Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

POLICY-AUGMENTED BAYESIAN NETWORK OPTIMIZATION WITH GLOBAL CONVERGENCE

Junkai Zhao
Jun Luo

Antai College of Economics and Management
Shanghai Jiao Tong University
1954 Huashan Road
Shanghai, 200030, China

Wei Xie

Mechanical and Industrial Engineering
Northeastern University
360 Huntington Avenue
Boston, MA 02115, USA

ABSTRACT

Driven by critical challenges in biomanufacturing, including high complexity and high uncertainty, we propose global optimization methods on the policy-augmented Bayesian network (PABN), characterizing risk- and science-based understanding of underlying bioprocess mechanisms, to guide the optimal control. We first develop a sequential optimization algorithm based on deep kernel learning (DKL) for PABN with general state transition dynamics, which can learn the spatial dependence of mean response through a deep neural network. In addition, to improve the interpretability and computational efficiency of policy optimization, a global metamodel is introduced to guide linear Gaussian PABN optimization, which explicitly accounts for the correlation of input-to-output pathways obtained under different candidate policies. Our empirical study provides the ablation analysis and the interpretation analysis of the DKL, and also shows that both proposed approaches demonstrate promising performance compared to the standard Bayesian optimization with Gaussian process.

1 INTRODUCTION

The biopharmaceutical manufacturing industry plays an important role in supporting public health and economic growth. However, biomanufacturing faces several critical challenges, including high complexity, high variability, and very limited process observations (Hong et al. 2018). The trajectory dynamics in the biomanufacturing processes are determined by sophisticated time-varying mechanisms, which are highly complex and variable (Kasemiire et al. 2021). Due to the long analytical testing times required for biopharmaceutical materials, historical process data are often very limited, which makes the modeling and control of biomanufacturing processes challenging (Gottschalk et al. 2012).

Existing biomanufacturing process modeling methodologies can be categorized into two classes: first-principle models and data-driven models. First-principle models rely on ordinary or partial differential equations (ODEs/PDEs) based mechanistic models representing the dynamics of bioprocesses (Luo et al. 2021). Compared to the black-box simulation model (Cheng et al. 2023), they are often built on the scientific understanding of the causal relationships between the key factors in the bioprocesses. However, first-principle models often cannot provide good predictions due to the limitations of existing scientific understanding. In addition, they are usually deterministic and ignore model estimation uncertainty. Data-driven models aim to build general statistical and machine learning models based on real-world data (Park et al. 2021). The main drawback of data-driven models is that they are not easily interpretable and often require sufficient historical data (Del Rio-Chanona et al. 2019).

Driven by these challenges and limitations of existing methodologies, Zheng et al. (2023) propose the policy-augmented Bayesian network (PABN), which represents the causal interactions and dynamics

within and between different unit operations to guide bioprocess control. It is a hybrid ("mechanistic + statistical") model that can leverage existing kinetic models and facilitate learning from data. Such temporal graphical models with linear Gaussian dependencies are often called Kalman filters, and their nonlinear variants are called extended Kalman filters (Koller and Friedman 2009). The Kalman filter and its variants are widely applied to obtain good de-noising estimation and filtering when the system model has small uncertainty. However, compared to the PABN model, they may perform poorly in the presence of limited data and large model uncertainty (Yi and Zorzi 2021). Also, they often focus on the state estimation and ignore the policy optimization. Therefore, we model the biomanufacturing process as PABN suggested by Zheng et al. (2023).

To search for the optimal control policy, Zheng et al. (2023) introduce a projected stochastic gradient ascent approach to maximize the expected cumulative reward over the space of parametric policies. However, the state transition function of the PABN considered in their study is linear, and the gradient-based policy optimization method can only guarantee local convergence. Bayesian optimization (BO) considers the problem of finding a global optimum of an unknown objective function (Frazier 2018). Astudillo and Frazier (2021) propose a Bayesian optimization approach for function networks, where each function takes the output of its parent nodes as input. Although the structure of function network is similar to PABN, their study assumes that the function in each layer of the network is known, which cannot be directly applied in biomanufacturing. Bowden et al. (2021) introduce a deep kernel Bayesian optimization approach, which adopts the deep kernel combining classical kernel with the deep neural network. However, Bayesian optimization with the deep kernel has not been explored in the context of simulation optimization, and there is limited investigation into the impact of neural network structure and interpretation of the deep kernel.

In this study, we aim to find the global optimal policy in PABN. We first consider the PABN with general nonlinear state transition functions and propose a Bayesian optimization approach to search for the optimal policy parameters. The classical kernel function used in the Gaussian process (GP) metamodel only depends on the distance between candidate solutions, which makes it difficult to capture the global spatial interdependence within PABN. Instead, we adopt the deep kernel, which learns the spatial interdependence of PABN through a deep neural network. Compared with the standard GP assisted Bayesian optimization, it takes advantage of the flexibility and expressive power of neural networks to explore the general spatial interdependence of the mean response. We also explore the impact of the neural network architecture and the interpretation of the output from the deep kernel, which shows that the spatial covariance learned by the deep kernel is associated with the pathway similarity under different candidate policies. However, since the deep kernel learning (DKL) does not incorporate the dynamics of PABN explicitly, it may lack interpretability. Therefore, we propose another interpretable policy optimization approach for linear Gaussian PABN. Built on a predictor of the mean response derived explicitly to account for input-output pathway correlation and the global interdependency of PABN, this approach can achieve global convergence with fewer samples. By reusing the calculations, the computational cost can be reduced significantly.

The paper is organized as follows. In Section 2, we introduce the general PABN model. We propose the DKL-based sequential optimization for general PABN in Section 3. In Section 4, we focus on the linear Gaussian PABN, derive an interpretable global metamodel predictor, and develop the optimization algorithm which can improve interpretability and computational efficiency. Then, we study the empirical performance of two proposed approaches in Section 5 and conclude this paper in Section 6.

2 GENERAL POLICY-AUGMENTED KNOWLEDGE GRAPH

In this section, we first review the PABN introduced in Zheng et al. (2023). A typical biomanufacturing system consists of multiple unit operations, including upstream fermentation and downstream purification to meet quality requirements; see Figure 1. The outputs of the biomanufacturing system (e.g., drug quality and productivity) are impacted by many interacting factors. In general, these factors can be divided into critical process parameters (CPPs) and critical quality attributes (CQAs). For the purpose of this discussion, one can consider CQAs as the "states" of the process, e.g., the concentrations of biomass. CPPs can be

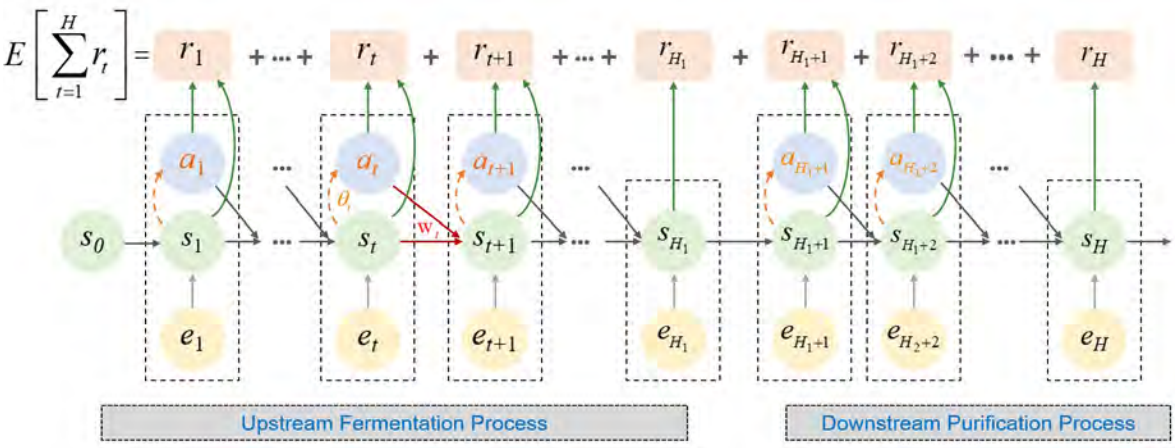


Figure 1: An illustration of a PABN with arrows representing interactions from Zheng et al. (2023).

viewed as "actions" (such as feeding strategies) that need to be controlled to optimize the manufacturing output metrics. We use $\boldsymbol{s}_t = \left(s_t^1, ..., s_t^n\right)$ and $\boldsymbol{a}_t = \left(a_t^1, ..., a_t^m\right)$ to denote, respectively, the state and action vectors at time t, where $1 \leq t \leq H$ with H representing the planning horizon. The dimensions of the state and action variables may be time-dependent, but for simplicity, we keep them constant in this study.

In general, the dynamics of the state in a bioprocess can be characterized by ODE/PDE-based kinetic models. Suppose that $\boldsymbol{s}_t$ evolves according to the ordinary differential equation $\frac{d\boldsymbol{s}_t}{dt} = f_t(\boldsymbol{s}_t, \boldsymbol{a}_t)$, where $f_t(\boldsymbol{s}_t, \boldsymbol{a}_t)$ encodes the causal interdependencies between various CPPs (i.e., $\boldsymbol{s}_t$) and CQAs (i.e., $\boldsymbol{a}_t$) in period t. Suppose that the functional form of $f_t(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{w}_t)$ is known, with unknown parameters $\boldsymbol{w}_t$ estimated from data. Built on the prior knowledge of the existing PDE/ODE-based kinetics, Zheng et al. (2023) introduced the state transition hybrid model, i.e.,

$$\boldsymbol{s}_{t+1} = f_t(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\beta}_t) + \boldsymbol{e}_{t+1}, \tag{1}$$

where the residual $\boldsymbol{e}_{t+1}$ is a random vector representing the uncontrollable factors. Suppose that the residuals follow a multivariate Gaussian distribution, i.e., $\boldsymbol{e} = (\boldsymbol{e}_1^\top, \boldsymbol{e}_2^\top, \ldots, \boldsymbol{e}_H^\top) \sim \mathcal{N}(\mathbf{0}, \mathbf{V})$, where v_{ij} in ith row and jth column of $\mathbf{V}$ denotes covariance between the ith component and the jth component of $\boldsymbol{e}$. Then the distribution of the entire trajectory $\boldsymbol{\tau} = (\boldsymbol{s}_1, \boldsymbol{a}_1, \boldsymbol{s}_2, \boldsymbol{a}_2, ..., \boldsymbol{s}_H)$ of the stochastic decision process (SDP) can be written as $p(\boldsymbol{\tau}) = p(\boldsymbol{s}_1) \prod_{t=1}^{H-1} p(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t) p(\boldsymbol{a}_t)$. Let $\boldsymbol{w} = \{\{\boldsymbol{\beta}_t\}_{t=1}^{H-1}, \mathbf{V}\}$ denote the model parameters. Given the historical data $\mathscr{D} = \left\{\boldsymbol{\tau}^{(n)}\right\}_{n=1}^{R}$, the posterior distribution of model parameters, $p(\boldsymbol{w}|\mathscr{D})$, quantifies the model parameter estimation uncertainty.

At each time t, the decisions are selected according to $\boldsymbol{a}_t = \pi_t(\boldsymbol{s}_t; \boldsymbol{\theta}_t)$, where the policy π_t maps the state vector $\boldsymbol{s}_t$ into the space of all possible action values and $\boldsymbol{\theta}_t$ represents the policy parameters. The parametric policy $\pi_{\boldsymbol{\theta}} = \{\pi_t\}_{t=1}^{H-1}$ is the collection of these mappings over the entire planning period and is fully characterized by $\boldsymbol{\theta} = \{\boldsymbol{\theta}_t\}_{t=1}^{H-1}$. Let the reward function at each t-th period be $r_t(\boldsymbol{s}_t, \boldsymbol{a}_t)$ and the random cumulative reward earned by following the policy specified by $\boldsymbol{\theta}$ during the planning period be $r(\boldsymbol{\theta}) = \sum_{t=1}^{H} r_t(\boldsymbol{s}_t, \boldsymbol{a}_t)$, the expected cumulative reward for the given model parameters $\boldsymbol{w}$ can be written as $J(\boldsymbol{\theta}; \boldsymbol{w}) = \mathrm{E}_{\boldsymbol{e}}\left[r(\boldsymbol{\theta}) | \boldsymbol{s}_0, \boldsymbol{w}\right]$. Our goal is to obtain the optimal policy accounting for both model uncertainty and inherent stochasticity, i.e.,

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta} \in \mathscr{P}} \mathscr{J}(\boldsymbol{\theta}) \quad \text{with} \quad \mathscr{J}(\boldsymbol{\theta}) = \mathrm{E}_{\boldsymbol{w}}\left[J(\boldsymbol{\theta}; \boldsymbol{w})\right], \tag{2}$$

with the expectation taken over the posterior distribution of $\boldsymbol{w}$ given the historical data $\mathscr{D}$. Recall that $J(\boldsymbol{\theta}; \boldsymbol{w})$ is an expected value over the stochastic uncertainty. The objective $\mathscr{J}(\boldsymbol{\theta})$ in (2) takes an additional expectation to account for model uncertainty. For linear state transition and policy functions, Zheng et al. (2023) introduce the policy-augmented Bayesian network (PABN) as illustrated in Figure 1.

3 POLICY OPTIMIZATION WITH DKL FOR GENERAL PABN

In this section, we consider the policy optimization for general PABN. The objective function $\mathscr{J}(\boldsymbol{\theta})$ usually has a complex form, which lacks properties like convexity and linearity, and its evaluation is often noisy. Therefore, gradient descent policy optimization methods can get stuck in local optimum (Chau et al. 2014). Bayesian optimization methods have gained considerable attention in the global optimization of complex stochastic systems (Sun et al. 2014). They sequentially optimize a Gaussian process (GP), characterizing the belief of the objective function, by balancing exploration and exploitation to guarantee the global convergence. However, the GP metamodel with classical kernels like the RBF and Matérn kernel are stationary, i.e., the spatial correlation or covariance function depends only on the distance between the inputs. Thus, they are unable to learn the spatial interdependence of the PABN model from the data.

To overcome this limitation, we develop a sequential optimization algorithm with DKL, which combines deep learning with the nonparametric benefit of GP (Wilson et al. 2016). Let $r(\boldsymbol{\theta})$ denote the simulation output of the mean response $\mathscr{J}(\boldsymbol{\theta}) = \mathrm{E}_{\boldsymbol{w},\boldsymbol{e}}[r(\boldsymbol{\theta})]$ at any feasible candidate policy specified by $\boldsymbol{\theta}$, i.e., $r(\boldsymbol{\theta}) = M(\boldsymbol{\theta}) + \varepsilon(\boldsymbol{\theta})$, where $M(\boldsymbol{\theta})$ is a GP with zero mean characterizing the prior knowledge of the unknown mean response surface $\mathscr{J}(\boldsymbol{\theta})$. The term $\varepsilon(\boldsymbol{\theta})$ represents the simulation error, which follows the normal distribution with mean zero and variance $\sigma^2(\boldsymbol{\theta})$ and captures the randomness from intrinsic stochasticity.

Let $k_B(\cdot,\cdot|\boldsymbol{\gamma}_B)$ denote the classical kernel like the RBF kernel with hyperparameters $\boldsymbol{\gamma}_B$. In DKL, we transform the input $\boldsymbol{\theta}$ by $g(\boldsymbol{\theta},\boldsymbol{\gamma}_N)$, which is a deep neural network with parameters $\boldsymbol{\gamma}_N$. The covariance of the mean response $\mathscr{J}(\boldsymbol{\theta})$ and $\mathscr{J}(\boldsymbol{\theta}')$ becomes

$$k_D(\boldsymbol{\theta},\boldsymbol{\theta}'|\boldsymbol{\gamma}_N,\boldsymbol{\gamma}_B) = k_B(g(\boldsymbol{\theta},\boldsymbol{\gamma}_N), g(\boldsymbol{\theta}',\boldsymbol{\gamma}_N)|\boldsymbol{\gamma}_N,\boldsymbol{\gamma}_B),$$

where $\boldsymbol{\gamma}_N$ and $\boldsymbol{\gamma}_B$ are the hyperparameters that need to be learned from simulation results. The deep kernel $k_D(\cdot,\cdot)$ incorporates more flexibility than the classical kernel, which can improve the predictive power of GP. It is used to learn the performance similarity of $\mathscr{J}(\boldsymbol{\theta})$ at evaluated and unevaluated policy parameters with a well-trained neural network. This means that the candidate policies have high covariance not only because of their short spatial distance, but also because of similar dynamics in the PABN model. Therefore, the modified kernel can characterize more general spatial interdependencies in the PABN model and it can be used to guide the exploration and the exploitation more effectively in the solution space.

Given k design points $\{\boldsymbol{\theta}^{(i)}\}_{i=1}^k$ and their simulated results $\{\hat{r}^{(i)}\}_{i=1}^k$, we can jointly learn the parameters $\boldsymbol{\gamma} = \{\boldsymbol{\gamma}_N, \boldsymbol{\gamma}_B\}$ by maximizing the log likelihood, which is defined as

$$LL = -\frac{1}{2}\hat{\boldsymbol{r}}^\top (\boldsymbol{K}_{\boldsymbol{\gamma}} + \boldsymbol{\Sigma}_\varepsilon)^{-1}\hat{\boldsymbol{r}} - \frac{1}{2}\log|\boldsymbol{K}_{\boldsymbol{\gamma}} + \boldsymbol{\Sigma}_\varepsilon| - \frac{k}{2}\log 2\pi,$$

where $\boldsymbol{K}_{\boldsymbol{\gamma}}$ is the covariance matrix between the mean response at the design points, $\boldsymbol{\Sigma}_\varepsilon = \mathrm{diag}\{\sigma^2(\boldsymbol{\theta}^{(1)}), \sigma^2(\boldsymbol{\theta}^{(2)}), \ldots, \sigma^2(\boldsymbol{\theta}^{(k)})\}$, and $\hat{\boldsymbol{r}} = (\hat{r}^{(1)}, \hat{r}^{(2)}, \ldots, \hat{r}^{(k)})$ is the vector of simulation outputs at design points. The derivative of the log marginal likelihood with respect to the parameter $\boldsymbol{\gamma}$ is:

$$\frac{\partial LL}{\partial \boldsymbol{\gamma}} = \frac{\partial LL}{\partial \boldsymbol{K}_{\boldsymbol{\gamma}}}\frac{\partial \boldsymbol{K}_{\boldsymbol{\gamma}}}{\partial \boldsymbol{\gamma}}, \text{ where } \frac{\partial LL}{\partial \boldsymbol{K}_{\boldsymbol{\gamma}}} = \frac{1}{2}((\boldsymbol{K}_{\boldsymbol{\gamma}} + \boldsymbol{\Sigma}_\varepsilon)^{-1}\hat{\boldsymbol{r}}\hat{\boldsymbol{r}}^\top (\boldsymbol{K}_{\boldsymbol{\gamma}} + \boldsymbol{\Sigma}_\varepsilon)^{-1} - (\boldsymbol{K}_{\boldsymbol{\gamma}} + \boldsymbol{\Sigma}_\varepsilon)^{-1}).$$

Through the chain rule of gradient calculation and gradient descent for negative log likelihood, the kernel parameters $\boldsymbol{\gamma}$ can be trained. Let $\boldsymbol{k}_{\boldsymbol{\gamma}}(\boldsymbol{\theta})$ denote the covariance vector between $\boldsymbol{\theta}$ and simulated design points, a sequential optimization procedure based on DKL is shown in Algorithm 1.

The structure of the deep kernel depends on the complexity of the PABN model. The deep kernel with a more complex architecture has more representation power and more flexibility. However, it can result in overfitting with limited evaluations and a high computational burden. In practice, we need to restrict the number of layers and hidden units. In addition, to reduce the computational time due to the training of $\boldsymbol{\gamma}$, a periodic update strategy can be adapted. We can update $\boldsymbol{\gamma}$ when several new data are collected. Though the sequential optimization algorithm with DKL can learn the spatial interdependence of the PABN from simulated data, it cannot incorporate the transition dynamics explicitly and therefore, lacks interpretability.

Algorithm 1: GP with Deep Kernel Learning

Input: PABN model, initial # of design points B_0, # of iterations B.
1. Sample B_0 design points from the design space and simulate their performance $\hat{\boldsymbol{r}}$;
2. Learn the optimal kernel parameters $\boldsymbol{\gamma}$ by maximizing the log likelihood LL;
3. **for** $b = 1, \ldots, B$ **do**
   Find the point $\boldsymbol{\theta}^*$ that maximize the acquisition function h:
   $\boldsymbol{\theta}^* = \text{argmax}_{\boldsymbol{\theta}} h(\mu(\boldsymbol{\theta}), \sigma^2(\boldsymbol{\theta}))$, where
   $\mu(\boldsymbol{\theta}) = \boldsymbol{k}_{\boldsymbol{\gamma}}(\boldsymbol{\theta})(\boldsymbol{K}_{\boldsymbol{\gamma}} + \boldsymbol{\Sigma}_\varepsilon)^{-1}\hat{\boldsymbol{r}}$ and $\sigma^2(\boldsymbol{\theta}) = k_{\boldsymbol{\gamma}}(\boldsymbol{\theta}, \boldsymbol{\theta}) - \boldsymbol{k}_{\boldsymbol{\gamma}}(\boldsymbol{\theta})(\boldsymbol{K}_{\boldsymbol{\gamma}} + \boldsymbol{\Sigma}_\varepsilon)^{-1}\boldsymbol{k}_{\boldsymbol{\gamma}}(\boldsymbol{\theta})^\top$;
   Run a simulation following the policy $\boldsymbol{\theta}^*$ and then update the design points and $\hat{\boldsymbol{r}}$;
   Update the optimal kernel parameters $\boldsymbol{\gamma}$;
4. Return the point $\boldsymbol{\theta}^*$ that maximize the posterior mean response, i.e., $\boldsymbol{\theta}^* = \text{argmax}_{\boldsymbol{\theta}} \mu(\boldsymbol{\theta})$.

4 POLICY OPTIMIZATION FOR LINEAR GAUSSIAN PABN

To overcome the limitation of the DKL-based optimization, in this section, we develop an interpretable policy optimization method and study the linear Gaussian PABN to derive analytical results. The linear state transition model assumption is valid for bioprocesses with online monitoring, i.e., monitoring frequency on a faster time scale than the evolution of bioprocess dynamics.

4.1 Method Description

In linear Gaussian PABN (Zheng et al. 2023), we assume that f_t is linear. Let $\boldsymbol{\beta}_t^s$ denote the $n \times n$ matrix whose (j,k)-th element is the linear coefficient β_t^{jk} corresponding to the effect of state s_t^j on the next state s_{t+1}^k. Similarly, let $\boldsymbol{\beta}_t^a$ be the $m \times n$ matrix of coefficients representing the effects of each component of $\boldsymbol{a}_t$ on each component of $\boldsymbol{s}_{t+1}$. Then, the stochastic process dynamics in equation (1) can be expressed as

$$\boldsymbol{s}_{t+1} = \boldsymbol{\mu}_{t+1}^s + (\boldsymbol{\beta}_t^s)^\top (\boldsymbol{s}_t - \boldsymbol{\mu}_t^s) + (\boldsymbol{\beta}_t^a)^\top (\boldsymbol{a}_t - \boldsymbol{\mu}_t^a) + \boldsymbol{e}_{t+1},$$

where $\boldsymbol{\mu}_t^s = (\mu_t^1, \ldots, \mu_t^n)$, $\boldsymbol{\mu}_t^a = (\lambda_t^1, \ldots, \lambda_t^m)$ and $\boldsymbol{e} = (\boldsymbol{e}_1^\top, \boldsymbol{e}_2^\top, \ldots, \boldsymbol{e}_H^\top) \sim \mathcal{N}(\boldsymbol{0}, \mathbf{V})$. The list of model parameters is denoted by $\boldsymbol{w} = (\boldsymbol{\mu}^s, \boldsymbol{\mu}^a, \boldsymbol{\beta}, \boldsymbol{V})$, where $\boldsymbol{\mu}^s = \{\boldsymbol{\mu}_t^s\}_{t=1}^H$, $\boldsymbol{\mu}^a = \{\boldsymbol{\mu}_t^a\}_{t=1}^{H-1}$, $\boldsymbol{\beta} = \{(\boldsymbol{\beta}^a, \boldsymbol{\beta}^s)\}_{t=1}^{H-1}$. The unknown model parameters $\boldsymbol{w}$ can be estimated from data. We also focus on linear policy and reward functions, i.e., $\boldsymbol{a}_t = \boldsymbol{\mu}_t^a + \boldsymbol{\theta}_t^\top (\boldsymbol{s}_t - \boldsymbol{\mu}_t^s)$, and $r_t(\boldsymbol{s}_t, \boldsymbol{a}_t) = m_t + \boldsymbol{b}_t^\top \boldsymbol{a}_t + \boldsymbol{c}_t^\top \boldsymbol{s}_t$, where $\boldsymbol{\theta}_t$ is an $n \times m$ coefficient matrix. The linear reward is often used in biomanufacturing; see for example Martagan et al. (2016).

Then, given model parameters $\boldsymbol{w}$, the cumulative reward $r(\boldsymbol{\theta}|\boldsymbol{w})$ becomes,

$$r(\boldsymbol{\theta}|\boldsymbol{w}) = \sum_{t=1}^H r_t(\boldsymbol{s}_t, \boldsymbol{a}_t|\boldsymbol{w}) = m + \sum_{t=1}^H \boldsymbol{\alpha}_t \mathbf{R}_{1,t-1}(\boldsymbol{s}_0 - \boldsymbol{\mu}_1^s) + \sum_{t=1}^H \boldsymbol{\alpha}_t \left(\sum_{i=1}^t \mathbf{R}_{i,t-1} \boldsymbol{e}_i \right),$$

where $\mathbf{R}_{i,t} = \prod_{j=i}^t \left[\left(\boldsymbol{\beta}_j^s\right)^\top + \left(\boldsymbol{\beta}_j^a\right)^\top \boldsymbol{\theta}_j^\top \right]$ represents the product of pathway coefficients from time step i to t and $\mathbf{R}_{i,i-1} = \mathbb{I}_{n \times n}$ is the $n \times n$ identity matrix. Let $\boldsymbol{\alpha}_t = \boldsymbol{b}_t^\top \boldsymbol{\theta}_t^\top + \boldsymbol{c}_t^\top$. Then we have $r(\boldsymbol{\theta}|\boldsymbol{w}) \sim \mathcal{N}(\text{E}[r(\boldsymbol{\theta}|\boldsymbol{w})], \text{Var}[r(\boldsymbol{\theta}|\boldsymbol{w})])$. The mean of $r(\boldsymbol{\theta}|\boldsymbol{w})$ and the covariance between $r(\boldsymbol{\theta}|\boldsymbol{w})$ and $r(\boldsymbol{\theta'}|\boldsymbol{w})$ are

$$\begin{aligned}
\text{E}[r(\boldsymbol{\theta}|\boldsymbol{w})] &= m + \sum_{t=1}^H \boldsymbol{\alpha}_t \mathbf{R}_{1,t-1}(\boldsymbol{s}_0 - \boldsymbol{\mu}_1^s), \\
\text{Cov}\left[r(\boldsymbol{\theta}|\boldsymbol{w}), r(\boldsymbol{\theta'}|\boldsymbol{w})\right] &= \text{Cov}\left[\sum_{t=1}^H \boldsymbol{\alpha}_t \left(\sum_{i=1}^t \boldsymbol{R}_{i,t-1} \boldsymbol{e}_i \right), \sum_{t=1}^H \boldsymbol{\alpha}_t' \left(\sum_{i=1}^t \boldsymbol{R}_{i,t-1}' \boldsymbol{e}_i' \right) \right] \\
&= \sum_{i=1}^H \sum_{j=1}^H \left[\boldsymbol{R}_i \text{Cov}(\boldsymbol{e}_i, \boldsymbol{e}_j') \boldsymbol{R}_j'^\top \right] = \boldsymbol{R} \mathbf{V} \boldsymbol{R}'^\top,
\end{aligned} \tag{3}$$

Algorithm 2: Pathway Correlation based Optimization for Linear Gaussian PABN

Input: initial # of design points B_0, # of iterations B, # of model parameters L.
1. Sample L model parameters $\{\boldsymbol{w}_\ell\}_{\ell=1}^{L}$;
2. Sample B_0 design points from the design space and simulate their performance with $\{\boldsymbol{w}_\ell\}_{\ell=1}^{L}$;
3. **for** $b = 1,\ldots,B$ **do**
   Find the point $\boldsymbol{\theta}^{(0)*} = \text{argmax}_{\boldsymbol{\theta}^{(0)}} \bar{\bar{\mu}}^{(0)}$, where $\bar{\mu}^{(0)}(\boldsymbol{w}_\ell)$ is calculated using Algorithm 3;
   Simulate the performance of $\boldsymbol{\theta}^{(0)*}$ and update the design points and their performance;
4. Return the point $\boldsymbol{\theta}^{(0)*} = \text{argmax}_{\boldsymbol{\theta}^{(0)}} \bar{\bar{\mu}}^{(0)}$.

where $\boldsymbol{R}_i = \sum_{t=i}^{H} \boldsymbol{\alpha}_t \mathbf{R}_{i,t-1}$ is a $1 \times n$ matrix representing the overall pathway coefficient from time step i to the cumulative reward, and $\boldsymbol{R} = (\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_H)$ Then the reward becomes $r(\boldsymbol{\theta}|\boldsymbol{w}) = m + \boldsymbol{R}_1(\boldsymbol{s}_0 - \boldsymbol{\mu}_1^s) + \boldsymbol{R}\boldsymbol{e}$.

Suppose we need to assess the expected reward of a candidate policy specified by $\boldsymbol{\theta}^{(0)}$. Given k design points $\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots, \boldsymbol{\theta}^{(k)}$ and model parameters $\boldsymbol{w}$, let $r^{(i)}$ denote the cumulative reward of $\boldsymbol{\theta}^{(i)}$, we have

$$\boldsymbol{r} = \left(r^{(0)}, r^{(1)}, \ldots, r^{(k)}\right)^\top = m + \left(\boldsymbol{R}_1^{(0)}, \boldsymbol{R}_1^{(1)}, \ldots, \boldsymbol{R}_1^{(k)}\right)^\top (\boldsymbol{s}_0 - \boldsymbol{\mu}_1^s) + \left(\boldsymbol{R}^{(0)}, \boldsymbol{R}^{(1)}, \ldots, \boldsymbol{R}^{(k)}\right)^\top \boldsymbol{e}.$$

As $\boldsymbol{e} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{V})$, we have $(r^{(0)}, r^{(1)}, \ldots, r^{(k)})^\top \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\mu_i = m + \boldsymbol{R}_1^{(i)}(\boldsymbol{s}_0 - \boldsymbol{\mu}_1^s)$ is the ith component of $\boldsymbol{\mu}$ and $\sigma_{ij} = \boldsymbol{R}^{(i)} \boldsymbol{V} \boldsymbol{R}^{(j)\top}$ is the component at the ith row and jth column of $\boldsymbol{\Sigma}$. If we partition the $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ as $\boldsymbol{\mu} = \begin{pmatrix} \mu_0 \\ \boldsymbol{\mu}_1 \end{pmatrix}, \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{00} & \boldsymbol{\Sigma}_{01} \\ \boldsymbol{\Sigma}_{10} & \boldsymbol{\Sigma}_{11} \end{pmatrix}$, given k design points and their evaluations $\hat{\boldsymbol{r}} = (\hat{r}^{(1)}, \ldots \hat{r}^{(k)})$ under $\boldsymbol{w}$, where $\hat{r}^{(i)} = m + \boldsymbol{R}_1(\boldsymbol{s}_0 - \boldsymbol{\mu}_1^s) + \boldsymbol{R}\boldsymbol{e}^{(i)}$, the predictive distribution of $r^{(0)}$ is $\mathcal{N}(\bar{\mu}^{(0)}(\boldsymbol{w}), \bar{\sigma}^{(0)}(\boldsymbol{w}))$, where

$$\bar{\mu}^{(0)}(\boldsymbol{w}) = \mu_0 + \boldsymbol{\Sigma}_{01}\boldsymbol{\Sigma}_{11}^{-1}(\hat{\boldsymbol{r}} - \boldsymbol{\mu}_1) \quad \text{and} \quad \bar{\sigma}^{(0)}(\boldsymbol{w}) = \sigma_{00} - \boldsymbol{\Sigma}_{01}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{10}. \tag{4}$$

Similar to the Bayesian optimization, equation (4) provides the predictive mean for the performance of any given policy $\boldsymbol{\theta}$. However, it incorporates the spatial interdependence with the analytical covariance, i.e., equation (3), which represents the pathway similarity between any two candidate policies. As a result, equation (4) serves as a more interpretable predictor compared to the predictive mean in Bayesian optimization. It can also contribute to the interpretation of deep kernel learning as shown in Section 5. Although equation (3) can be optimized directly, it does not offer the same level of insight into the learning process of the deep kernel.

Now suppose that for each design point $\boldsymbol{\theta}^{(i)}$, we have evaluations for $r^{(i)}$. We can calculate $\bar{\mu}^{(0)}(\boldsymbol{w}_\ell)$ for $\ell = 1, 2, \ldots, L$, respectively. The sample mean estimation of expected cumulative reward $\mathscr{J}\left(\boldsymbol{\theta}^{(0)}\right)$ is $\bar{\bar{\mu}}^{(0)} = \frac{1}{L}\sum_{\ell=1}^{L} \bar{\mu}^{(0)}(\boldsymbol{w}_\ell)$, where $\boldsymbol{w}_\ell \sim p(\boldsymbol{w}|\mathscr{D})$ for $\ell = 1, 2, \ldots, L$, accounting for model uncertainty. The next point to be evaluated $\boldsymbol{\theta}^{(0)*}$ is the feasible point with the largest $\bar{\bar{\mu}}^{(0)}$, i.e.

$$\boldsymbol{\theta}^{(0)*} = \text{argmax}_{\boldsymbol{\theta}^{(0)}} \bar{\bar{\mu}}^{(0)}. \tag{5}$$

Then we can run additional L simulations on $\boldsymbol{\theta}^{(0)*}$ with different model parameters and repeat the procedure (4) to (5) until the total simulation budget is exhausted. The complete process is shown in Algorithm 2.

4.2 Algorithm for Predictive Mean Calculation

In this section, we provide an efficient algorithm to calculate the predictive mean in equation (4) in Algorithm 3. For a given point $\boldsymbol{\theta}^{(0)}$ to be predicted, we firstly need to calculate all the pathway coefficients $\mathbf{R}_{t_1,t_2}^{(0)}$ to obtain $\mathbf{R}^{(0)}$. As $\mathbf{R}_{t_1,t_2}^{(0)}$ can be calculated by reusing the calculation of $\mathbf{R}_{t_1+1,t_2}^{(0)}$, we can recursively calculate the $\mathbf{R}_{t_1,t_2}^{(0)}$, which reduces the computational cost by a factor of $\mathrm{O}(H)$ in step 1 compared to the brute force algorithm without reusing the calculation. Next, we need to calculate the updated $\boldsymbol{\Sigma}$ between

Algorithm 3: Calculate the Predictive Mean Given Model Parameters $\boldsymbol{w}$

Input: point to be predicted $\boldsymbol{\theta}^{(0)}$, the model parameters of the linear Gaussian PABN model $\boldsymbol{w} = \left\{\boldsymbol{\beta}^s_{1:H-1}, \boldsymbol{\beta}^a_{1:H-1}, \boldsymbol{\mu}^s_{1:H}, \boldsymbol{\mu}^a_{1:H-1}, \boldsymbol{V}\right\}$, existing design points $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(k)}$ and their evaluations $\hat{\boldsymbol{r}}$ under $\boldsymbol{w}$, the pathway coefficients of jth design point $\boldsymbol{R}^{(j)}$, covariance matrix between design points $\boldsymbol{\Sigma}$, the rewards coefficients $\boldsymbol{b}_{1:H}$ and $\boldsymbol{c}_{1:H}$, $\mathbf{R}^{(0)}_{i+1,i} = I_{n\times n}$ for $i = 1, 2, \ldots, H$.

1. Calculate all the pathway coefficients to the cumulative reward at $\boldsymbol{\theta}^{(0)}$:

for $t_2 = 1, 2, \ldots, H$ **do**
  for $t_1 = t_2, \ldots, 1$ **do**
    $\mathbf{R}^{(0)}_{t_1,t_2} = (\boldsymbol{\beta}^s_{t_1} + \boldsymbol{\theta}^{(0)}_{t_1}\boldsymbol{\beta}^a_{t_1})^\top \mathbf{R}^{(0)}_{t_1+1,t_2}$;

for $i = 1, \ldots, H$ **do**
  $\boldsymbol{R}^{(0)}_i = 0$;
  for $t = i, \ldots, H$ **do**
    $\boldsymbol{R}^{(0)}_i = \boldsymbol{R}^{(0)}_i + (\boldsymbol{\theta}^{(0)}_t \boldsymbol{b}_t + \boldsymbol{c}_t)^\top \boldsymbol{R}^{(0)}_{i,t-1}$;

2. Calculate the updated covariance matrices: $\boldsymbol{R}^{(0)} = (\boldsymbol{R}^{(0)}_1, \boldsymbol{R}^{(0)}_2, \ldots, \boldsymbol{R}^{(0)}_H)$

$\boldsymbol{\sigma}_{0j} = \boldsymbol{R}^{(0)} \boldsymbol{V} \boldsymbol{R}^{(j)}$ for $j = 0, 1, \ldots, k$, $\Sigma_{01} = (\sigma_{01}, \sigma_{02}, \ldots \sigma_{0k})$, $\boldsymbol{\Sigma}_{11} = \boldsymbol{\Sigma}$, $\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{00} & \boldsymbol{\Sigma}_{01} \\ \boldsymbol{\Sigma}_{10} & \boldsymbol{\Sigma}_{11} \end{pmatrix}$

3. Update the inverse of the covariance matrices:

$v_{k+1} = \sigma_{00} - \boldsymbol{\Sigma}_{01}\boldsymbol{\Sigma}^{-1}_{11}\boldsymbol{\Sigma}_{10}$, $g_{k+1} = -v^{-1}_{k+1}\boldsymbol{\Sigma}^{-1}_{11}\boldsymbol{\Sigma}_{10}$, $\boldsymbol{\Sigma}^{-1} = \begin{pmatrix} \boldsymbol{\Sigma}^{-1}_{11} + g_{k+1}g^\top_{k+1}v_{k+1} & g_{k+1} \\ g^\top_{k+1} & v^{-1}_{k+1} \end{pmatrix}$;

4. Calculate the predictive mean at $\boldsymbol{\theta}^{(0)}$: $\bar{\mu}^{(0)}(\boldsymbol{w}) = \mu_0 + \boldsymbol{\Sigma}_{01}\boldsymbol{\Sigma}^{-1}_{11}(\hat{\boldsymbol{r}} - \boldsymbol{\mu}_1)$

$\boldsymbol{\theta}^{(0)}$ and the design points. As the pathway coefficient $\mathbf{R}^{(i)}$ of the evaluated point i has already been calculated during the previous update, we can reuse the results. Therefore, in step 2, we can reduce the computational cost of calculating $\mathbf{R}^{(i)}$. In step 3, for the inverse of the current covariance matrix $\boldsymbol{\Sigma}$, we have $\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{10} \\ \boldsymbol{\Sigma}_{01} & \sigma_{00} \end{pmatrix}$, $\boldsymbol{\Sigma}^{-1} = \begin{pmatrix} \boldsymbol{\Sigma}^{-1}_{11} + g_{k+1}g^\top_{k+1}v_{k+1} & g_{k+1} \\ g^\top_{k+1} & v^{-1}_{k+1} \end{pmatrix}$, where $g_{k+1} = -v^{-1}_{k+1}\boldsymbol{\Sigma}^{-1}_{11}\boldsymbol{\Sigma}_{10}$ and $v_{k+1} = \sigma_{00} - \boldsymbol{\Sigma}_{01}\boldsymbol{\Sigma}^{-1}_{11}\boldsymbol{\Sigma}_{10}$. Compared to the direct inverse which costs $\mathrm{O}(k^3)$, the computational cost now becomes $\mathrm{O}(k^2)$, which can save $\mathrm{O}(k)$ cost. The computational cost is $\mathrm{O}(H^2n^2(m+n))$, $\mathrm{O}(kH^2n^2)$, and $\mathrm{O}\left(k^2\right)$ in steps 1 to 3, respectively. Therefore, the overall computational complexity is $\mathrm{O}(H^2n^2(m+n)) + \mathrm{O}\left(k^2\right)$.

Proposition 1 The cost to compute the predictive mean $\bar{\mu}^{(0)}$ is $\mathrm{O}(H^2n^2(m+n)) + \mathrm{O}\left(k^2\right)$ for Algorithm 3 and $\mathrm{O}(kH^3n^2(m+n)) + \mathrm{O}\left(k^3\right)$ for brute force algorithm without reusing.

The storing and reusing method shares the similar idea with the message passing, which utilizes the graph structure to achieve fast computation. The major difference is that the message passing is used for the marginalization and inference of the probability distribution in the general graphical model, however, the storing and reusing method is used for the computation of the covariance for the linear Gaussian PABN.

5 EMPIRICAL STUDY

For the empirical results, we first present three experiments for the ablation analysis and the interpretation analysis of DKL in Section 5.1. Basically, Experiment 1 presents the impact of the number of layers on the performance of the DKL. Experiment 2 shows that the covariance learned by the DKL can capture the performance similarity between different candidate policies. Experiment 3 further illustrates that the DKL can potentially learn the pathway similarity of the PABN model. Then in Section 5.2, we compare the performance of the DKL-based method and the pathway correlation based method with the standard BO in optimizing general PABN and linear Gaussian PABN, respectively.

For the general PABN model in this study, the transition function is set to be $f(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{w}_t) = \boldsymbol{\beta}_t^s(\boldsymbol{s}_t + \boldsymbol{\alpha}_t^s \sin(\boldsymbol{s}_t)) + \boldsymbol{\beta}_t^a(\boldsymbol{a}_t + \boldsymbol{\alpha}_t^a \sin(\boldsymbol{a}_t)) + \boldsymbol{m}_t$, where the model parameters $\boldsymbol{w}_t = \{\boldsymbol{\beta}_t^s, \boldsymbol{\alpha}_t^s, \boldsymbol{\beta}_t^a, \boldsymbol{\alpha}_t^a, \boldsymbol{m}_t\}$. It involves both the monotonicity term as well as the nonlinearity term. The policy and reward functions are set to be linear. The dimension of actions $m = 1$, the dimension of states $n = 3$, and the time horizon $H = 3$, which leads to a 6-dimensional optimization problem. The feasible region of the policy parameters is chosen by constraining each individual policy parameter to an interval. For the linear PABN model, the transition function is linear, and the problem size is also set to be 6 dimensions with $n = 3$, $m = 1$ and $H = 3$.

5.1 Ablation Study and Interpretation Analysis

We first investigate the impact of the number of layers in the neural network of the deep kernel for the general PABN in Experiment 1. In specific, we simulate the performance of 100 candidate policies as the training set, which are selected by Latin Hypercube Sampling (LHS). Additionally, we simulate the performance of another 20 policy parameters as the test set, which are also chosen by LHS. The Gaussian processes with different deep kernels of varying numbers of layers are used to fit the training set. Then they are evaluated in terms of the mean squared error (MSE) on the test set. For the neural network in the deep kernel, we fix the number of nodes in the input layer, the hidden layer, and the output layer to be 6, 50, and 2. All the activation functions are set to be ReLU. The number of layers is set to 0 (classical kernel), 1 (shallow kernel), 4 (deep kernel), and 20 (very deep kernel), respectively.

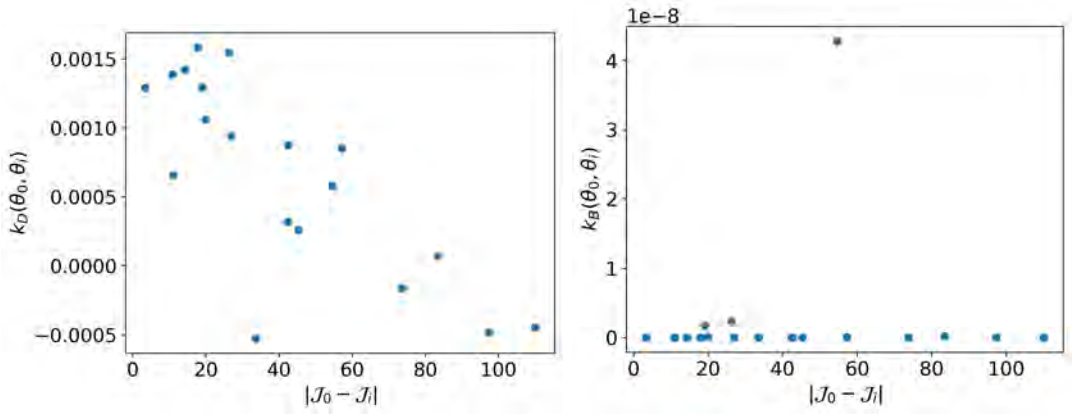
Table 1 shows the MSEs with 95% confidence intervals (CI) of deep kernels with different numbers of layers. The results are estimated based on 10 macro-replications. We can see the MSE first decreases as the number of layers increases and then increases when the number of layers becomes 20 (i.e., very deep kernel). As the number of layers increases, the deep kernel provides more flexibility and therefore improves the fitting power of the Gaussian process metamodel. However, when the number of layers becomes excessively high, the Gaussian process may overfit the noisy simulation outputs, especially during the early stages or iterations of Bayesian optimization.

Table 1: MSE with 95% CI of the deep kernels with different number of layers.

Number of Layers	MSE	Half Length of CI
0	3392.79	1245.68
1	2892.81	977.90
4	827.55	279.87
20	1410.99	274.23

In Experiment 2, we aim to show that the deep kernel can learn more complex information about spatial interdependence than a pure classical kernel. Notice that the classical kernel measures the closeness or similarity of the policy parameters in the original space and assumes that the policy parameters close to each other are likely to exhibit similar performance. However, the deep kernel learns the embedding of the policy parameters, which maps the policy parameters into the new space and measures the similarity of the policy parameters in the new space. Therefore, it incorporates not only the closeness of the policy parameters but also the similarity of the resulting graphical structure of PABN. In Experiment 2, we choose the RBF kernel as the classical kernel, and the architecture of the neural network in DKL is set as a 6-layer fully connected neural network with (6, 50, 50, 50, 50, 2) nodes. The design points used to train the Gaussian process metamodel with the deep kernel and the classical kernel are selected by LHS, and the number of points is set to 1000. Subsequently, we sample 20 test points by LHS and calculate the covariance and the absolute difference of the true objective function estimated by side experiments.

One representative result in Experiment 2 is shown in Figure 2. The x-axis is the absolute difference of the true objective function between a fixed test point $\boldsymbol{\theta}_0$ and each remaining test point $\boldsymbol{\theta}_i$. Meanwhile, the y-axis represents the covariance between the performance of $\boldsymbol{\theta}_0$ and the performance of each remaining test point $\boldsymbol{\theta}_i$, obtained from the deep kernel and the RBF kernel, respectively. It is clear that the covariance



(a) Covariance of deep kernel (b) Covariance of RBF kernel

Figure 2: Covariance obtained from two kernels against the absolute difference of the true objective value.

Table 2: PCC between the learned covariance and the absolute difference of true objective value.

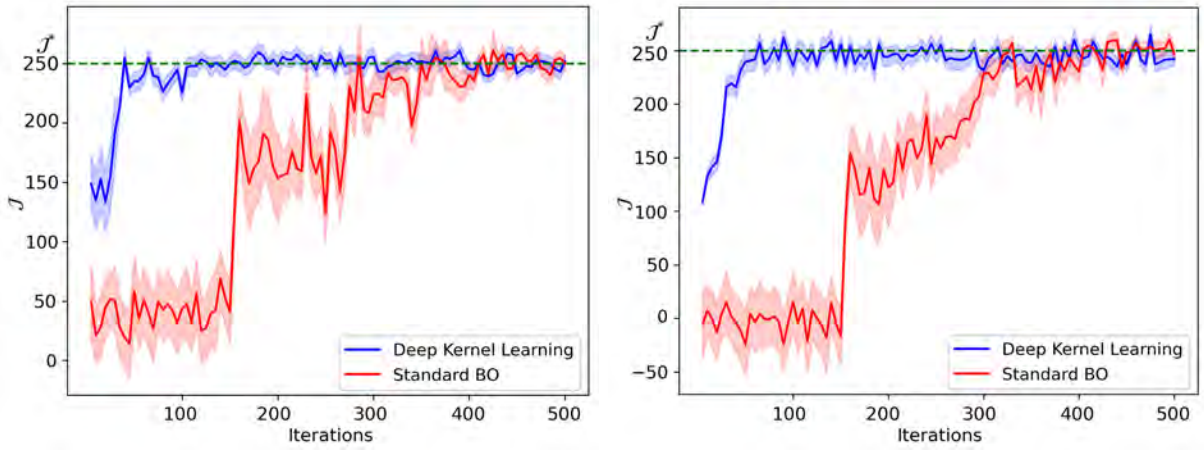
	Pearson Correlation Coefficient	P-value
Deep Kernel	-0.41	$7.58e-18$
RBF Kernel	-0.04	0.34

obtained from the deep kernel is negatively correlated with the absolute difference of the objective function, i.e., two input points with similar performances have high covariance, indicating that the deep kernel effectively measures the similarity between the performances of test points. However, this pattern cannot be observed from the covariance obtained using the RBF kernel since the RBF kernel cannot learn the similarity except distance. Notably, we can see that the covariance obtained from the RBF kernel is significantly smaller than the covariance obtained from the deep kernel since the test points generated by LHS are uniformly spread throughout the policy parameter space, resulting in considerable distance between them. We also calculate the Pearson correlation coefficients (PCC) (McNamara et al. 2014) between the covariance of the performances of test points and the absolute difference of the true objective function for both the deep kernel and the RBF kernel in Table 2, which validates the above conclusion.

To further interpret what can be learned by deep kernel learning, we consider the Gaussian process metamodel with deep kernel and RBF kernel for the linear PABN model. The construction of the training set and test set follows the procedure outlined in Experiment 2. The architecture of the neural network in DKL is set to be a 6-layer fully connected neural network with (6, 50, 50, 50, 50, 2) nodes. We calculate the learned covariance between the performances of the test policy parameters obtained from the deep kernel and the RBF kernel, respectively. Then we calculate the Pearson correlation coefficient between the learned covariance and the true covariance, which is estimated by equation (3). The results of Experiment 3 are presented in Table 3. It is evident that the covariance function learned by DKL exhibits a significant positive linear correlation with the true covariance of the linear PABN. The true covariance of linear PABN, which captures the pathway similarity in the graphical model. Therefore, DKL demonstrates the ability to learn the similarity of the resulting graphical structure of the PABN for the Gaussian process metamodel, i.e., the spatial interdependence of the performance over the feasible space of policy parameters. As a

Table 3: PCC between the learned covariance and the estimated true covariance of the linear PABN.

	Pearson Correlation Coefficient	P-value
Deep Kernel	0.32	$2.42e-10$
RBF Kernel	0.05	0.35



(a) Upper confidence bound

(b) Expected improvement

Figure 3: The convergence behaviors of the mean objective and 95% CI for general PABN.

result, the deep kernel can benefit both the prediction of the mean response surface and the exploration of the unevaluated policy parameters.

5.2 Optimization

In this section, we present the performance of the DKL-based method (i.e., Algorithm 1) in optimizing general PABN as well as the performance of the pathway correlation based method (i.e., Algorithm 2) in optimizing linear Gaussian PABN. We compare both methods with the standard BO method in Wang et al. (2020) in terms of empirical convergence rate and running time.

We first compare the DKL-based method with the standard BO method in optimizing the general PABN model. For DKL based optimization, we apply the RBF kernel as the classical kernel $k_B(\cdot,\cdot)$. The architecture of the neural network is fully connected with (6, 50, 50, 50, 50, 2) nodes. The activation function is the ReLU function. For standard BO, the kernel is also set to be an RBF kernel. The acquisition functions used in both methods are upper confidence bound (UCB) and expected improvement (EI). The initial points are selected with LHS, and the number is set to be 30.

In Figure 3, we demonstrate the true objective value obtained at each iteration for both the DKL-based optimization method and the standard BO. We plot the mean performance of both methods by solid lines and the 95% confidence intervals (CI) by shaded areas, which are estimated over 10 macro-replications. The actual optimal objective function value J^* is estimated by randomly sampling policy parameters from the feasible space with a sample size of 1×10^6. We can see that the DKL based optimization method converges to the global optimum within 100 iterations, while the standard BO method converges after around 300 iterations. It can also be observed that DKL-based optimization exhibits robust performance across different acquisition functions. The running times after iterations 50, 100, and 200 for both methods are recorded in Table 4. The DKL-based optimization is more time-consuming than standard BO as it involves the training of a neural network. However, as the optimization progresses, the time difference due to training the neural network becomes less significant compared to the overall running time.

Next, we compare the pathway correlation based method proposed in Section 4 with standard BO for linear Gaussian PABN optimization. In the pathway correlation based method, to optimize $\bar{\mu}$, we use

Table 4: Running times of the DKL-based optimization and the standard BO for general PABN.

	DKL-UCB	Standard BO-UCB	DKL-EI	Standard BO-EI
iter = 50	12.47s	5.66s	15.57s	6.78s
iter = 100	37.83s	18.06s	41.16s	19.98s
iter = 200	78.19s	56.19s	80.32s	57.43s

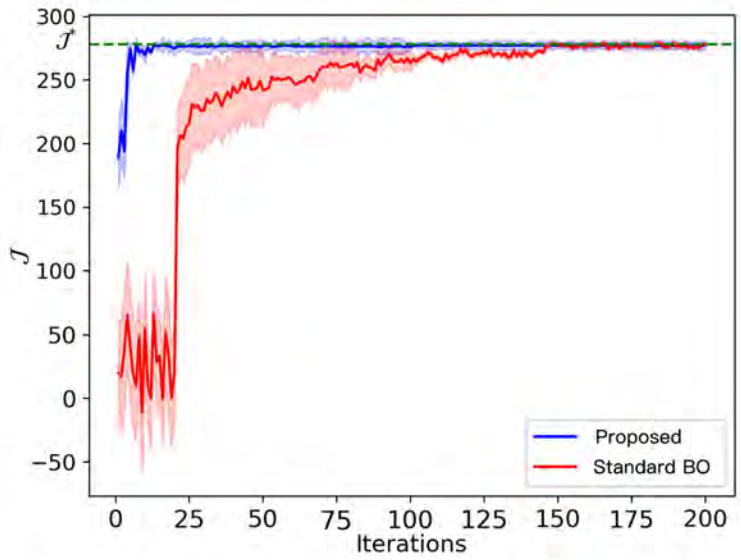


Figure 4: Mean objective value obtained with 95% CI at each iteration for linear Gaussian PABN.

Table 5: Running times of proposed, brute force and standard BO algorithm for linear Gaussian PABN.

	Proposed	Brute Force	Standard BO
iter = 10	2.84s	84.93s	0.40s
iter = 50	16.12s	417.87s	4.33s
iter = 100	36.07s	1038.73s	17.79s

a quasi-Newton method, the limited-memory Broyden–Fletcher–Goldfarb–Shanno for bound-constrained problems (L-BFGS-B) algorithm with multiple start points. We first compare the speed of convergence for both methods. The mean performance as well as the 95% confidence intervals are shown in Figure 4, which are estimated from 10 macro-replications. The actual optimal objective function value J^* is estimated by randomly sampling policy parameters from the feasible space with a sample size of 1×10^6. We can see that the proposed pathway correlation based method attains the optimal objective value with less than 20 iterations. However, the standard BO converges to the optimal objective value after 100 iterations, which is more sample inefficient than our method. Therefore, we can leverage the structural information and the transition pathway of stochasticity in PABN to improve the optimization. We also compare the time efficiency of the proposed pathway correlation based method with the brute force implementation, which does not adapt the storing and reusing strategy, and the standard BO. The running times averaged over 10 macro replications after various iterations are recorded in Table 5. Compared to the brute force implementation, our method can be more efficient with the reuse of previous calculations. Compared to the standard BO, our method is relatively slower, but as the iterations increase, the gap becomes less significant.

6 CONCLUSION

In this paper, we propose the global optimization methods for the PABN hybrid model in biomanufacturing. For general PABN, a DKL-based sequential optimization algorithm is developed, which can learn the spatial interdependence of PABN. Empirical experiments provide the ablation study and interpretation analysis of DKL. The results show that the DKL-based method works better than the standard BO method. To achieve interpretable optimization, we utilize the pathway correlation information explicitly in linear Gaussian PABN optimization, which empirically converges faster. In the future research, we will consider to design deep kernels and acquisition functions tailored to the policy optimization of PABN models that involve highly heterogeneous noise. Additionally, we will develop an interpretable optimization method for PABN with general transition functions.

REFERENCES

Astudillo, R., and P. Frazier. 2021. "Bayesian Optimization of Function Networks". *Advances in Neural Information Processing Systems* 34:14463–14475.

Bowden, J., J. Song, Y. Chen, Y. Yue, and T. Desautels. 2021. "Deep Kernel Bayesian Optimization". Technical Report No. LLNL-CONF-819001, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States).

Chau, M., M. C. Fu, H. Qu, and I. O. Ryzhov. 2014. "Simulation Optimization: A Tutorial Overview and Recent Developments in Gradient-Based Methods". In *Proceedings of the Winter Simulation Conference 2014*, edited by A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. J. Buckley, and J. A. Miller, 21–35. Institute of Electrical and Electronics Engineers, Inc.

Cheng, Z., J. Luo, and R. Wu. 2023. "On the Finite-Sample Statistical Validity of Adaptive Fully Sequential Procedures". *European Journal of Operational Research* 307(1):266–278.

Del Rio-Chanona, E. A., X. Cong, E. Bradford, D. Zhang, and K. Jing. 2019. "Review of Advanced Physical and Data-Driven Models for Dynamic Bioprocess Simulation: Case Study of Algae–Bacteria Consortium Wastewater Treatment". *Biotechnology and Bioengineering* 116(2):342–353.

Frazier, P. I. 2018. "Bayesian Optimization". In *Recent Advances in Optimization and Modeling of Contemporary Problems*, 255–278. INFORMS.

Gottschalk, U., K. Brorson, and A. A. Shukla. 2012. "The Need for Innovation in Biomanufacturing". *Nature Biotechnology* 30(6):489–492.

Hong, M. S., K. A. Severson, M. Jiang, A. E. Lu, J. C. Love, and R. D. Braatz. 2018. "Challenges and Opportunities in Biopharmaceutical Manufacturing Control". *Computers & Chemical Engineering* 110:106–114.

Kasemiire, A., H. T. Avohou, C. De Bleye, P.-Y. Sacre, E. Dumont, P. Hubert, and E. Ziemons. 2021. "Design of Experiments and Design Space Approaches in the Pharmaceutical Bioprocess Optimization". *European Journal of Pharmaceutics and Biopharmaceutics* 166:144–154.

Koller, D., and N. Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Luo, Y., V. Kurian, and B. A. Ogunnaike. 2021. "Bioprocess Systems Analysis, Modeling, Estimation, and Control". *Current Opinion in Chemical Engineering* 33:100705.

Martagan, T., A. Krishnamurthy, and C. T. Maravelias. 2016. "Optimal Condition-Based Harvesting Policies for Biomanufacturing Operations with Failure Risks". *IIE Transactions* 48(5):440–461.

McNamara, C. G., Á. Tejero-Cantero, S. Trouche, N. Campo-Urriza, and D. Dupret. 2014. "Dopaminergic Neurons Promote Hippocampal Reactivation and Spatial Memory Persistence". *Nature Neuroscience* 17(12):1658–1660.

Park, S.-Y., C.-H. Park, D.-H. Choi, J. K. Hong, and D.-Y. Lee. 2021. "Bioprocess Digital Twins of Mammalian Cell Culture for Advanced Biomanufacturing". *Current Opinion in Chemical Engineering* 33:100702.

Sun, L., L. J. Hong, and Z. Hu. 2014. "Balancing Exploitation and Exploration in Discrete Optimization via Simulation through a Gaussian Process-Based Search". *Operations Research* 62(6):1416–1438.

Wang, H., S. H. Ng, and X. Zhang. 2020. "A Gaussian Process Based Algorithm for Stochastic Simulation Optimization with Input Distribution Uncertainty". In *Proceedings of the Winter Simulation Conference 2020*, edited by K.-H. G. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, 2899–2910. Institute of Electrical and Electronics Engineers, Inc.

Wilson, A. G., Z. Hu, R. Salakhutdinov, and E. P. Xing. 2016. "Deep Kernel Learning". In *Artificial Intelligence and Statistics*, 370–378. PMLR.

Yi, S., and M. Zorzi. 2021. "Robust Kalman Filtering under Model Uncertainty: The Case of Degenerate Densities". *IEEE Transactions on Automatic Control* 67(7):3458–3471.

Zheng, H., W. Xie, I. O. Ryzhov, and D. Xie. 2023. "Policy Optimization in Dynamic Bayesian Network Hybrid Models of Biomanufacturing Processes". *INFORMS Journal on Computing* 35(1):66–82.

AUTHOR BIOGRAPHIES

JUNKAI ZHAO is a Ph.D. student in the Antai College of Economics and Management at Shanghai Jiao Tong University. His research interest is simulation optimization. His email address is zhaojunkai@sjtu.edu.cn.

WEI XIE is an assistant professor in Mechanical and Industrial Engineering at Northeastern University. Her research interests include AI/ML, computer simulation, data analytics, and stochastic optimization. Her email address is w.xie@northeastern.edu.

JUN LUO is a professor in the Antai College of Economics and Management at Shanghai Jiao Tong University. His primary research interests are simulation optimization and statistical learning. His email address is jluo_ms@sjtu.edu.cn.