and FOPs parts. The BO part includes the time for updating/fitting the GP surrogate and minimizing the acquisition function, whereas the FOPs part accounts for the time required to solve the individual FOPs and compute the loss function value. We observe little differences in the total computation times when the dimension of the problem increases from 2 to 4. Reductions in the total computation time of 46% and 33% are achieved in the 2- and 4-dimension problems, respectively, with FOPs solved in parallel. The breakdown times indicate that more computational resources are used to solve the FOPs, and the speed-ups are mainly attributed to the reduced time from the parallel implementation as we see 47∼56% time reduction in solving the FOPs. Notably, the computation time of the BO part increases around 10∼15% when the dimension of unknown parameters is doubled.

5.1.2 Uncertainty quantification of parameter estimates using profile likelihood

We next apply the proposed PL analysis to evaluate the uncertainty and identifiability of the parameter estimates of BO4IO in the two-dimensional case. All parameters of the 10 random instances show finite OA-CIs, indicating that they are all structurally identifiable. For simplicity, we choose a random instance from the two-dimensional case as an example. The full-space PL

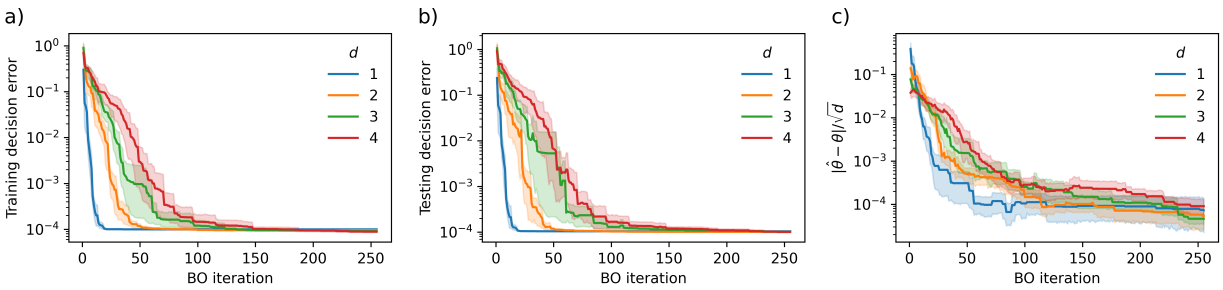

Figure 2: Effect of the dimensionality (d) of θ on the accuracy of the estimated FOPs. Convergence analysis of (a) training error, (b) testing error, and (c) parameter error with varying d. Training and testing errors refer to the average standardized decision error defined as $\sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{R}}(\bar{v}_{ik} - \hat{v}_{ik})^2/|\mathcal{I}|/|\mathcal{R}|$ and calculated based on the training and testing datasets, respectively, whereas the parameter error denotes the difference between the ground-truth (θ) and the estimated ($\hat{\theta}^*$) values. Here, the solid lines and shaded areas respectively denote the medians and confidence intervals of the corresponding error across the 10 random instances. The synthetic dataset of each random instance is generated under the setting of $|\mathcal{I}| = 50$ and $\sigma = 0.01$.

n_θ	Parallelism	Median time (s)			Reduction (%)	
		Total	BO	FOPs	Total	FOPs
2	No	3703.0	731.6	2953.7	45.9	55.6
	Yes	2005.0	698.4	1312.1		
4	No	3149.4	812.9	2369.5	33.2	46.7
	Yes	2104.8	825.6	1262.2		

Table 1: Computational time of BO4IO as FOPs solved with and without parallel computing. Results based on 10 random instances with $|\mathcal{I}| = 50$, $\sigma = 0.1$, and 250 BO iterations. If parallelism is implemented, 50 FOPs are sequentially solved by 10 parallel workers to global optimality. The BO time includes initializing/updating the GP surrogate and minimizing the acquisition function, whereas the FOPs time denotes the time for solving the $|\mathcal{I}|$ FOPs.

on the two parameters, θ_1 and θ_2, are shown in Figures 3a and 3d, whereas the zoomed-in region are shown in Figures 3b and 3e, respectively. As mentioned, both parameters are identifiable with finite and small OA-CIs. We further evaluate the outer-approximation PL and the approximate CIs across the BO iterations, as shown in Figures 3c and 3f. We observe a monotonic decrease in the width of the OA-CI; the shrinking CI can be attributed to a more accurate GP surrogate of the loss function as the number of BO iterations increases.

Next, again using the two-dimensional case as an example, we test the algorithm performance under varying noise levels (σ) in the observed decisions (Figure 4). The average testing decision and θ error increase as the noise level increases; nonetheless, all conditions converge to the lowest testing errors that can be reached ($\sim \sigma^2$) at 250 iterations. We further perform the PL analysis to the same random instance discussed in Figure 3 at different BO iterations to trace the changes in the parameter CIs over time, as shown in (Figures 4c and 4d). While we observe wider OA-CIs under higher noise levels, which aligns with the traditional PL analysis (Raue et al., 2009)

where decreasing data quality reduces the parameter identifiability with a wider CI, the estimated parameters remain structurally identifiable under the high noise level.

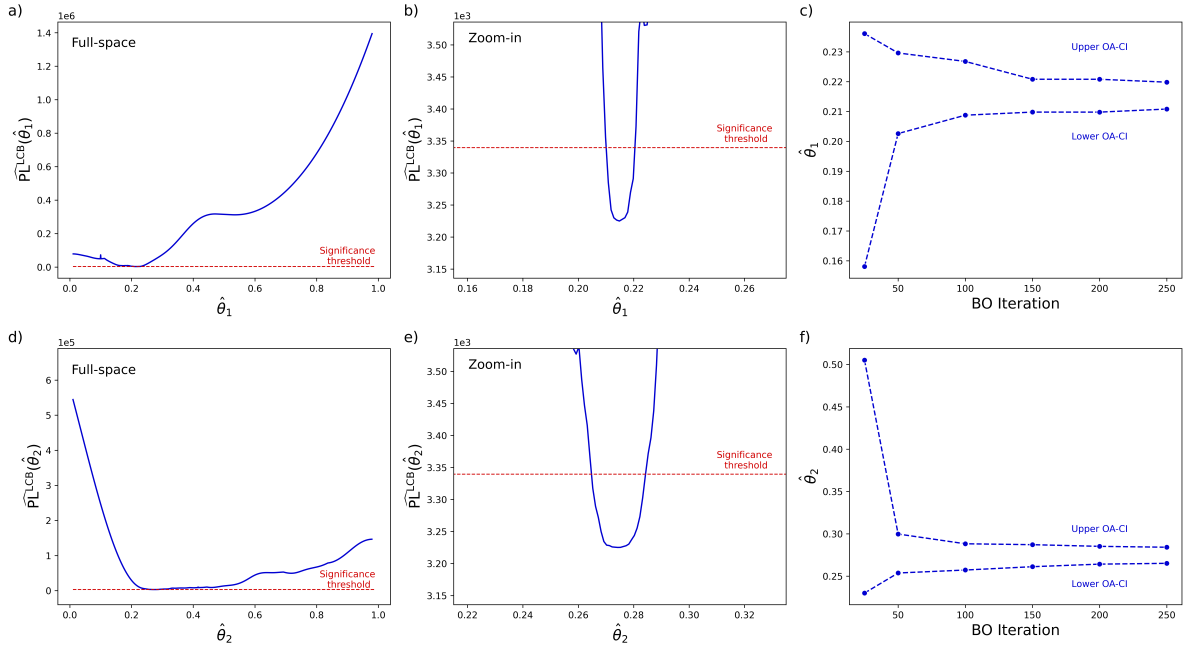

Figure 3: Approximation of profile likelihood and identifiability of parameter estimates in a two-dimensional FBA problem ($d = 2$). a) and d) show the full-space $\widehat{\text{PL}}^{\text{LCB}}$ of θ_1 and θ_2, whereas b) and d) are the zoomed-in regions around the significant threshold level as defined in (7). c) and f) trace the changes in the upper and lower bounds of the OA-CIs of two parameters over the BO iterations. The PL analysis is performed using $\Delta_\alpha = \chi^2(0.05, 1)$ and $\rho = 3.84$ (95% confidence level).

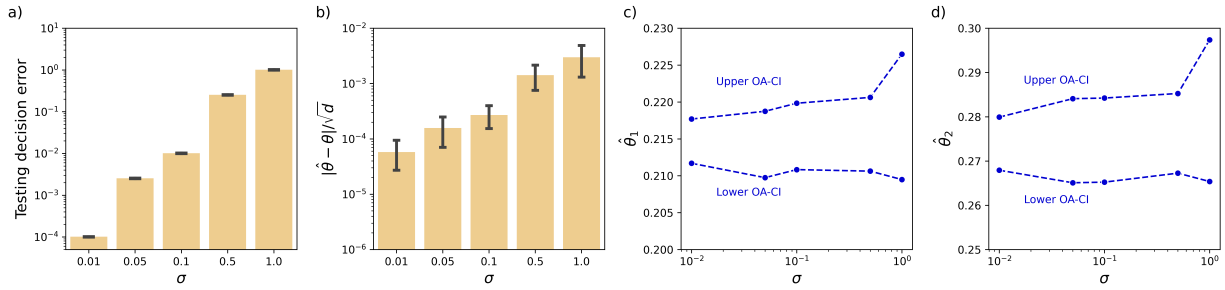

Figure 4: Impact of noise levels on the BO4IO performance. (a) Testing decision error and b) parameter error at 250 iterations under varying noise levels. Outer-approximation confidence intervals (OA-CIs) of two parameters, (b) θ_1 and (c) θ_2, under varying noise levels (σ). OA-CIs are derived using $\Delta_\alpha = \chi^2(0.05, 1)$ and $\rho = 3.84$ (95% confidence level).

5.2 Case study 2: Learning market demands in the standard pooling problem

In the second case study, we consider a standard pooling problem where an operator blends a set of feedstocks in a pooling network to create different final products that meet desired qualities and demands while minimizing the total cost. Provided below is the formulation of a standard pooling problem (Misener and Floudas, 2009):

$$\underset{f,y,z,q}{\text{minimize}} \quad \sum_{(s,l)\in T_f} \eta_s f_{sl} - \sum_{(l,j)\in T_y} \phi^y_{lj} y_{lj} - \sum_{(s,j)\in T_z} \phi^z_{sj} z_{sj} \tag{10a}$$

$$\text{subject to} \quad \sum_{l:(s,l)\in T_f} f_{sl} + \sum_{j:(s,j)\in T_z} z_{sj} \le A^U_s \quad \forall s \in \mathcal{S} \tag{10b}$$

$$\sum_{s:(s,l)\in T_f} f_{sl} - \sum_{j:(l,j)\in T_y} y_{lj} = 0 \quad \forall l \in \mathcal{L} \tag{10c}$$

$$\sum_{s:(s,l)\in T_f} C_{sk} f_{sl} - p_{lk} \sum_{j:(l,j)\in T_y} y_{lj} = 0 \quad \forall l \in \mathcal{L},\ k \in \mathcal{K} \tag{10d}$$

$$\sum_{l:(l,j)\in T_y} p_{lk} y_{lj} + \sum_{s:(s,j)\in T_z} C_s z_{sj} \le P^U_{jk} \left(\sum_{l:(l,j)\in T_y} y_{lj} + \sum_{s:(s,j)\in T_z} z_{sj} \right) \quad \forall j \in \mathcal{J},\ k \in \mathcal{K} \tag{10e}$$

$$\sum_{l:(l,j)\in T_y} y_{lj} + \sum_{s:(s,j)\in T_z} z_{sj} \le \theta_j \quad \forall j \in \mathcal{J} \tag{10f}$$

$$f_{sl} \ge 0 \quad \forall s \in \mathcal{S},\ l \in \mathcal{L} \tag{10g}$$

$$y_{lj} \ge 0 \quad \forall l \in \mathcal{L},\ j \in \mathcal{J} \tag{10h}$$

$$z_{sj} \ge 0 \quad \forall s \in \mathcal{S},\ j \in \mathcal{J} \tag{10i}$$

$$p_{lk} \ge 0 \quad \forall l \in \mathcal{L},\ k \in \mathcal{K}, \tag{10j}$$

where $\mathcal{S}$, $\mathcal{L}$, $\mathcal{J}$, and $\mathcal{K}$ are the sets of input feedstocks, mixing pools, output products, and quality attributes, respectively. As incoming feedstocks can connect to a pool or directly to an output, sets T_f, T_y, and T_z denote the existing streams from input s to pool l, pool l to output j, and input s to output j, respectively. The quality per unit of feedstock s of attribute k is denoted as C_{sk}. The cost per unit of feedstock s is denoted as η_s. The revenue per unit flow from pool l to output j and input s to output j are denoted by ϕ^y_{ij} and ϕ^z_{sj}, respectively. The upper limit of the quality attribute k in each output product j is noted as P^U_{jk}. Decision variables include f_{sl}, y_{lj}, and z_{sj} denoting the flow from input s to pool l, pool l to product j, and input s to output j, respectively, whereas the quality level in pool l of attribute k is denoted by p_{lk}.

In problem (10), we assume that each feedstock s has limited availability A^U_s as indicated in constraints (10b); the material and quality balance at pool l are maintained through constraints (10c) and (10d), respectively; the upper acceptable product quality constraint is set in (10e). We consider a scenario in which the demand for each product j (θ_j in constraints (10f)) is unknown as the unknown dimension denoted as d. Notably, constraints (10e) contain bilinear terms that render the overall optimization problem nonconvex. The goal is to apply BO4IO to estimate the θ_j values by observing a part of the decisions, namely f_i and y_i, based on varying input conditions $u_i = (A^U, P^U, \eta, \phi^y, \phi^z)_i$ in a set of observations $\mathcal{I}$.

We test the proposed BO4IO framework on two benchmark pooling networks, Haverly1 (Haverly, 1978) and Foulds3 (Foulds et al., 1992) where their network specifications ($|\mathcal{S}|$,$|\mathcal{L}|$,$|\mathcal{J}|$,$|\mathcal{K}|$) are (3,1,2,1) and (32,8,16,1), respectively. For each random instance of the IOP, the synthetic dataset is generated as follows. We first generate a set of ground-truth $\theta_j \sim \mathcal{U}(0.5, 1.0)$ for every $j \in \mathcal{J}$ with randomized feedstock price $\eta \sim \mathcal{U}(\bar{\eta}^{\min}, \bar{\eta}^{\max})$ and product revenue $\phi lj^y, \phi sj^z \sim \mathcal{U}(\bar{\phi}^{\min}, \bar{\phi}^{\max})$, where $\bar{\eta}^{\min/\max}$ and $\bar{\phi}^{\min/\max}$ denote the minimum or maximum of the nominal η and ϕ values in the original problems. Using the same values of $(\theta, \eta, \phi^y, \phi^z)$ and the nominal values of P^U, we solved (10) with randomized availability $A^U \sim \mathcal{U}(0.5, 1.0)$ to collect the true optimal solutions $x_i^* = (f_i^*, y_i^*)$ for every $i \in \mathcal{I}$. Lastly, we generated noisy observations of decisions $x_i = x_i^* + \gamma$, where $\gamma \sim \mathcal{N}(0, \sigma^2 \mathbb{I})$ for all $i \in \mathcal{I}$. A separate set of testing data is generated based on the same randomization procedure with 50 experiments. The computational statistics are obtained from the results of 10 random instances of each IOP.

5.2.1 Robust performance against network sizes

We first test how robust the algorithm performance is under different pooling network sizes by applying it to the two benchmark problems with $|\mathcal{I}| = 50$ and $\sigma = 0.05$. To provide a fair comparison between the two problems, we apply the algorithm to learn different dimensionality d of the unknown demands θ in the Foulds3 problem. We trace the progressions of training, testing, and parameter errors over the BO iterations, as shown in Figure 5. The algorithm efficiently learns the unknown parameters regardless of the network complexity, as we observe a similar convergence speed in training and testing errors in the two network problems under the same number of unknown parameters. Consistent with the first case study, a lower convergence rate is observed when the dimensionality increases in the Foulds3 problem. It should be noted that Foulds3 shows a higher parameter error than Haverly1, and the error increases with the dimensionality. We further perform the proposed PL analysis and find that this is due to the larger numbers of non-identifiable parameters in the Foulds3 problems (data not shown).

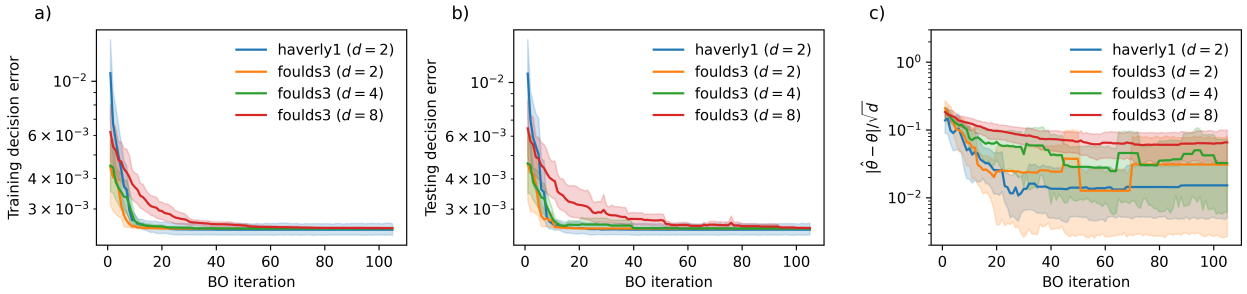

Figure 5: Effect of network size to the BO4IO performance. Convergence analysis of (a) training error, (b) testing error, and (c) parameter error with two benchmark problems, as the numbers shown in the parentheses in the legend denote the varying dimensionality d. Training and testing loss are defined as the average standardized decision error, $\sum_{i\in\mathcal{I}}(x_i - \hat{x}_i)^\top(x_i - \hat{x}_i)/|\mathcal{I}|/|\mathcal{R}|$ and calculated based on the training and testing datasets, respectively, whereas parameter error denotes the difference between the ground-truth (θ) and estimated ($\hat{\theta}^*$) values. Here, the solid lines and shaded areas respectively denote the medians and confidence intervals of the corresponding loss across the 10 random instances. The synthetic dataset of each random instance is generated under the setting of $|\mathcal{I}| = 50$ and $\sigma = 0.05$.

5.2.2 Data efficiency and identifiability analysis

We next test the data efficiency of the algorithm by learning 8 unknown demands in Foulds3 based on different sizes of the training datasets, namely $|\mathcal{I}| = \{10, 25, 50, 100, 200\}$. The convergence profiles for the training and testing errors under different conditions are shown in Figures 6a and 6b. One can see that BO4IO requires only few data points to achieve a low decision error as there is little difference in the convergence rates for the different training dataset sizes and no significant difference in the final testing error as shown in Figure 6c.

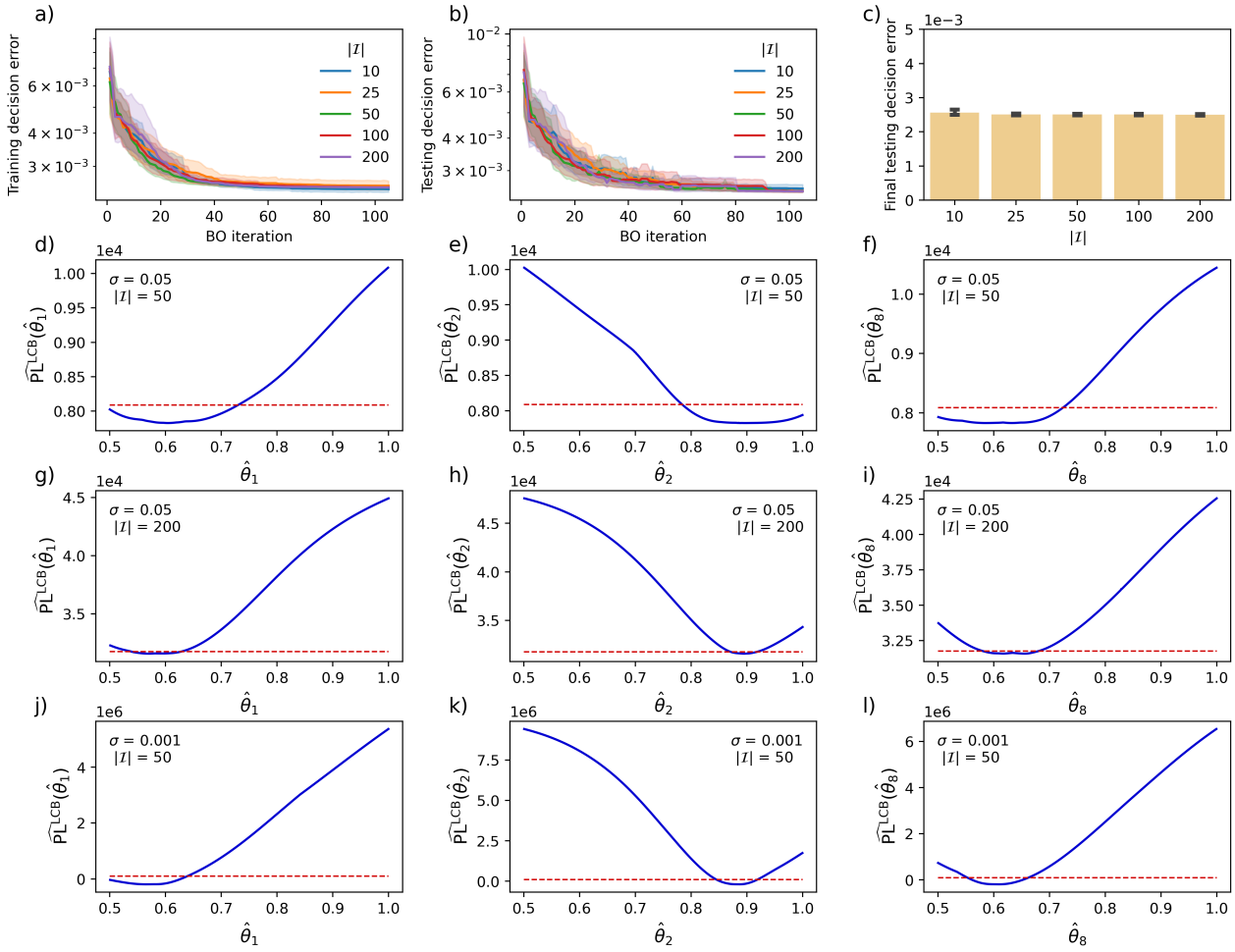

Figure 6: Effect of training dataset size on the BO4IO performance using Foulds3 with $d = 8$ and $\sigma = 0.05$ as an example. Convergence analysis of (a) training error and (b) testing error as well as (c) final testing error at 100 iterations under different sizes of training datasets. d)-f) show the approximate PL of three practically non-identifiable parameters in the selected random IOP instance under $|\mathcal{I}| = 50$ and $\sigma = 0.05$. g)-i) show the approximate PL of the same three parameters as d)-f) estimated under $|\mathcal{I}| = 200$ and $\sigma = 0.05$. j)-l) show the approximate PL of the same three parameters as d)-f) estimated under $|\mathcal{I}| = 50$ and $\sigma = 0.001$. In a) and b), the solid lines and shaded areas respectively denote the medians and confidence intervals of the errors across the 10 random instances. In d)-l), the red dashed line denotes the significance threshold for the identifiability analysis. The PL analysis is performed using $\Delta_\alpha = \chi^2(0.05, 1)$ and $\rho = 3.84$ (95% confidence level).

We further perform the PL analysis to evaluate the impact of training dataset size on parameter identifiability. For simplicity, we select one random IOP instance as an example. The instance shows 5 structurally identifiable parameters (data not shown) and 3 practically non-identifiable parameters (Figure 6d-f) when learned from a training dataset with $|\mathcal{I}| = 50$ and $\sigma = 0.05$. Practical non-identifiability is typically due to insufficient data quality or quantity (Raue et al., 2009). One can see that most of the three practically non-identifiable parameters become structurally identifiable when we further increase the number of data points to $|\mathcal{I}| = 200$ (Figure 6g-i) or reduce the observation noise to $\sigma = 0.001$ ($|\mathcal{I}| = 200$ Figure 6j-l). Notably, the OA-CIs of the 5 other parameters maintain their structural identifiability with smaller OA-CIs (data not shown). The proposed PL analysis provides a means of uncertainty quantification on the BO4IO parameter estimates, and this example demonstrates its importance in assessing the identifiability of the unknown parameters.

5.3 Case study 3: Learning quality requirements in the generalized pooling problem

In the third case study, we apply the BO4IO algorithm to learn the product quality parameters in the constraints of a generalized pooling problem (Misener and Floudas, 2009), which is an extension of (10) with additional discrete variables. The problem can be formulated as the following nonconvex mixed-integer nonlinear program:

$$\underset{f,y,z,q,\gamma^{\text{pool}},\gamma^{\text{init}}}{\text{minimize}} \quad \sum_{(s,l)\in T_f} (\eta_s + \eta^f_{sl}) f_{sl} - \sum_{(l,j)\in T_y} \phi^y_{lj} y_{lj} - \sum_{(s,j)\in T_z} \phi^z_{sj} z_{sj} + \sum_{l\in\mathcal{L}} \eta^{\text{pool}}_l \gamma^{\text{pool}}_l + \sum_{s\in\mathcal{S}} \eta^{\text{init}}_s \gamma^{\text{init}}_s \tag{11a}$$

$$\text{subject to} \quad \sum_{l:(s,l)\in T_f} f_{sl} + \sum_{j:(s,j)\in T_z} z_{sj} \le A^U_s \gamma^{\text{init}}_s \quad \forall s \in \mathcal{S} \tag{11b}$$

$$\sum_{s:(s,l)\in T_f} f_{sl} - \sum_{j:(l,j)\in T_y} y_{lj} = 0 \quad \forall l \in \mathcal{L} \tag{11c}$$

$$\sum_{s:(s,l)\in T_f} C_{sk} f_{sl} - p_{lk} \sum_{j:(l,j)\in T_y} y_{lj} = 0 \quad \forall l \in \mathcal{L},\ k \in \mathcal{K} \tag{11d}$$

$$\sum_{l:(l,j)\in T_y} p_{lk} y_{lj} + \sum_{s:(s,j)\in T_z} C_{sk} z_{sj} \le \theta_{jk} \Big(\sum_{l:(l,j)\in T_y} y_{lj} + \sum_{s:(s,j)\in T_z} z_{sj} \Big) \quad \forall j \in \mathcal{J},\ k \in \mathcal{K} \tag{11e}$$

$$\sum_{l:(l,j)\in T_y} y_{lj} + \sum_{s:(s,j)\in T_z} z_{sj} = D_j \quad \forall j \in \mathcal{J} \tag{11f}$$

$$\sum_{j:(l,j)\in T_y} y_{lj} \le S_l \gamma^{\text{pool}}_l \quad \forall l \in \mathcal{L} \tag{11g}$$

$$\sum_{l:(s,l)\in T_f} f_{sl} \le A^U_s \gamma^{\text{init}}_s \quad \forall s \in \mathcal{S} \tag{11h}$$

$$f_{sl} \ge 0 \quad \forall s \in \mathcal{S},\ l \in \mathcal{L} \tag{11i}$$

$$y_{lj} \ge 0 \quad \forall l \in \mathcal{L},\ j \in \mathcal{J} \tag{11j}$$

$$z_{sj} \ge 0 \quad \forall s \in \mathcal{S},\ j \in \mathcal{J} \tag{11k}$$

$$p_{lk} \ge 0 \quad \forall l \in \mathcal{L},\ k \in \mathcal{K} \tag{11l}$$

$$\gamma_s^{\text{init}} \in \{0,1\} \quad \forall s \in \mathcal{S} \tag{11m}$$

$$\gamma_l^{\text{pool}} \in \{0,1\} \quad \forall l \in \mathcal{L}. \tag{11n}$$

The model is an extension to (10) with some minor changes. Binary variables are introduced to consider the installation decisions of feedstock s and pool l, denoted as γ_s^{init} and γ_l^{pool}, respectively. In addition, new parameters are added to consider the installation cost of feedstock s and pool l, denoted as η_s^{init} and η_l^{pool}, whereas η_{sl}^f is the cost per unit flow from feedstock s to pool l. Two additional constraints (11g) and (11h) are included to constrain the flow variables accounting for the binary feedstock and pool installation, where S_l denotes the pool capacity of pool l.

We consider a benchmark problem called Lee1 (Lee and Grossmann, 2003), which has a network specification ($|\mathcal{S}|$,$|\mathcal{L}|$,$|\mathcal{J}|$,$|\mathcal{K}|$) of (5,4,3,2). This study addresses the scenario in which each product stream j has a demand that needs to be satisfied (D_j in (11f)) with unknown quality requirements (θ_{jk} in (11e)). We assume the the quality requirements θ_{jk} of all attributes sum up to one such that $\sum_{k\in\mathcal{K}} \theta_{j,k} = 1$ for every $j \in \mathcal{J}$. Since Lee1 contains two quality attributes, we only need to estimate one of the quality attributes, and the number of unknown parameters thus reduces to $|\mathcal{J}| = 3$. The synthetic data are generated based on the following procedure. For each instance of IOP, we first generate the ground-truth quality requirements $\theta_{j1} \sim \mathcal{U}(0.2, 0.6)$ and randomized product revenue $\phi_{lj}^y, \phi_{sj}^z \sim \mathcal{U}(0.5\bar{\phi}^{\text{min}}, 1.5\bar{\phi}^{\text{max}})$, where $\bar{\phi}^{\text{min/max}}$ denote the minimum or maximum of the nominal ϕ values in the original problem. Using the fixed θ, ϕ^y, ϕ^z values, we then solve the $|\mathcal{I}|$ FOPs with randomized $A_s^U, D_j \sim \mathcal{U}(0.5, 1.5)$. Similar to the second case study, only a subset of decisions is collected to generate the noisy observations, where $x_i^* = f_i^*$ and $x_i = x_i^* + \gamma$ with $\gamma \sim \mathcal{N}(0, \sigma^2\mathbb{I})$ for all $i \in \mathcal{I}$. A separate testing dataset of 50 experiments is generated to validate the model prediction.

We apply the algorithm to 10 random instances with $|\mathcal{I}| = 50$ and $\sigma = 0.05$. The convergence plots are shown in Figure 7. The training and testing decision errors converge to a low level within 200 iterations. Compared to the standard pooling problem with a similar specification and number of unknowns (e.g. Haverly1), the algorithm converges more slowly and achieves a higher parameter error in the generalized pooling problem. To further investigate the cause of the slow convergence, we plot the true loss function as a function of each unknown parameter (while keeping the other parameters at their best-found values) for Lee1 and Haverly1, and compare it with the predicted GP posterior, as shown in Figures 8 and 9. In the generalized pooling case (Figure 8), we observe discrete (step) changes in the loss function values, and the confidence intervals of the GP posterior do not fully cover the true function. Despite the nonsmoothness of the true loss function, the GP model provides a continuous approximation and leads to a parameter estimate that is close to the true minimum, albeit with a slightly higher parameter error compared to the standard pooling problem. In contrast, the GP posterior provides a very good approximation to the true loss function in the standard pooling problem (Figure 9) since the loss function is smooth, and the algorithm finds a solution very close to the global optimum. This analysis indicates that the slow convergence in the generalized pooling problem could be attributed to the discrete nature of the mixed-integer problem, where the poor GP fitness to the discontinuous and nonsmooth region can potentially hinder the algorithm. One may consider nonstationary kernels (Paciorek and Schervish, 2003) or other surrogate types, e.g. deep GP or Bayesian neural networks (Damianou and Lawrence, 2013; Springenberg et al., 2016), to capture the nonsmoothness and discontinuity

of the target function, which we plan to explore more in future work.

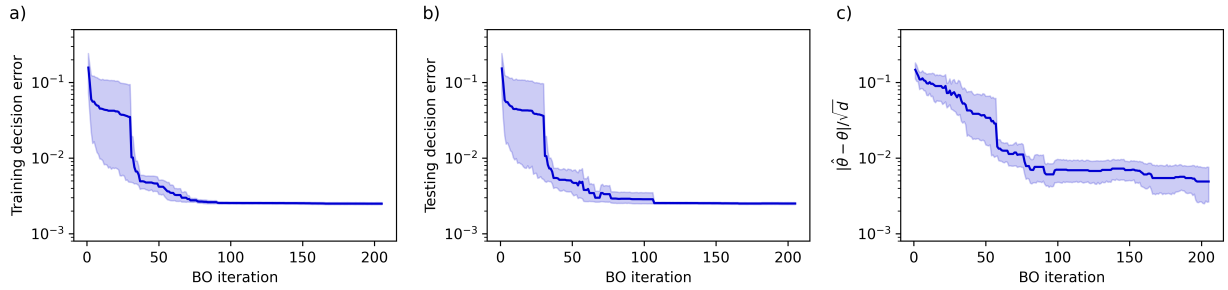

Figure 7: Algorithm performance for the generalized pooling problem Lee1. Convergence analysis of (a) training error, (b) testing error, and (c) parameter error with three unknown parameters. Training and testing errors are defined as the average standardized decision errors, $\sum_{i\in\mathcal{I}}(x_i - \hat{x}_i)^\top(x_i - \hat{x}_i)/|\mathcal{I}|/|\mathcal{R}|$ and calculated based on the training and testing datasets, respectively, whereas parameter error denotes the difference between the ground-truth (θ) and estimated ($\hat{\theta}^*$) values. Here, the solid lines and shaded areas respectively denote the medians and confidence intervals of the corresponding loss across the 10 random instances. The synthetic dataset of each random instance is generated under the setting of $|\mathcal{I}| = 50$ and $\sigma = 0.05$.

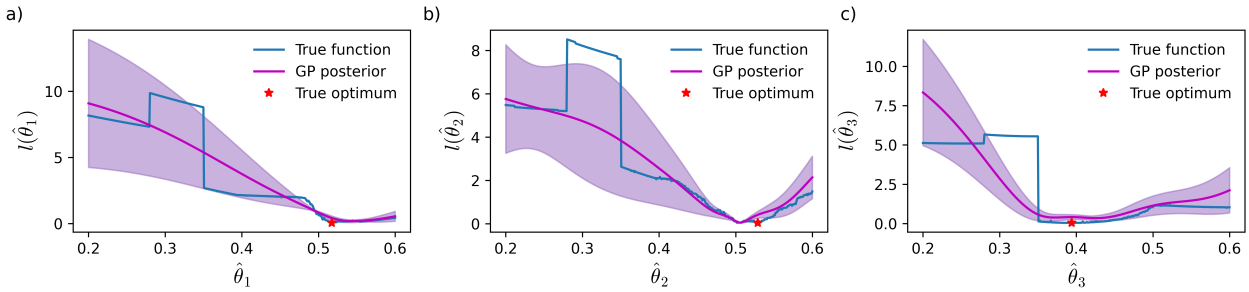

Figure 8: Comparison of the changes in the true loss function value (blue curve) and the predicted GP posterior around the best-found solution in the generalized pooling problem (Lee1). The purple lines and shaded areas denote the means and confidence intervals of the predicted GP posteriors.

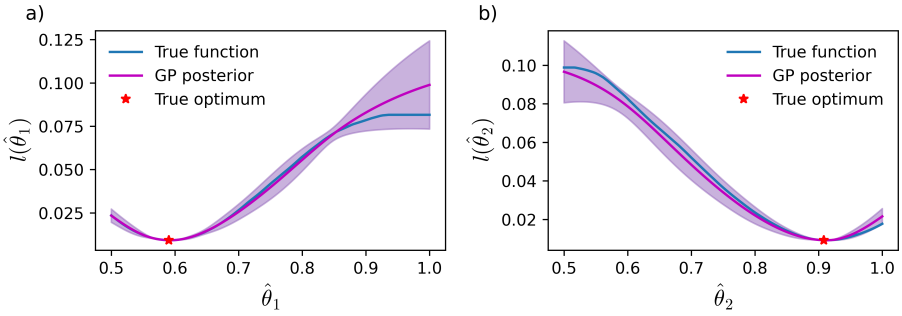

Figure 9: Comparison of the changes in the true loss function value (blue curve) and the predicted GP posterior around the best-found solution in the standard pooling problem (Haverly1). The purple lines and shaded areas denote the means and confidence intervals of the predicted GP posteriors.

6 Conclusions

In this work, we developed a Bayesian optimization approach for data-driven inverse optimization problem, which aims to infer unknown parameters of an optimization model from observed decisions and is commonly formulated as a large-scale bilevel optimization problem. In the proposed BO4IO framework, the IO loss function is treated as a black box and approximated with a Gaussian process model. At each iteration of the algorithm, the loss function is evaluated and the next candidate solution is determined by minimizing an acquisition function based on the GP posterior. The key advantage of BO4IO is that the evaluation of the loss function can be achieved by directly solving the forward optimization problem with the current parameter estimates for every data point, which circumvents the need for a complex single-level reformulation of the bilevel program or a special, e.g. cutting-plane-based, algorithm. Consequently, BO4IO remains applicable even when the FOP is nonlinear and nonconvex, and it can consider both continuous and discrete decision variables. Furthermore, we leverage the GP posterior to derive confidence bounds on the profile likelihood to compute inner and outer approximations of the confidence interval on each parameter estimate, which provides a means of uncertainty quantification and assessing the identifiability of the unknown parameters.

We tested the proposed algorithm and PL method in three computational case studies. The extensive computational results demonstrate the efficacy of BO4IO in estimating unknown parameters in various classes of FOPs, ranging from convex nonlinear to nonconvex mixed-integer nonlinear programs. In the tested instances, the algorithm proved to be efficient in finding parameter estimates with very low prediction errors in relatively few ($\sim$100) BO iterations. Using the proposed PL analysis, we were able to determine the different extents of identifiability for the unknown parameters and also demonstrated how the identifiability of a parameter can change with the quantity and quality of data.

Finally, we would like to highlight that several important directions for future work remain, including integrating high-dimensional BO techniques (Snoek et al., 2015; Eriksson and Jankowiak, 2021) for learning large sets of parameter and developing a customized surrogate for mixed-integer problems to further improve the algorithm performance.

Acknowledgement

The authors thank Vikram Kumar, who was an undergraduate student at the University of Minnesota at the time, for conducting preliminary computational results for this work. The authors gratefully acknowledge the support from the National Science Foundation under Grants #2044077 and #2237616. Computational resources were provided by the Minnesota Supercomputing Institute at the University of Minnesota.

References

Ravindra K Ahuja and James B Orlin. Inverse optimization. *Operations research*, 49(5):771–783, 2001.

Temitayo Ajayi, Taewoo Lee, and Andrew J Schaefer. Objective selection for cancer treatment: An inverse optimization approach. *Operations Research*, 70(3):1717–1738, 2022.

Anil Aswani, Zuo-Jun Shen, and Auyon Siddiq. Inverse optimization with noisy data. *Operations Research*, 66(3):870–892, 2018.

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

Eva Balsa-Canto, Antonio A Alonso, and Julio R Banga. Computational procedures for optimal experimental design in biological systems. *IET systems biology*, 2(4):163–172, 2008.

Samuel Bandara, Johannes P Schlöder, Roland Eils, Hans Georg Bock, and Tobias Meyer. Optimal experimental design for parameter estimation of a cell signaling model. *PLoS computational biology*, 5(11):e1000558, 2009.

Ror Bellman and Karl Johan Åström. On structural identifiability. *Mathematical biosciences*, 7(3-4): 329–339, 1970.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. No-regret bayesian optimization with unknown hyperparameters. *Journal of Machine Learning Research*, 20(50):1–24, 2019.

Dimitris Bertsimas, Vishal Gupta, and Ioannis Ch Paschalidis. Data-driven estimation in equilibrium using inverse optimization. *Mathematical Programming*, 153:595–633, 2015.

Burcu Beykal, Styliani Avraamidou, Ioannis PE Pistikopoulos, Melis Onel, and Efstratios N Pistikopoulos. Domino: Data-driven optimization of bi-level mixed-integer nonlinear problems. *Journal of Global Optimization*, 78:1–36, 2020.

Hendrik PJ Bonarius, Vassily Hatzimanikatis, Koen PH Meesters, Cornelis D De Gooijer, Georg Schmid, and Johannes Tramper. Metabolic flux analysis of hybridoma cells in different culture media using mass balances. *Biotechnology and bioengineering*, 50(3):299–318, 1996.

Anthony P Burgard and Costas D Maranas. Optimization-based framework for inferring and testing hypothesized metabolic objective functions. *Biotechnology and bioengineering*, 82(6):670–677, 2003.

Didier Burton and Ph L Toint. On an instance of the inverse shortest paths problem. *Mathematical programming*, 53:45–61, 1992.

Michael L. Bynum, Gabriel A. Hackebeil, William E. Hart, Carl D. Laird, Bethany L. Nicholson, John D. Siirola, Jean-Paul Watson, and David L. Woodruff. *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, third edition, 2021.

Timothy CY Chan and Neal Kaw. Inverse optimization for the recovery of constraint parameters. *European Journal of Operational Research*, 282(2):415–427, 2020.

Timothy CY Chan, Tim Craig, Taewoo Lee, and Michael B Sharpe. Generalized inverse multi-objective optimization with application to cancer therapy. *Operations Research*, 62(3):680–695, 2014.

Timothy CY Chan, Rafid Mahmood, and Ian Yihang Zhu. Inverse optimization: Theory and applications. *Operations Research*, 2023.

Yee Wen Choon, Mohd Saberi Mohamad, Safaai Deris, Rosli Md Illias, Chuii Khim Chong, Lian En Chai, Sigeru Omatu, and Juan Manuel Corchado. Differential bees flux balance analysis with optknock for in silico microbial strains optimization. *PloS one*, 9(7):e102744, 2014.

Claudio Cobelli and Joseph J Distefano III. Parameter and structural identifiability concepts and ambiguities: a critical review and analysis. *American Journal of Physiology-Regulatory, Integrative and Comparative Physiology*, 239(1):R7–R24, 1980.

Andreas Damianou and Neil D Lawrence. Deep gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR, 2013.

Vedat Dogan and Steven Prestwich. Bilevel optimization by conditional bayesian optimization. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 243–258. Springer, 2023.

David Eriksson and Martin Jankowiak. High-dimensional bayesian optimization with sparse axis-aligned subspaces. In *Uncertainty in Artificial Intelligence*, pages 493–503. PMLR, 2021.

Ricardo Fernández-Blanco, Juan Miguel Morales, Salvador Pineda, and Álvaro Porras. Inverse optimization with kernel regression: Application to the power forecasting and bidding of a fleet of electric vehicles. *Computers & Operations Research*, 134:105405, 2021.

Leslie Richard Foulds, Dag Haugland, and Kurt Jørnsten. A bilinear approach to the pooling problem. *Optimization*, 24(1-2):165–180, 1992.

Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Peter I Frazier and Jialei Wang. Bayesian optimization for materials design. *Information science for materials discovery and design*, pages 45–75, 2016.

Carlos Eduardo García Sánchez and Rodrigo Gonzalo Torres Sáez. Comparison and analysis of objective functions in flux balance analysis. *Biotechnology progress*, 30(5):985–991, 2014.

Carlos Eduardo García Sánchez, César Augusto Vargas García, and Rodrigo Gonzalo Torres Sáez. Predictive potential of flux balance analysis of saccharomyces cerevisiae using as optimization function combinations of cell compartmental objectives. 2012.

Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in neural information processing systems*, 31, 2018.

Archis Ghate. Imputing radiobiological parameters of the linear-quadratic dose-response model from a radiotherapy fractionation plan. *Physics in Medicine & Biology*, 65(22):225009, 2020.

Kimia Ghobadi and Houra Mahmoudzadeh. Inferring linear feasible regions using inverse optimization. *European Journal of Operational Research*, 290(3):829–843, 2021.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Stewart Greenhill, Santu Rana, Sunil Gupta, Pratibha Vellanki, and Svetha Venkatesh. Bayesian optimization for adaptive experimental design: A review. *IEEE access*, 8:13937–13948, 2020.

Rishabh Gupta and Qi Zhang. Decomposition and adaptive sampling for data-driven inverse linear optimization. *INFORMS Journal on Computing*, 34(5):2720–2735, 2022.

Rishabh Gupta and Qi Zhang. Efficient learning of decision-making models: A penalty block coordinate descent algorithm for data-driven inverse optimization. *Computers & Chemical Engineering*, 170:108123, 2023.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL https://www.gurobi.com.

Co A Haverly. Studies of the behavior of recursion for the pooling problem. *Acm sigmap bulletin*, (25):19–28, 1978.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 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*, pages 507–523. Springer, 2011.

Waltraud Huyer and Arnold Neumaier. Snobfit–stable noisy optimization by branch and fit. *ACM Transactions on Mathematical Software (TOMS)*, 35(2):1–25, 2008.

Garud Iyengar and Wanmo Kang. Inverse conic programming with applications. *Operations Research Letters*, 33(3):319–330, 2005.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13:455–492, 1998.

Milind Joshi, Andreas Seidel-Morgenstern, and Andreas Kremling. Exploiting the bootstrap method for quantifying parameter confidence intervals in dynamical systems. *Metabolic engineering*, 8(5):447–455, 2006.

Benjamin Karg and Sergio Lucia. Efficient representation and approximation of model predictive control laws via deep learning. *IEEE Transactions on Cybernetics*, 50(9):3866–3878, 2020.

Arezou Keshavarz, Yang Wang, and Stephen Boyd. Imputing a convex objective function. In *2011 IEEE international symposium on intelligent control*, pages 613–619. IEEE, 2011.

Emmanuel Kieffer, Grégoire Danoy, Pascal Bouvry, and Anass Nagih. Bayesian optimization approach of general bi-level problems. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1614–1621, 2017.

Zachary A King, Justin Lu, Andreas Dräger, Philip Miller, Stephen Federowicz, Joshua A Lerman, Ali Ebrahim, Bernhard O Palsson, and Nathan E Lewis. Bigg models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic acids research*, 44(D1):D515–D522, 2016.

Clemens Kreutz and Jens Timmer. Systems biology: experimental design. *The FEBS journal*, 276 (4):923–942, 2009.

Akshay Kudva, Farshud Sorourifar, and Joel A Paulson. Constrained robust bayesian optimization of expensive noisy black-box functions with guaranteed regret bounds. *AIChE Journal*, 68 (12):e17857, 2022.

Akshay Kudva, Wei-Ting Tang, and Joel A Paulson. Robust bayesian optimization for flexibility analysis of expensive simulation-based models with rigorous uncertainty bounds. *Computers & Chemical Engineering*, 181:108515, 2024.

Deok-Sun Lee, Henry Burd, Jiangxia Liu, Eivind Almaas, Olaf Wiest, Albert-László Barabási, Zoltán N Oltvai, and Vinayak Kapatral. Comparative genome-scale metabolic reconstruction and flux balance analysis of multiple staphylococcus aureus genomes identify novel antimicrobial drug targets. *Journal of bacteriology*, 191(12):4015–4024, 2009.

Kyongbum Lee, Daehee Hwang, Tadaaki Yokoyama, George Stephanopoulos, Gregory N Stephanopoulos, and Martin L Yarmush. Identification of optimal classification functions for biological sample and state discrimination from metabolic profiling data. *Bioinformatics*, 20(6): 959–969, 2004.

Sangbum Lee and Ignacio E Grossmann. Global optimization of nonlinear generalized disjunctive programming with bilinear equality constraints: applications to process networks. *Computers & chemical engineering*, 27(11):1557–1575, 2003.

Nathan E Lewis, Kim K Hixson, Tom M Conrad, Joshua A Lerman, Pep Charusanti, Ashoka D Polpitiya, Joshua N Adkins, Gunnar Schramm, Samuel O Purvine, Daniel Lopez-Ferrer, et al. Omic data from evolved e. coli are consistent with computed optimal growth from genome-scale models. *Molecular systems biology*, 6(1):390, 2010.

Jonathan Yu-Meng Li. Inverse optimization of convex risk functions. *Management Science*, 67(11): 7113–7141, 2021.

Tim Maiwald, Helge Hass, Bernhard Steiert, Joep Vanlier, Raphael Engesser, Andreas Raue, Friederike Kipkeew, Hans H Bock, Daniel Kaschek, Clemens Kreutz, et al. Driving the model to its limit: profile likelihood based model reduction. *PloS one*, 11(9):e0162366, 2016.

Ruth Misener and Christodoulos A Floudas. Advances for the pooling problem: Modeling, global optimization, and computational studies. *Applied and Computational Mathematics*, 8(1):3–22, 2009.

Jonas Mockus. Application of bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization*, 4:347–365, 1994.

Peyman Mohajerin Esfahani, Soroosh Shafieezadeh-Abadeh, Grani A Hanasusanto, and Daniel Kuhn. Data-driven inverse optimization with imperfect information. *Mathematical Programming*, 167:191–234, 2018.

Deepak Nagrath, Marco Avila-Elchiver, François Berthiaume, Arno W Tilles, Achille Messac, and Martin L Yarmush. Soft constraints-based multiobjective framework for flux balance analysis. *Metabolic engineering*, 12(5):429–445, 2010.

Michael C Neale and Michael B Miller. The use of likelihood-based confidence intervals in genetic models. *Behavior genetics*, 27:113–120, 1997.

Avlant Nilsson and Jens Nielsen. Genome scale metabolic modeling of cancer. *Metabolic engineering*, 43:103–112, 2017.

Jeffrey D Orth, Ronan MT Fleming, and Bernhard Ø Palsson. Reconstruction and use of microbial metabolic networks: the core escherichia coli metabolic model as an educational guide. *EcoSal plus*, 4(1):10–1128, 2010a.

Jeffrey D Orth, Ines Thiele, and Bernhard Ø Palsson. What is flux balance analysis? *Nature biotechnology*, 28(3):245–248, 2010b.

Christopher Paciorek and Mark Schervish. Nonstationary covariance functions for gaussian process regression. *Advances in neural information processing systems*, 16, 2003.

Filipa Pardelha, Maria GE Albuquerque, Maria AM Reis, João ML Dias, and Rui Oliveira. Flux balance analysis of mixed microbial cultures: Application to the production of polyhydroxyalkanoates from complex mixtures of volatile fatty acids. *Journal of biotechnology*, 162(2-3):336–345, 2012.

Andreas Raue, Clemens Kreutz, Thomas Maiwald, Julie Bachmann, Marcel Schilling, Ursula Klingmüller, and Jens Timmer. Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. *Bioinformatics*, 25(15):1923–1929, 2009.

Luis Miguel Rios and Nikolaos V Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, 2013.

Joanne M Savinell and Bernhard O Palsson. Network analysis of intermediary metabolism using linear optimization. i. development of mathematical formalism. *Journal of theoretical biology*, 154 (4):421–454, 1992.

Andrew J Schaefer. Inverse integer programming. *Optimization Letters*, 3:483–489, 2009.

Paul JH Schoemaker. The quest for optimality: A positive heuristic of science? *Behavioral and brain sciences*, 14(2):205–215, 1991.

Robert Schuetz, Lars Kuepfer, and Uwe Sauer. Systematic evaluation of objective functions for predicting intracellular fluxes in escherichia coli. *Molecular systems biology*, 3(1):119, 2007.

Robert Schuetz, Nicola Zamboni, Mattia Zampieri, Matthias Heinemann, and Uwe Sauer. Multidimensional optimality of microbial metabolism. *Science*, 336(6081):601–604, 2012.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE transactions on evolutionary computation*, 22 (2):276–295, 2017.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180. PMLR, 2015.

Michael E Sobel. Asymptotic confidence intervals for indirect effects in structural equation models. *Sociological methodology*, 13:290–312, 1982.

Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. *Advances in neural information processing systems*, 29, 2016.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Haoran Sun, Xiangyi Chen, Qingjiang Shi, Mingyi Hong, Xiao Fu, and Nicholas D Sidiropoulos. Learning to optimize: Training deep neural networks for interference management. *IEEE Transactions on Signal Processing*, 66(20):5438–5453, 2018.

El-Ghazali Talbi. A taxonomy of metaheuristics for bi-level optimization. In *Metaheuristics for bi-level optimization*, pages 1–39. Springer, 2013.

Korkut Uygun, Howard WT Matthew, and Yinlun Huang. Investigation of metabolic objectives in cultured hepatocytes. *Biotechnology and bioengineering*, 97(3):622–637, 2007.

Lizhi Wang. Cutting plane algorithms for the inverse mixed integer linear programming problem. *Operations research letters*, 37(2):114–116, 2009.

Franz-Georg Wieland, Adrian L Hauber, Marcus Rosenblatt, Christian Tönsing, and Jens Timmer. On structural and practical identifiability. *Current Opinion in Systems Biology*, 25:60–69, 2021.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Shi Yu, Haoran Wang, and Chaosheng Dong. Learning risk preferences from investment portfolios using inverse optimization. *Research in International Business and Finance*, 64:101879, 2023.

Jianzhong Zhang and Chengxian Xu. Inverse optimization for linearly constrained convex separable programming problems. *European Journal of Operational Research*, 200(3):671–679, 2010.

Jianzhong Zhang and Liwei Zhang. An augmented lagrangian method for a class of inverse quadratic programming problems. *Applied Mathematics and Optimization*, 61(1):57, 2010.

Jing Zhang, Sepideh Pourazarm, Christos G Cassandras, and Ioannis Ch Paschalidis. The price of anarchy in transportation networks: Data-driven evaluation and reduction strategies. *Proceedings of the IEEE*, 106(4):538–553, 2018.

Qi Zhao, Arion I Stettner, Ed Reznik, Ioannis Ch Paschalidis, and Daniel Segrè. Mapping the landscape of metabolic goals of a cell. *Genome biology*, 17:1–11, 2016.