Computers and Chemical Engineering 186 (2024) 108686

Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

Taking the human out of decomposition-based optimization via artificial intelligence, Part II: Learning to initialize

Ilias Mitrai, Prodromos Daoutidis *

Department of Chemical Engineering and Materials Science, University of Minnesota, Minneapolis, MN 55455, United States of America

ARTICLE INFO

Keywords:
Algorithm configuration
Decomposition based solution algorithm
Machine learning
Active learning
Supervised learning
Mixed-integer MPC

ABSTRACT

The repeated solution of large-scale optimization problems arises frequently in process systems engineering tasks. Decomposition-based solution methods have been widely used to reduce the corresponding computational time, yet their implementation has multiple steps that are difficult to configure. We propose a machine learning approach to learn the optimal initialization of such algorithms which minimizes the computational time. Active and supervised learning is used to learn a surrogate model that predicts the computational performance for a given initialization. We apply this approach to the initialization of Generalized Benders Decomposition for the solution of mixed-integer model predictive control problems. The surrogate models are used to find the optimal initialization, which corresponds to the number of initial cuts that should be added in the master problem. The results show that the proposed approach can lead to a significant reduction in solution time, and active learning can reduce the data required for learning.

1. Introduction

Decomposition-based optimization algorithms have been used widely to solve complex and large-scale optimization problems in a broad range of applications in chemical engineering, such as production scheduling and planning, supply chain management, mixed-integer optimal control, and real-time operation. These algorithms exploit the underlying structure of a problem and decompose it into a number of easier-to-solve subproblems. Typical examples include Benders (Benders, 1962) and Generalized Benders Decomposition (Geoffrion, 1972), Lagrangian decomposition (Guignard and Kim, 1987), Alternating Direction Method of Multipliers (ADMM) (Boyd et al., 2011), and cross decomposition (Van Roy, 1983).

Despite the wide success of these algorithms, their efficiency over monolithic methods is not known a-priory and their implementation, especially in an online setting, is challenging due to the many steps involved and the underlying computational complexity of every step. Automatically determining when to use and how to implement a decomposition-based solution algorithm can simplify their implementation and reduce the solution time for complex optimization problems. In the companion paper (Mitrai and Daoutidis, 2023b) we proposed a graph classification approach to determine when to use a decomposition-based solution algorithm. In this paper, we focus on the implementation of decomposition-based solution methods.

The application of a decomposition-based solution algorithm has three steps: (1) problem decomposition, (2) coordination scheme, and (3) initialization of the algorithm. In the first step, the original problem is decomposed into a number of easier-to-solve subproblems. This step requires knowledge of the underlying structure of the problem. Automatic decomposition approaches either represent the problem as a graph and employ methods from network science to learn the underlying structure (Allman et al., 2019; Mitrai and Daoutidis, 2020; Mitrai et al., 2022; Mitrai and Daoutidis, 2021; Jalving et al., 2018) or use machine learning (ML) to analyze the computational efficiency of different decompositions (Basso et al., 2020; Basso and Ceselli, 2023). The coordination step determines the exchange of information between the different subproblems. For distributed algorithms, the coordination determines the update of the dual variables whereas for hierarchical algorithms it involves the addition of cuts. Finally, the last step is the initialization of the algorithm. Unlike monolithic algorithms which may require an initial guess for the variables, decomposition-based algorithms require additional information regarding the values of the dual variables for distributed algorithms or an initial set of cuts for hierarchical algorithms. The configuration of these steps can have an important effect on the computational performance of a decomposition-based solution algorithm, however, determining the best configuration for a given problem is nontrivial.

These steps can be viewed as hyperparameters of the solution algorithm. Therefore, given an optimization problem and a decomposition-based solution algorithm, one must find the optimal values of the algorithm parameters such that a desired performance function, e.g., the

* Corresponding author.
E-mail address: daout001@umn.edu (P. Daoutidis).

https://doi.org/10.1016/j.compchemeng.2024.108686
Received 1 October 2023; Received in revised form 22 March 2024; Accepted 9 April 2024
Available online 12 April 2024

solution time, is optimized. Formally this is known as the algorithm configuration problem and is stated as follows (Eggensperger et al., 2019; Schede et al., 2022):

Problem 1 (*Algorithm Configuration*)**.** Given an optimization problem P, an optimization algorithm α with parameters $n \in \mathcal{N}$, and a performance function $m : \mathcal{P} \times \Pi \mapsto \mathcal{M}$, determine the optimal values of the parameters n^* that optimize m

$$n^* \in \arg\min_{n \in \mathcal{N}} \; m(n, P). \tag{1}$$

The algorithm configuration problem has three components, the optimization problem P which belongs in some class of optimization problems $\mathcal{P}$, the parameter space, i.e., all possible values of the parameters represented by $\mathcal{N}$, and the performance space $\mathcal{M}$ which is a metric to compare the different configurations. We note that the version of the problem presented above is known as the per-instance algorithm configuration problem since the optimal parameters are determined only for a given problem P. Alternatively, one can find the optimal values of the parameters for a class or set of optimization problems. The algorithm configuration can be either static, i.e., the parameters of the algorithms remain fixed during the solution process, or dynamic, where the parameters adapt as the solution procedure evolves. The solution of the algorithm configuration problem is challenging since optimization solvers, either monolithic or decomposition-based, have multiple algorithmic steps and each step can have different parameters. Furthermore, optimization solvers employ a number of heuristics; although these can accelerate the solution on average, their efficacy for a given problem is not known a-priory. Therefore, finding the optimal configuration of an optimization solver is a challenging black-box optimization problem since the number of possible combinations of parameters can be very large, their optimal values may change significantly for different classes of optimization problems, and the evaluation of a given configuration can be computationally expensive.

Automated algorithm configuration approaches aim to either solve the algorithm configuration problem in a computationally efficient way or approximate its solution. In the former approach, Bayesian optimization and derivative-free methods have been employed to tune optimization algorithms by optimizing directly the black-box performance function (Liu et al., 2019; Chen et al., 2011; Hutter et al., 2009, 2011, 2010). In the latter approach, ML tools have been used to approximate the performance function m with a surrogate one $\hat{m}$ and then identify the optimal values of the parameters n^*. These methods have been traditionally applied for tuning monolithic solvers, either by considering all the parameters simultaneously (Iommazzo et al., 2020) or specific algorithmic steps such as branching (Lodi and Zarpellon, 2017; Khalil et al., 2016; Balcan et al., 2018; Di Liberto et al., 2016; Gupta et al., 2020, 2022; Liu et al., 2022), cutting plane methods (Tang et al., 2020; Huang et al., 2022; Paulus et al., 2022), estimating active constraints (Misra et al., 2022; Bertsimas and Stellato, 2021, 2022), and improving primal heuristics (Ding et al., 2020). ML tools have also been used to tune decomposition-based solution methods. For hierarchical decomposition-based methods, such as column generation and Benders Decomposition, ML is used to aid the cut selection process using classification techniques (Morabit et al., 2021; Jia and Shen, 2021; Lee et al., 2020), whereas for distributed algorithms, such as ADMM, machine learning is used to determine the values of the dual variables and penalty parameters (Biagioni et al., 2020; Zeng et al., 2022). In both cases, ML is used in the coordination step of these algorithms.

These ML approaches are based on a handcrafted feature representation of an optimization problem which is the input to the surrogate model $\hat{m}$ (Smith-Miles and Lopes, 2012; Hutter et al., 2014; Bengio et al., 2021; Chen et al., 2021), and involve two steps; the first is offline where multiple problems are solved for different values of the parameters in order to create a training dataset used to learn a surrogate model for the performance function. Once the surrogate model is trained, it is used online to identify the best set of parameters for a given problem. The main limitation of these ML approaches is data availability since generating a large training data set can be computationally expensive.

In this work, we focus on the *initialization* of decomposition-based solution methods. Specifically, we focus on cutting plane-based hierarchical solution methods, such as Benders and Generalized Benders Decomposition. These algorithms are based on the observation that if a subset of the variables, called complicating variables, is fixed then the problem is either easier to solve or has a special structure. Thus the original problem is decomposed into two problems; a master problem which considers the complicating variables and a subproblem which considers the non-complicating variables and whose solution depends on the values of the complicating variables. The solution of the master problem and the subproblem is coordinated via the addition of optimality and feasibility cuts, which inform the master problem about the bounds and the feasibility of the problem respectively. In the standard application of the algorithm, the master problem is initially solved without any cuts. Since the cuts contain information about the effect of the complicating variables on the subproblem, the addition of cuts in the first iteration can potentially lead to a reduction in the computational time since fewer iterations might be necessary. However, the addition of a large number of cuts may increase the computational complexity of the master problem which in turn may increase the solution time. Hence it is important to identify the number of cuts that balances the amount of information added to the master problem with the increase in computational complexity. This balance is especially important for online applications where an optimization problem is solved repeatedly to compensate for updated process information. Depending on the application, either only the parameters of the optimization problem change, such as in model predictive control applications (Tang et al., 2018; Mitrai and Daoutidis, 2023a), or the parameters and the number of variables and constraints can change, such as in online scheduling applications (Mitrai and Daoutidis, 2022a; Risbeck et al., 2019).

In this paper, we propose an ML approach to learn how to initialize Generalized Benders Decomposition. The proposed approach has two steps. In the first, ML is used to learn a surrogate model that estimates the solution time of the optimization problem for a given number of cuts added to the master problem in the first iteration of the algorithm. In the second step, the surrogate model is used online to determine the optimal number of cuts that should be added in the master problem in the first iteration. We apply the proposed approach to a case study on the real-time operation of process systems, where a mixed-integer economic model predictive control problem is solved. Specifically, we assume that the system is an isothermal continuously stirred tank reactor (CSTR) that can manufacture a number of products, and multiple disturbances can affect the operation of the system. The mixed-integer economic model predictive control problem is solved using a hybrid multicut Generalized Benders Decomposition proposed in Mitrai and Daoutidis (2022b), initialized using the proposed approach. The results show that (1) the optimal initialization can be achieved in an automated way without human intervention, (2) the proper initialization can lead to a significant reduction in solution time, and (3) active learning can guide the learning process either during the initial development of such frameworks or for cases where generating the training dataset is computationally expensive.

The rest of the paper is organized as follows: In Section 2 we present the Generalized Benders Decomposition algorithm and the different acceleration techniques, in Section 3 we pose the initialization of Generalized Benders Decomposition as an algorithm configuration problem, in Section 4 we present the proposed approach, and in Section 5 we present the case study and the numerical results. Finally, in Section 6, given the results in the first part of this two-series paper (Mitrai and Daoutidis, 2023b) we present a unified framework for automated decomposition-based solution algorithm selection and configuration.

2. Generalized Benders Decomposition

2.1. Standard implementation

We will assume that the following problem (denoted as P) must be solved:

$$\begin{aligned} P(p) := \underset{z,x,y}{\text{minimize}} \quad & f_1(z,x;p_m) + f_2(x,y;p_s) \\ \text{subject to} \quad & g_1(z,x;p_m) \le 0 \\ & h_1(z,x;p_m) = 0 \\ & g_2(x,y;p_s) \le 0 \\ & h_2(x,y;p_s) = 0 \\ & z \in \mathbb{Z}^{n_z}, x \in \mathbb{R}^{n_x^c} \times \mathbb{Z}^{n_x^d}, y \in \mathbb{R}^{n_y}, \end{aligned} \tag{2}$$

where $p = [p_m, p_s]^\top$ are the parameters of the problem, and $z \in \mathbb{Z}^{n_z}$, $x \in \mathbb{R}^{n_x^c} \times \mathbb{Z}^{n_x^d}$, $y \in \mathbb{R}^{n_y}$ are the variables. The solution of this problem depends on the values of the parameters p. In this problem, we observe that if the variables z, x are fixed, then the resulting problem is a continuous optimization problem whose solution depends on the values of the fixed variables x, which are the complicating variables, and the parameters p_s. Given this structure, we can apply Generalized Benders Decomposition, by assigning the z, x variables and the associated constraints in the master problem and the other variables (y) and constraints in the subproblem. Under this decomposition, parameters p_m affect only the master problem and parameters p_s affect only the subproblem. The subproblem is

$$\begin{aligned} S(x,p_s) := \underset{\bar{x},y}{\text{minimize}} \quad & f_2(\bar{x},y;p_s) \\ \text{subject to} \quad & g_2(\bar{x},y;p_s) \le 0 \\ & h_2(\bar{x},y;p_s) = 0 \\ & \bar{x} = x \quad : \quad \lambda \\ & \bar{x} \in \mathbb{R}^{n_x^c + n_x^d}, y \in \mathbb{R}^{n_y}, \end{aligned} \tag{3}$$

where λ are the Lagrangian multipliers for the equality constraint $\bar{x} = x$. The solution of this problem depends on the values of the complicating variables x and the parameters p_s, and for a given value of $x = \bar{x}$ the value function of the subproblem can be approximated as follows:

$$S(x,p_s) \ge S(\bar{x},p_s) - \bar{\lambda}(x - \bar{x}), \tag{4}$$

where $\bar{\lambda}$ is equal to the value of the Lagrangian multiplier at the optimal solution of the subproblem when solved for $x = \bar{x}$. Note that we assume that the subproblem is always feasible for all values of x. The master problem is:

$$\begin{aligned} \mathcal{M}(\cdot) := \underset{z,x,\eta}{\text{minimize}} \quad & f_1(z,x;p_m) + \eta \\ \text{subject to} \quad & g_1(z,x;p_m) \le 0 \\ & h_1(z,x;p_m) = 0 \\ & \eta \ge S(\bar{x}^l,p_s) - \lambda^l(x - \bar{x}^l) \ \forall l \in \mathcal{L} \\ & z \in \mathbb{Z}^{n_z}, x \in \mathbb{R}^{n_x^c} \times \mathbb{Z}^{n_x^d}, \end{aligned} \tag{5}$$

where $\mathcal{M}(\cdot) = \mathcal{M}(p_m, \mathcal{L})$, l is the iteration number and the set $\mathcal{L}$ denotes the index of the Benders cuts. The steps of GBD are presented in Algorithm 1.

2.2. Acceleration techniques for Benders Decomposition

The algorithm alternates between the solution of the master problem and the subproblem, therefore the computational performance depends on the complexity of the master problem and subproblem, i.e., the solution time per iteration, the number of infeasible subproblems that must be solved, and the quality of cuts that are generated during the solution. Two approaches can be followed to handle these issues. The first one is based on the theoretical aspects of the algorithm and the underlying geometry of the problem. Common strategies in this approach are problem reformulation and decomposition (Crainic et al., 2016; Magnanti and Wong, 1981), the addition of valid inequalities in the master problem to reduce the number of infeasible subproblems (Saharidis et al., 2011), multicut implementation (You and Grossmann, 2013) for stochastic optimization problems, cut generation and management (Magnanti and Wong, 1981; Su et al., 2015; Saharidis and Ierapetritou, 2010; Pacqueau et al., 2012; Varelmann et al., 2022), and regularization/stabilization of the master problem (Ruszczyński and Świetanowski, 1997; Linderoth and Wright, 2003). The second approach involves the use of ML techniques. For example, ML has been used to develop a classifier to determine which cuts should be added in the master problem during the multicut implementation of the algorithm for the solution of two-stage mixed-integer stochastic optimization problems (Jia and Shen, 2021; Lee et al., 2020). ML has also been used to approximate the solution of the subproblem reducing the solution time for cases where the subproblem is computationally complex, such as two-stage stochastic optimization problems (Larsen et al., 2023) and mixed-integer model predictive control problems (Mitrai and Daoutidis, 2023a). These acceleration methods, both the ones based on the underlying geometry and the ones using ML, reduce the computational time either by reducing the solution time per iteration or by reducing the number of iterations.

Data: Optimization problem
Result: Upper, lower bound and variable values
Set $UB = \infty$, $LB = -\infty$;
Set tolerance and optimality gap (tol);
Initialize the algorithm;
while $(UB - LB)/LB \ge tol/100$ **do**
  Solve the master problem Eq. (5) and obtain LB, x;
  Solve the subproblem Eq. (3) and obtain y;
  Add Benders cut Eq. (4);
  Update the upper bound $f_1(z,x;p_m) + f_2(x,y;p_s)$;
end

Algorithm 1: Generalized Benders Decomposition

3. Initialization of GBD as an algorithm configuration problem

We will assume that problem $P(p)$ must be solved repeatedly given new values of the parameters p. The problem that we will address is the following:

Problem 2. Given an optimization problem $P(p)$ determine the optimal number of cuts to add in the master problem in the first iteration of Generalized Benders Decomposition, such that the CPU time is minimized.

We can pose this as a per-instance algorithm configuration problem as follows

$$\underset{n \in \mathcal{N}_{cuts}}{\text{minimize}} \quad m(n, P(p)), \tag{6}$$

where the optimization problem is $P(p)$, the parameter space $\mathcal{N}_{cuts}$ represents the cuts that can be added, and the performance function is the solution time $\mathcal{M} = \mathbb{R}_+$. The solution of the algorithm configuration problem for Generalized Benders Decomposition has three main challenges. The first, which is common in algorithm configuration problems, is that the performance function m is not known a-priory. The second issue is related to the parameter space, i.e., all the cuts that can potentially be used to initialize the master problem. In general, selecting which cuts to use is a challenging problem that arises in the solution of many classes of problems such as quadratic (Baltean-Lugojan et al., 2019; Marousi and Kokossis, 2022), and mixed-integer linear programming problems (Morabit et al., 2021; Chi et al., 2022). Regarding Generalized Benders Decomposition, the number of cuts that

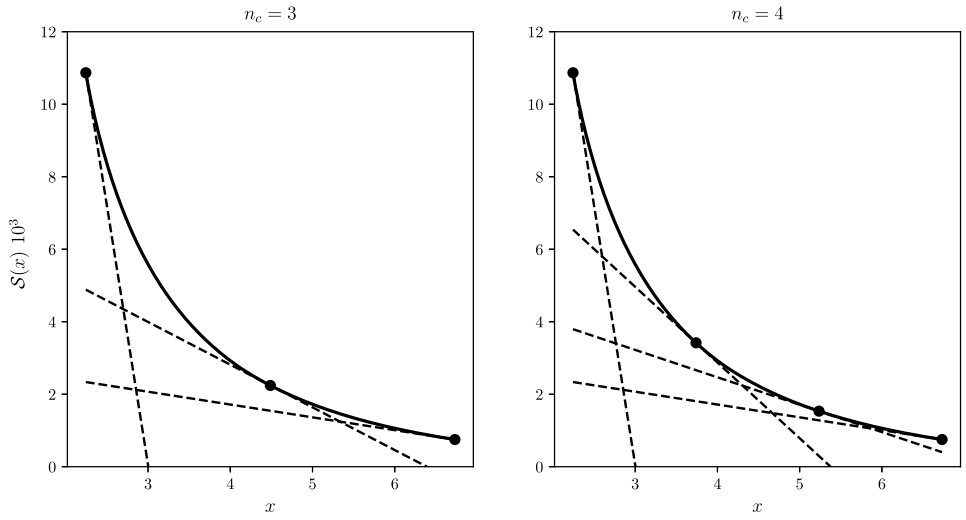

Fig. 1. Domain discretization for the case study considered in Section 5 for a transition from product 1 to 2 for three and four number of cuts (n_c). The solid line is the value function, $x \in [2.24, 6.73]$ is the complicating variable, the dotted lines are the value function approximations, i.e., Benders cuts, evaluated ad the points indicated by the dots.

can be evaluated depends on the number and type of complicating variables. For cases where the complicating variables are integer and n_c cuts must be added, the cut selection process leads to a combinatorial optimization problem since one must select n_c cuts from all possible cuts. The situation is more complex when the complicating variables are continuous, since in this case an infinite number of cuts can be added even for one complicated variable. Finally, all the parameters of the problem can change simultaneously, thus the cuts must be evaluated continuously as the parameter of the problem change. Hence, although the addition of cuts in the master problem might reduce the number of iterations required for convergence, the solution time might increase since multiple subproblems must be solved to evaluate the initial set of cuts.

We will assume that (1) the parameters of the subproblem do not change, (2) all the complicating variables are continuous, and (3) n_c cuts can be added by discretizing the domain of the complicating variables ($x \in [x^{lb}, x^{ub}]$) into n_c uniform points. The first assumption guarantees that the Benders cuts must be evaluated only once since even if the parameters of the master problem change, the Benders cuts are still valid estimators of the value function of the subproblem, since the parameters of the subproblem do not change. Therefore, the process of adding these cuts to the master problem does not affect the total solution time. The second and third assumptions determine the cut selection strategy. In this work, the number of cuts determines the number of points that will be used to approximate the value function of the subproblem via Benders cuts. An example is presented in Fig. 1, where the value function corresponds to the system discussed in Section 5 and the approximation with three and four cuts is presented. This setting enables us to simplify the cut selection process for the case of continuous complicating variables. Notice that in this setting the cuts are uniformly distributed in the domain of the complicating variables.

4. Learning to initialize via supervised and active learning

The goal is to learn a surrogate model $\hat{m}$ which approximates the solution time and can be used to identify the optimal number of cuts to add in the master problem,

$$n^* \in \arg\min_{n \in \mathbb{Z}_+} \ \hat{m}(n, \nu(\mathcal{P}(p))), \tag{7}$$

where $\nu(P(\mathbf{p}))$ are some features of problem $P(p)$. We propose two ML approaches for learning the surrogate model.

4.1. Supervised learning approach

The estimation of the parameters of a surrogate model requires data that capture the relation between the features of an optimization problem $\nu(P(p))$ and the number of cuts n with the solution time. Such data can be generated using the procedure presented in Algorithm 2, where first random values of the parameters of the master problem (p_m) are generated based on some underlying probability distribution. Next, the optimization problem $P(p_i)$ is solved for a fixed number of cuts n_i, and the solution time y_i is obtained. Finally, the features of the problem $\nu(P(p_i))$ are obtained and the tuplet $(n_i, \nu(P(p_i)))$ and the solution time y_i are stored. Once this procedure is completed, we obtain the dataset $\mathcal{D} = \{(n_i, \nu(P(p_i))), y_i\}_{i=1}^{N_{data}}$. This dataset can be used to learn the parameters of the surrogate model by optimizing some loss function such as the squared error between the model prediction and the data. Once the learning step is completed, the surrogate model can be used to learn how many cuts to add for online applications as presented in Algorithm 3, where given new values of the parameters of the optimization problem, the features of the problem are obtained and the optimal number of cuts is computed by optimizing the surrogate model. Finally, the cuts are added to the master problem and Generalized Benders Decomposition is implemented.

4.2. Active learning approach

The main limitation of the supervised learning approach is the computational time required to generate the training data, since for every value of the parameters p_i and number of cuts n_i the problem must be solved to obtain the solution time. This approach can be computationally expensive, even intractable for complex optimization problems. To resolve this we propose the application of active learning (Settles, 2009), a commonly used approach in ML tasks where the features of the data are known but obtaining their label is costly or time-consuming. Unlike supervised learning where all the data are available for training, in active learning the model itself determines which data should be labeled and thus be used for training the surrogate model.

The active learning paradigm has three components. The first is the unlabeled data which can be generated de novo (known as membership query synthesis) (Angluin, 1988), can become available in an online setting (stream-based selective sampling) (Cohn et al., 1994), or can

Data: Optimization problem, number of data points N_{data}, number of discretization points N_{cuts}, upper and lower bounds for complicating variables $x \in [x^{lb}, x^{ub}]$, $\check{p} = [p_4, p_5, p_6]$
Result: Surrogate model $\hat{m}$
$i = 1$;
while $i \leq N_{data}$ **do**
  Generate parameters $p_1, p_2, p_3 \rightarrow p_i = [p_1, p_2, p_3, \check{p}]$;
  for $j = 2 : N_{cuts}$ **do**
    Solve problem $\mathcal{P}(p_i)$ using j cuts for each x;
    Obtain CPU time y_i;
    Obtain features of the problem $s_j = (\nu(\mathcal{P}(p_i)), j)$;
    Append data $\{s_j, y_j\}$;
  end
  $i = i + 1$;
end
Using data $\{s_i, y_i\}_{i=1}^{N_{data}(N_{cuts}-1)}$ learn parameters of a surrogate model $\hat{m}$;

Algorithm 2: Learning the relation between number of cuts and CPU time for a general optimization problem for continuous complicating variables

be gathered at once (pool-based sampling) (Lewis and Gale, 1995). The second aspect is the query strategy, which determines which data should be labeled. This decision is taken by considering the informativeness of the available unlabeled data. Typical query strategies are uncertainty sampling (Lewis and Gale, 1995), query by committee (Seung et al., 1992), and expected model change (Settles et al., 2007). We refer the reader to Settles (2009) for a detailed discussion of the different sampling methods. The last component is an oracle which generates the label for a given input. Typical example of an oracle is a human expert, a computer simulation or the outcome of an experiment.

In this work, we will use the pool-based active learning paradigm with uncertainty-based sampling, where the data point for which the model is the least certain is labeled. The basic steps of the application of active learning for learning the solution time of Generalized Benders Decomposition are presented in Fig. 2.

4.2.1. Generation of pool of labels and initial training set

For the application of active learning, first, the pool of labels is created. We generate random values of the parameter p_i and for every parameter the features of the optimization problem $\nu(P(p_i))$ are obtained and a number of cuts n_i is selected. This forms the pool of features $C_p = \{s_i\}_{i=1}^{N_{pool}}$ ($s_i = \{n_i, \nu(P(p_i))\}$). Next, a small set of training data is obtained by sampling $N_{initial}$ data points from the pool and evaluating the solution time y_i. This is the initial training set $C = \{s_i, y_i\}_{i=1}^{N_{initial}}$ which we will refer to as labeled training set. Given the unlabeled pool C_p and the labeled dataset C, the surrogate model considers all the data in the pool, and the data point $s = \{n_s, \nu(P(p_s))\}$ for which the prediction is the least certain about is selected for labeling.

4.2.2. Uncertainty-based sampling using Gaussian processes

The selection of the data point requires quantification of the prediction uncertainty. Gaussian Process Regression (GPR) is a non-parametric Bayesian approach that can provide uncertainty measures (Williams and Rasmussen, 2006). GPR is based on the Gaussian Process which is a stochastic process that defines a distribution over functions. Specifically, given observations of the input variables $X = [x_1, \ldots, x_N]$ and measurements of the output $Y = [y_1, \ldots, y_N]$, the relationship between X and Y is modeled as a Gaussian multivariate distribution. GPR seeks to learn a mapping $f : X \mapsto Y$, i.e., $y = f(x)$, with mean $m(x)$ and covariance $k(x, x')$ where ($\mathbb{E}$ is the expected value)

$$\begin{aligned} m(x) &= \mathbb{E}[f(x)] \\ k(x, x') &= \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]. \end{aligned} \tag{8}$$

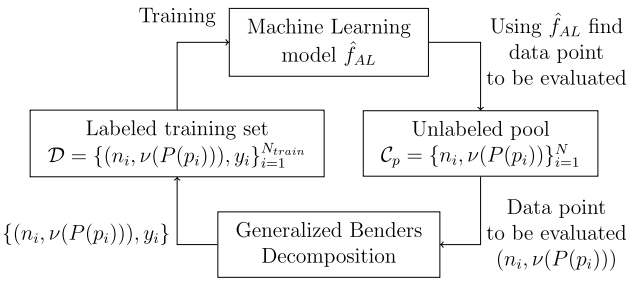

Fig. 2. Learning to initialize Generalized Benders Decomposition via active learning framework.

This is done under the assumption that the data are independent and the probability to observe an output given the observations can be factored over cases in the training set. The Gaussian Process, is written as $f(x) \sim \mathcal{GP}(m(x), k(x, x'))$. Different kernel functions $k(\cdot, \cdot)$ can be used, however in this paper we use the Matern kernel given by the following equation

$$k(\cdot, \cdot) = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\frac{\sqrt{2\nu}}{\ell} d(\cdot, \cdot) \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu}}{\ell} d(\cdot, \cdot) \right), \tag{9}$$

where $d(\cdot, \cdot)$ is the Euclidean distance between features s_i and s_j, $\Gamma(\cdot)$ is the gamma function, $K_{\nu}(\cdot)$ is the modified Bessel function, and ℓ, ν are tunable hyperparameters. During training the parameters of the mean and kernel function are estimated based on the available data. This step estimates the posterior distribution over functions that best explain the data. This posterior distribution is also Gaussian and is used to make predictions for a new data point. We refer the reader to Williams and Rasmussen (2006) for detailed explanation of these steps.

4.2.3. Oracle

The oracle is the Generalized Benders Decomposition algorithm which given the parameters of an optimization problem and a number of cuts, solves the optimization problem and returns the solution time. We note that although here we consider the standard version of Generalized Benders Decomposition as the oracle, in principle, other versions can be incorporated, such as multicut implementation, partial Benders decomposition etc.

4.2.4. Active learning loop

The active learning loop is presented in Fig. 2 and Algorithm 4. The inputs are the pool of unlabeled data C_p, the initial training set C, and the surrogate model $\hat{m}$ which is a Gaussian Process with Matern kernel. First, the model is trained using the initial training dataset. Next, the model is used to predict the solution time and uncertainty around the prediction for all the unlabeled data and identify the datapoint s with the maximum uncertainty. This data point is passed to the Generalized Benders Decomposition algorithm and the solution time (the label) is recorded, the labeled datapoint $\{s, y\}$ is appended in the training dataset, and the datapoint with label s is removed form the pool. The surrogate model is trained again using the new training dataset and this loop continues until the maximum number of iterations is reached. The outcome of this approach is the surrogate model $\hat{m}$.

Remark 1. In this paper, we used a pool-based active learning approach with uncertainty-based sampling. However different active learning paradigms have been proposed. These paradigms can potentially be exploited for learning how to initialize decomposition-based and monolithic-based solution algorithms for different applications. For example, stream-based selective sampling can be used for learning online the surrogate models for the performance function, thus enabling the online learning of the optimal initialization.

Data: Surrogate model $\hat{m}$, Optimization problem P, value of parameters p
Result: Problem solution
Compute the features of the problem $\nu(P(p))$;
Determine the optimal number of cuts $n^*_{cuts} = \arg\min_{n \in \mathcal{N}_{cuts}} \hat{m}(N, \nu(P(p)))$;
Add n^*_{cuts} cuts in the master problem;
Solve the optimization problem using Algorithm 1;

Algorithm 3: Regression based initialization of Generalized Benders decomposition

Data: Number of evaluations N, initial labeled dataset $C = \{s, y\}$, Pool of labels C_p, Surrogate model $\hat{m}$
Result: Surrogate model $\hat{m}$
Train surrogate model $\hat{m}$ on the initial dataset C;
while $i \leq N$ **do**
  Select a data point with features s from pool C_p based on maximum uncertainty sampling strategy $s \in \arg\max_{s \in C_p} \sigma(s)$;
  Evaluate label y for s using GBD;
  Append data $\{s, y\}$ in set $C = C \cup \{s, y\}$;
  Remove data-point s from pool $C_p = C_p \setminus \{s\}$;
  Train surrogate model $\hat{m}$ using set C;
  $i = i + 1$;
end

Algorithm 4: Learning surrogate model for solution time via active learning

5. Application to mixed-integer economic model predictive control for real-time operation of chemical processes

In this section, we apply the proposed method for learning to initialize Generalized Benders Decomposition for the solution of mixed-integer economic model predictive control problems that arise in the operation of chemical processes. Specifically, we consider a continuous manufacturing system whose operation can be affected by disturbances in the scheduling, e.g., change in demand, and control level, e.g., change in the inlet conditions of the process, as presented in Fig. 3. Once a disturbance affects the system an optimization problem is solved to determine the production sequence and the transitions between the products. In this section, we present the optimization model, the decomposition-based solution approach, the data generation process, and the evaluation of the different learning approaches.

5.1. Optimization model

5.1.1. Scheduling model

We will consider the case where an isothermal CSTR is used to produce N_p products over a time horizon of H hours which is discretized into N_s slots ($N_s = N_p$). We will assume that while the system is following a nominal schedule a disturbance affects the system at time T_0, as presented in Fig. 3. Under this setting, every slot k except the first one has two regimes; a production regime where a product is manufactured and a transition regime where a transition occurs from the operating point of the product manufactured at slot k to the operating point of the product manufactured at slot $k+1$. The first slot has three regimes; a transition regime that captures the transition from some intermediate state (where the system is due to the disturbance) to the operating point of the product manufactured in the first slot, a production regime, and another transition regime which considers the transition from the product manufactured at the first slot to the product manufactured in the second slot.

We define a binary variable W_{ik} which is equal to one if product i is manufactured in slot k and zero otherwise. Also, we define a binary variable Z_{ijk} which is equal to one if a transition occurs from product i to j in slot k, and variable $\hat{Z}_i$ which is equal to one if a transition occurs from an intermediate state to product i in the first slot. At every time point, only one product can be manufactured. The logic constraints are:

$$\begin{aligned} &\sum_{i=1}^{N_p} W_{ik} = 1 \ \forall k = 1, \dots, N_s \\ &Z_{ijk} \geq W_{ik} + W_{j,k+1} - 1 \ \forall i, j, k \neq N_s \\ &\hat{Z}_i = W_{i1} \ \forall i \end{aligned} \tag{10}$$

The starting time of slot k is T^s_k, the ending time T^e_k, the production time of product i in slot k is Θ_{ik}, and the transition time in slot k is θ^t_k. The timing constraints are:

$$\begin{aligned} &T^e_k = T^s_k + \sum_{i=1}^{N_p} \Theta_{ik} + \theta^t_k \ \forall k = 1, \dots, N_s \\ &T^s_{k+1} = T^e_k \ \forall k = 1, \dots, N_s - 1 \\ &T^e_{N_s} = H - T_0 \\ &\Theta_{ik} \leq H W_{ik} \ \forall i = 1, \dots, N_p, k = 1, \dots, N_s \\ &\theta^t_1 = \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} Z_{ij1} \theta_{ij1} + \sum_{i=1}^{N_p} \hat{Z}_i \hat{\theta}_i \\ &\theta^t_k = \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} Z_{ijk} \theta_{ijk} \ \forall k = 2, \dots, N_s \\ &\theta_{ijk} \geq \theta^{min}_{ij} \ \forall i, j, k \\ &\hat{\theta}_i \geq \hat{\theta}^{min}_i \ \forall i \end{aligned} \tag{11}$$

where θ_{ijk} is the transition time from product i to j in slot k, $\hat{\theta}_i$ is the transition time from an intermediate state to the steady state of product i, θ^{min}_{ij} is the minimum transition time from product i to j, and $\hat{\theta}^{min}_i$ is the minimum transition time from the intermediate state to the steady state of product i. The production rate of product i is r_i, the production amount of product i in slot k is q_{ik}, and the inventory of product i in slot k is I_{ik}. The production constraints are

$$\begin{aligned} &I_{ik} = I_{ik-1} + r_i \Theta_{ik} - S_{ik} \ \forall i = 1, \dots, N_p, k = 2, \dots, N_p \\ &I_{ik} = I^0_i + r_i \Theta_{ik} - S_{ik} \ \forall i = 1, \dots, N_p, k = 1, \end{aligned} \tag{12}$$

where I^0_i is the initial inventory of product i. The demand of product i is d_i and the due date for every product is in the end of the time horizon. The demand satisfaction constraints are

$$S_{iN_s} \geq d_i \ \forall i = 1, \dots, N_p \tag{13}$$

5.1.2. Dynamic model

We will assume that the system is described by a system of differential equations

$$\dot{x} = F(x, u), \tag{14}$$

where $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$ are the state and manipulated variables, and $F : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \mapsto \mathbb{R}^{n_x}$ are vector functions. We will consider simultaneously all transitions between the products and discretize the differential equations using the method of orthogonal collocation on finite elements (using N_f finite elements and N_c collocation points). We define state variable x_{ijkfc} and manipulated variable u_{ijkfc} for a transition from product i to j in slot k, finite element f and collocation point c. We also define variable $\hat{x}_{ifc}$ and manipulated variable $\hat{u}_{ifc}$ for a transition from the intermediate state to product i in finite element f and collocation point c in the first slot. The discretized differential

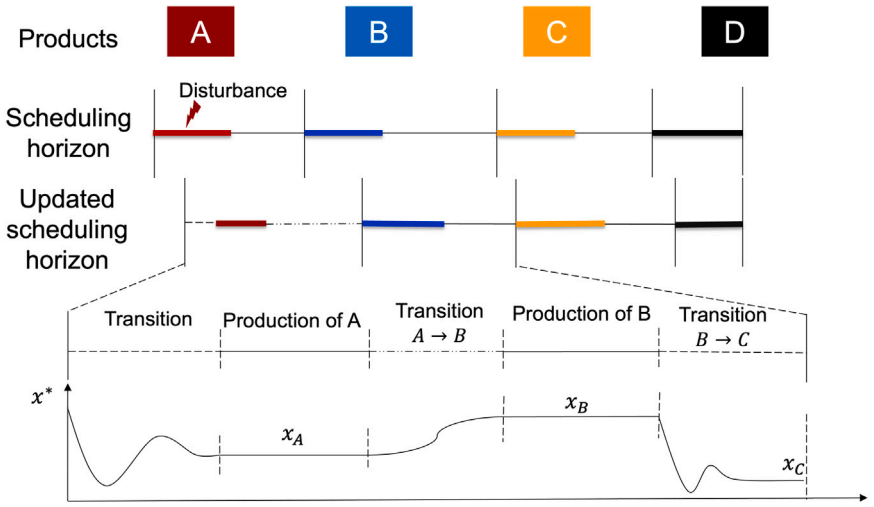

Fig. 3. Schematic of rescheduling.

equations for transitions between products are

$$\begin{aligned} x_{ijkfc} &= F_d(x_{ijkfc}, u_{ijkfc}, \theta_{ijk}) \ \forall i,j,k,f,c \\ x_{ijk11} &= x_i^{ss} \ \forall i,j,k \\ x_{ijkN_fN_c} &= x_j^{ss} \ \forall i,j,k \\ u_{ijk11} &= u_i^{ss} \ \forall i,j,k \\ u_{ijkN_fN_c} &= u_j^{ss} \ \forall i,j,k \end{aligned} \tag{15}$$

and the equations for the transition from the intermediate state to product i are:

$$\begin{aligned} \hat{x}_{ifc} &= F_d(\hat{x}_{ifc}, \hat{u}_{ifc}, \hat{\theta}_i) \ \forall i,l \\ \hat{x}_{i11} &= x^* \ \forall i \\ \hat{x}_{iN_fN_c} &= x_i^{ss} \ \forall i \\ \hat{u}_{i11} &= u^* \ \forall i \\ \hat{u}_{iN_fN_c} &= u_i^{ss} \ \forall i \end{aligned} \tag{16}$$

where x_i^{ss}, u_i^{ss} are the steady state values of the state and manipulated variables of product i. A detailed expression for the discretized differential equations can be found at Mitrai and Daoutidis (2022b).

5.1.3. Objective function

The objective function has three terms; the first is the profit Φ_1, the second is the transition cost between products Φ_2, and the third is the transition cost from the intermediate state Φ_3. These terms are equal to:

$$\begin{aligned} \Phi_1 &= \sum_{i,k} p_{ik} S_{ik} - C_{ik}^{oper} q_{ik} - C^{inv} I_{ik} - \sum_{ijk} C_{ijk}^{trans} Z_{ijk} \\ \Phi_2 &= \sum_{ijk} Z_{ijk} \alpha_u \left(\sum_{fc} N_f^{-1} t_{ijfck}^d \Lambda_{cN_c} (u_{ijfck} - u_j^{ss})^2 \right) \\ \Phi_3 &= \sum_{i} \hat{Z}_i \alpha_u \left(\sum_{fc} N_f^{-1} \hat{t}_{ifc}^d \Lambda_{cN_c} (\hat{u}_{ifc} - u_j^{ss})^2 \right) \end{aligned} \tag{17}$$

where p_i, C_i^{op} are the price and operating cost of product i, C^{inv} is the inventory cost, C_{ij}^{tr} is the transition cost from product i to j, a_u is a weight coefficient, and Λ is the collocation matrix.

5.1.4. Mixed-integer model predictive control problem and decomposition-based solution

The mixed-integer MPC problem is:

$$\begin{aligned} P(p): \ &\text{minimize} \quad \Phi_1 - \Phi_2 - \Phi_3 \\ &\text{subject to} \quad \text{Eqs. (10), (11), (12), (13), (15), (16).} \end{aligned} \tag{18}$$

where $p = \{\{d_i\}_{i=1}^{N_p}, \{I_i^0\}_{i=1}^{N_p}, T_0, \{\theta_{ij}^{min}\}_{i=1,j=1}^{N_pN_p}, \{\hat{\theta}_i\}_{i=1}^{N_p}, \{r_i\}_{i=1}^{N_p}, \{x_i^{ss}\}_{i=1}^{N_p}, \{u_i\}_{i=1}^{N_p}, x^*\}$. This problem has three sets of parameters; the ones related to the scheduling part of the problem $\hat{p} = \{\{d_i\}_{i=1}^{N_p}, \{I_i^0\}_{i=1}^{N_p}, T_0, \{\theta_{ij}^{min}\}_{i=1,j=1}^{N_pN_p}, \{\hat{\theta}_i\}_{i=1}^{N_p}, \{r_i\}_{i=1}^{N_p}\}$, ones related to the dynamic behavior of the system for transitions between products $\check{p} = \{\{x_i^{ss}\}_{i=1}^{N_p}, \{u_i^{ss}\}_{i=1}^{N_p}\}$, and the ones related to the transition from the intermediate state to the steady state of the different products $\tilde{p} = x^*$.

We will rewrite the mixed-integer economic MPC problem (Eq. (18)) as follows

$$\begin{aligned} \text{maximize} \quad & \Phi_1(w) - \sum_{ijk} Z_{ijk} f_{dyn}^{ijk}(\omega_{ijk}, \theta_{ijk}) - \sum_i \hat{Z}_i f_{dyn}^{i}(\hat{\omega}_i, \hat{\theta}_i) \\ \text{subject to} \quad & g_{sched}(w, \theta_{ijk}, \hat{\theta}_i; \hat{p}) \le 0 \\ & g_{ijk}^{dyn}(\theta_{ijk}, \omega_{ijk}; \check{p}) \le 0 \ \forall i,j,k \\ & \hat{g}_i^{dyn}(\hat{\theta}_i, \hat{\omega}_{ijk}; \hat{p}) \le 0 \ \forall i \end{aligned} \tag{19}$$

where $w = \{W_{ik}, Z_{ijk}, \hat{Z}_i, T_k^s, T_k^t, \Theta_{ik}, \theta_k^t, S_{ik}\}$ are scheduling variables, $\omega_{ijk} = \{x_{ijkfc}, u_{ijkfc}\}$ are variables associated with the dynamic behavior of the system for a transition from product i to product j in slot k, and $\hat{\omega}_i = \{\hat{x}_{ifc}, \hat{u}_{ifc}\}$ are variables associated with the dynamic behavior of the system for a transition from the intermediate state to the product i. g_{sched} are the scheduling constraints (Eqs. (10), (11), (12), (13)), g_{ijk}^{dyn} are the discretized differential equations for transitions between products (Eq. (15)), and $\hat{g}_i^{dyn}$ are the discretized differential equations for transition from the intermediate state to product i (Eq. (16))

If we fix the scheduling variables w and the transition times $\theta_{ijk}, \hat{\theta}_i$ then the dynamic optimization problems for all the transitions can be solved independently. We define as ϕ_{ijk} the value function of the dynamic optimization problem for a transition from product i to product j in slot k. The dynamic optimization problem for this transition is

$$\begin{aligned} \phi_{ijk}(\theta_{ijk}) := \underset{\omega_{ijk}, \tilde{\theta}_{ijk}}{\text{minimize}} \quad & f_{dyn}^{ijk}(\omega_{ijk}, \tilde{\theta}_{ijk}) \\ \text{subject to} \quad & g_{dyn}(\tilde{\theta}_{ijk}, \omega_{ijk}) \le 0 \\ & \tilde{\theta}_{ijk} = \theta_{ijk} \quad : \lambda_{ijk}. \end{aligned} \tag{20}$$

where λ_{ijk} is the Lagrangian multiplier for the equality constraint. Similarly, we define the value function for a transition from the intermediate state to product i, $\hat{\phi}_i$, and the dynamic optimization problem for this transition is

$$\begin{aligned} \hat{\phi}_i(\hat{\theta}_i, \check{p}) := \underset{\hat{\omega}_i, \check{\theta}_i}{\text{minimize}} \quad & f_{dyn}^{ijk}(\hat{\omega}_i, \check{\theta}_i) \\ \text{subject to} \quad & g_{dyn}(\check{\theta}_i, \hat{\omega}_i; \check{p}) \le 0 \\ & \check{\theta}_i = \hat{\theta}_i \quad : \hat{\lambda}_i. \end{aligned} \tag{21}$$

Data: Number of data points N_{data}, Maximum number of discretization points N_{cuts}, upper and lower bounds for complicating variables $\theta_{ijk} \in [\theta_{ij}^{min}, 5\theta_{ij}^{min}]$, Demand distribution for every product, Distribution of inlet concentration in the reactor, Scheduling horizon H
Result: Pool of unlabeled data C_p
1 $C_p = \{\}$;
2 $i = 0$;
3 **while** $i \leq N_{data}$ **do**
4 Select at random a time point $T_0 \sim U(0, H)$;
5 Introduce a disturbance in the inlet concentration of the rector $c_0 \sim U(0.8, 1.2)$;
6 Generate new demand for every product $d_i \sim U(\underline{d}_i, \bar{d}_i)$;
7 Compute inventory I_0 at time T_0;
8 Get the value of the concentration in the reactor x^*;
9 Obtain minimum transition times $\hat{\theta}_i^{min}$ from x^* to x_i^{ss};
10 Form and solve the master problem;
11 **if** *Master problem is feasible* **then**
12 **for** $n = 2 : N_{cuts}$ **do**
13 Add n cuts to add in the master problem;
14 Compute features s and append the data point in the pool C_p;
15 **end**
16 **else**
17 The demand cannot be satisfied at the end of the time horizon due to the disturbance, data point is not considered
18 **end**
19 **end**

Algorithm 5: Unlabeled data generation procedure for creating the pool of unlabeled data for active learning

We note that these dynamic optimization problems are always feasible, since the transition times are bounded from below by the minimum transition times, i.e., $\theta_{ijk} \geq \theta_{ij}^{min}$, $\hat{\theta}_i \geq \hat{\theta}_i^{min}$. The value functions $\phi_{ijk}, \hat{\phi}_i$ can be approximated with Benders cuts given by the following equations (Geoffrion, 1972)

$$\begin{aligned} \eta_{ijk} \geq & \phi_{ijk}(\bar{\theta}_{ijk}^l) - \lambda_{ijk}^l(\theta_{ijk} - \bar{\theta}_{ijk}^l) \\ \hat{\eta}_i \geq & \hat{\phi}_i(\bar{\hat{\theta}}_i^l) - \hat{\lambda}_i^l(\hat{\theta}_i - \bar{\hat{\theta}}_i^l), \end{aligned} \tag{22}$$

where $l \in \mathcal{L}$ denotes the number of points used to approximate the value functions. The original problem can be reformulated as

$$\begin{aligned} \text{maximize} \quad & \Phi_1(w) - \sum_{ijk} Z_{ijk}\eta_{ijk} - \sum_i \hat{Z}_i\hat{\eta}_i \\ \text{subject to} \quad & g_{sched}(w, \theta_{ijk}, \hat{\theta}_i) \leq 0 \\ & \eta_{ijk} \geq \phi_{ijk}^l(\bar{\theta}_{ijk}^l) - \lambda_{ijk}^l(\theta_{ijk} - \bar{\theta}_{ijk}^l) \ \forall i, j, k, l \\ & \hat{\eta}_i \geq \hat{\phi}_i(\bar{\hat{\theta}}_i) - \hat{\lambda}_i^l(\hat{\theta}_i - \bar{\hat{\theta}}_i^l) \ \forall i, l. \end{aligned} \tag{23}$$

To solve the problem we use a hybrid multicut Generalized Benders Decomposition algorithm proposed in Mitrai and Daoutidis (2022b). In this algorithm the solution of the master problem provides the production sequence and the transition times, and Benders cuts are added to approximate the transition cost. In this case, the dynamic optimization problems between the products depend only on the transition time, whereas the transition from the intermediate state depends on the transition time and the concentration of the intermediate state. Therefore, the initialization of the algorithm considers only the optimal number of cuts added to approximate the transitions between the products, i.e. how many points should be used to approximate ϕ_{ijk}. We use the same number of cuts for all transitions.

5.2. Application of active learning approach

For the application of active learning, first, we generate the pool of unlabeled data as presented in Algorithm 5. We assume that at a random time point T_0 between 0 and the end of the scheduling horizon, the demand of all the products and the inlet concentration of the reactor change simultaneously based on some probability distributions. The statistics of the demand are presented in Table 2 and we assume that the inlet concentration follows a uniform distribution with a low value of 0.8 and a high value of 1.2. Once the disturbance occurs, we compute the inventory, based on the realization of the initial schedule that was followed for T_0 hours and the concentration x^* inside the reactor. Next, the minimum transition time from the intermediate state x^* to the steady state x_i^{ss} is computed for all the products. Given this information, we formulate a new master problem where the scheduling horizon is $H - T_0$ and check its feasibility. If the master problem is infeasible, then the disturbance that was generated cannot be rejected such that the system can meet the demand at the end of the scheduling horizon. If the master problem is feasible, then different numbers of cuts are added to the master problem, as presented in lines 12–14 in Algorithm 5, and the features of the problem and the number of cuts are added in the unlabeled pool C_p. Note that the feasibility of the mixed-integer MPC problem can be determined by the feasibility of the master problem without any cuts since the subproblems are always feasible and the only source of infeasibility in the mixed-integer MPC problem is due to the inability to satisfy the demand.

The features of the problem, in this case, are the time point T_0, the concentration in the reactor x^*, the inlet flowrate Q, the demand of the products $\{d_i\}_{i=1}^{N_{prod}}$, inventory of the products $\{I_i^0\}_{i=1}^{N_{prod}}$, state of the system, i.e., production or transition, and the number of cuts added n_{cuts}. For the state of the system, we use one-hot encoding, i.e., $state \in \{0, 1\}$. Overall the features s_i for data point i are:

$$s_i = [T_0, x^*, Q, \{d_i\}_{i=1}^{N_{prod}}, \{I_i^0\}_{i=1}^{N_{prod}}, state, n_{cuts}]$$

These features form the pool $C_p = \{s_i\}_{i=1}^{N_{pool}}$ of data points with $N_{pool} = 49000$. Next, we generate a small number of data points ($N_{initial} = 10$) and evaluate the CPU time. We allow 100 evaluations $N = 100$, i.e., 100 data points are labeled. The computational time for obtaining the data is 726 s. The active learning approach is implemented using scikit-learn (Pedregosa et al., 2011) and the multicut Generalized Benders Decomposition is implemented in Pyomo (Hart et al., 2017), the master problem is solved using Gurobi (Gurobi Optimization, LLC, 2021) and the subproblem is solved with IPOPT (Wächter and Biegler, 2006). We note that the solution returned by Generalized Benders Decomposition is not guaranteed to be globally optimal since the problem in Eq. (18) is a non-convex MINLP.

5.3. Active learning results

We compare the proposed active learning approach, denoted as GP-AL, with a random sampling of 110 points from the pool using different surrogates models such as Gaussian Process (GP) with Matern kernel, Neural Network (NN), Random Forest (RF) and Decision Tree (DT). The hyperparameters of the Gaussian Process, Random Forest, and the Decision Tree were set equal to their default values while the neural network has 3 layers, 150 neurons per layer, and `tanh` as activation function.

We generate 100 new disturbances that are not part of the pool and were not used for training in either the active or supervised learning approach. The solution time statistics for the different models are presented in Table 1. From the results, we observe that the active learning approach leads on average to 66.5% reduction in CPU time compared to the standard application of Generalized Benders Decomposition, whereas the Gaussian process, Neural Network, Random Forest, and Decision Tree lead to 53%, 43%, 51% and 33% reduction respectively.

Additionally, for the 100 random disturbances considered, the solution time obtained from the active learning approach is always lower than the solution time without the addition of cuts. The minimum percentage reduction in solution time for the active learning approach is 0.09%, whereas for the surrogate models learned via random sampling, the minimum reduction is negative, i.e., the solution time of the proposed approach is higher than the original implementation of Generalized Benders Decomposition. These results indicate that the optimal number of cuts identified by optimizing the surrogate model trained via random sampling is highly suboptimal. Although these results can be justified by the fact that only 110 data points were used for training, they also highlight the importance of selecting the proper data points to label in cases where obtaining the labels is computationally expensive.

5.4. Application of supervised learning

We also consider the case where the labeling cost is not significant. Specifically, for every data point in the pool C_p with features s_i generated in the previous section, we solve the optimization problem and obtain the solution time y_i, leading to a data set $D = \{s_i, y_i\}_{i=1}^{49000}$. We use this data set for training three surrogate modes; a Decision Tree, a Random Forest, and a Neural Network using scikit-learn (Pedregosa et al., 2011). For the Decision Tree and the Random Forest, we used the default values of the hyperparameters. The Neural Network had 3 layers with 150 neurons, the activation function was tanh, the learning rate was equal to 10^{-4}, and the regularization parameter α was set equal to 0.01. Note that in this case, we do not train Gaussian Processes since the dataset has 49 000 data points and Gaussian Processes have cubic complexity on the number of samples.

Once the surrogate models were trained, we generate 100 random disturbances that change simultaneously the demands and the inlet concentration as in the previous section. These disturbances are different than the disturbances generated in the previous section. We solve the optimization problem for every disturbance by initializing the Generalized Benders Decomposition algorithm using the number of cuts suggested by optimizing the different surrogate models. The total CPU time for the different disturbances is presented in Fig. 4 and the solution time statistics in Table 3. From the results, we observe that the average total CPU time without the addition of cuts (No cuts) is 14.7 s. The selection of the optimal number of cuts to add to the master problem leads to a 70% reduction in CPU time. From the three surrogate models, the Neural Network shows the maximum improvement in total CPU time, although the average reduction is similar for all surrogate models. Furthermore, the minimum reduction is positive for all models, indicating that the solution time with the proposed initialization is lower than the original implementation of the algorithm without cuts. From Fig. 4 we observe that the variance in the solution time for the proposed approach is small. This indicates that the surrogate models extract information from the feature space when predicting the solution time. An initialization strategy where a random number of cuts is added will have a higher variance than the proposed approach since the solution time can vary between the minimum solution time and the solution time without any cuts. Finally, the time presented in Table 3 is the total time required to determine the optimal number of cuts and solve the problem. For case study considered, the time to determine the optimal number of cuts is in the order of 10^{-2} s for the decision tree and the neural network and in the order of 10^{-1} s for the random forest. Thus, the process of determining the number of cuts and adding them is significantly smaller than the solution time.

6. Conclusions and discussion

The repeated solution of large-scale decision-making problems arises frequently in the operation of process systems. The efficient solution of such problems with monolithic optimization algorithms can be challenging, especially in online settings. Although decomposition-based solution methods have been widely used to solve large-scale optimization problems, their off-the-shelf implementation is nontrivial. In this work, we proposed a ML-based approach for optimally initializing cutting plane-based decomposition-based solution algorithms, such as Generalized Benders. We use active learning to guide the generation of labeled data for learning a surrogate model that predicts the solution time of Generalized Benders Decomposition for a given problem and the initial set of cuts added in the master problem. The proposed approach is applied to a case study of mixed-integer economic model predictive control. The numerical results show that the optimal initialization of

Table 1
Computational time for the proposed approach for different surrogate models. NC refers to solving the problem without the addition of cuts in the first iteration.

Solution statistics	Initialization strategy					
	NC	GP-AL	GP	NN	RF	DT
Aver. CPU time	13.7	4.31	6.22	7.67	6.34	9.11
Aver. red.	–	66.5	53.44	43.52	51.61	33.64
Aver. fold red.	–	3.33	2.49	2.18	2.38	1.75
Max. red. (%)	–	81.3	81.41	79.57	81.96	77.55
Min. red. (%)	–	0.09	−2.66	−0.02	−26.47	−18.82
Max fold red.	–	5.35	5.38	4.48	5.54	4.45
Min fold red.	–	1.00	0.97	0.99	0.79	0.84

Table 2
Distribution of the demand.

Product	Nominal value	Distribution (Uniform)	
		Low	High
1	600	−100	100
2	550	−15	15
3	600	−30	30
4	1200	−20	20
5	2000	−400	400

Table 3
Computational time for the proposed approach for different surrogate models trained via supervised learning.

Solution statistics	Initialization strategy			
	No cuts	Neural networks	Random forests	Decision trees
Average CPU time (s)	14.7	3.63	3.78	3.74
Average reduction (%)	–	71.71	70.50	70.53
Average fold reduction	–	4.23	4.01	4.10
Max. reduction (%)	–	84.85	83.88	86.40
Min. reduction (%)	–	25.66	17.30	21.13
Max fold reduction	–	6.60	6.20	7.35
Min fold reduction	–	1.34	1.20	1.26

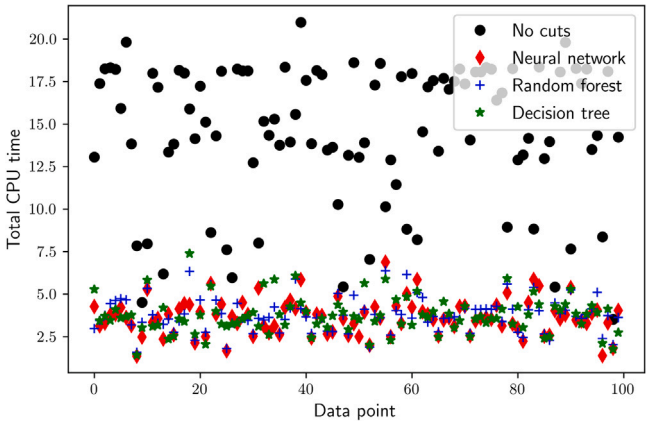

Fig. 4. Solution time of the proposed approach with different surrogate models for 49 000 training data points.

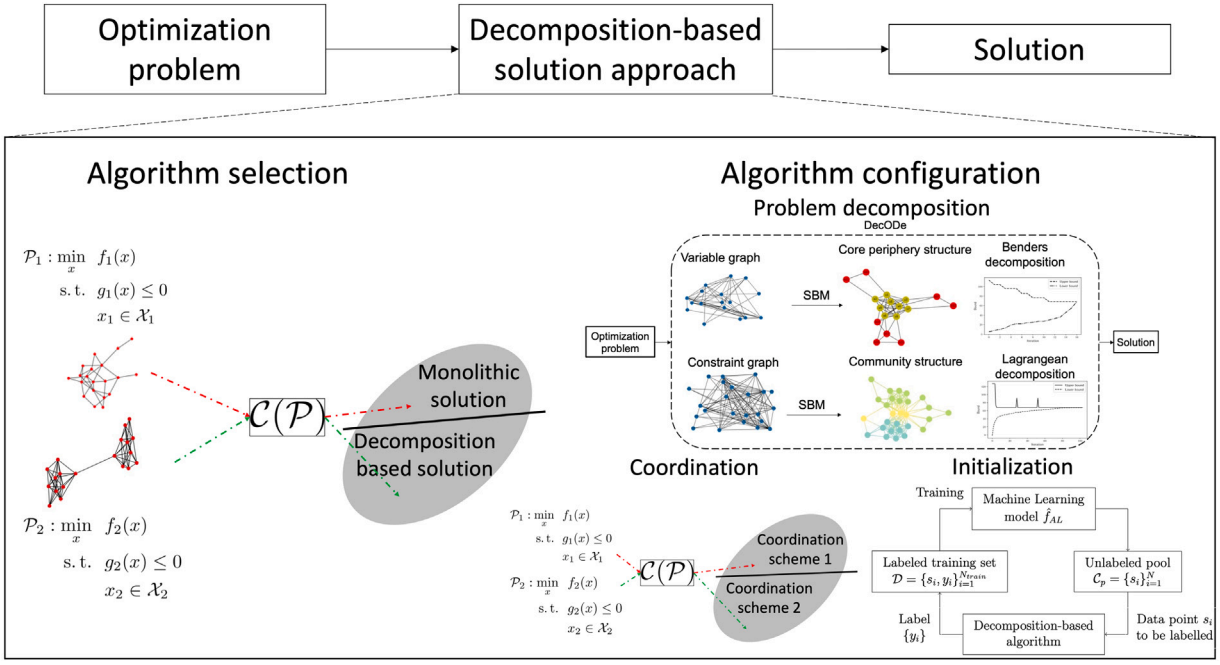

Fig. 5. Framework for automated decomposition-based solution algorithm selection and configuration via artificial intelligence and network science.

the algorithm can significantly reduce the solution time up to 70 % and the computation of the optimal number of cuts using the learn surrogate model does not incur additional computational cost. These results highlight the ability of active learning to guide the data generation process for cases where obtaining the solution time is computationally expensive.

In general, the solution of an optimization problem with decomposition-based solution methods poses four questions: (1) Whether a decomposition-based method should be selected over a monolithic one, (2) how to decompose the optimization problem, (3) which coordination scheme should be used, and (4) how to initialize the algorithm. The results presented in this paper and in Mitrai and Daoutidis (2023b), in tandem with our previous work on learning the underlying structure of an optimization problem (Allman et al., 2019; Mitrai et al., 2022; Mitrai and Daoutidis, 2021), show that artificial intelligence tools in combination with network science can be used to create an automated framework for decomposition-based solution algorithm selection and configuration.

The overall framework is presented in Fig. 5 and has two parts. The first focuses on algorithm selection, where given an optimization problem graph classification techniques can be used to determine whether, and potentially which, decomposition-based solution algorithm should be used. In the second part, the configuration of the selected decomposition-based solution algorithm is considered. The configuration has three aspects, problem decomposition, coordination, and initialization. For the problem decomposition, structure detection algorithms can be implemented to learn the underlying structure of the problem and use it as the basis for the application of the decomposition-based solution algorithm using DecODe (Mitrai et al., 2022).

Regarding the coordination, although it was not considered in this paper or in Mitrai and Daoutidis (2023b), it is a critical aspect of decomposition-based solution. For hierarchical decomposition-based methods, the coordination is done via cuts and the different coordination schemes correspond to different cut generation and management techniques. For distributed algorithms, the coordination considers different methods to update the dual variables or the Lagrangian multipliers. Consider for example the case of Lagrangian decomposition, where the dual variables can be updated using subgradient, cutting plane, bundle, and trust region methods (Conejo et al., 2006). The choice of which coordination scheme is the best, especially for mixed-integer nonlinear programming problems is not obvious. ML tools such as classification and regression, possibly coupled with geometric deep learning, might aid the selection of the coordination scheme as presented in Fig. 5. Finally, for the initialization, supervised learning can be used to learn surrogate models for the computational performance of the algorithm which can subsequently be used to optimally initialize the algorithm.

This framework, and any ML-based approach, requires data for learning the parameters of the surrogate models used in the different tasks. Although multiple libraries of optimization problems exist (MINLPLib, 2018; NLP, 2022; Gleixner et al., 2021), these are unlabeled data since for a given problem the solution time for a given algorithm and a given configuration is not known. Building a library where not only the optimization problem but also the solution time and possibly other information regarding the solution is stored would significantly reduce the required time for building such ML-based approaches for decomposition-based solution algorithms. Finally, the computational effort required to obtain the labels can be reduced either via active learning or semi-supervised learning (Balestriero et al., 2023).

CRediT authorship contribution statement

Ilias Mitrai: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Prodromos Daoutidis:** Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

Financial support from NSF-CBET (award number 2313289) is gratefully acknowledged.

References

Allman, A., Tang, W., Daoutidis, P., 2019. DeCODe: a community-based algorithm for generating high-quality decompositions of optimization problems. Optim. Eng. 20 (4), 1067–1084.

Angluin, D., 1988. Queries and concept learning. Mach. Learn. 2 (4), 319–342.

Balcan, M.-F., Dick, T., Sandholm, T., Vitercik, E., 2018. Learning to branch. In: International Conference on Machine Learning. PMLR, pp. 344–353.

Balestriero, R., Ibrahim, M., Sobal, V., Morcos, A., Shekhar, S., Goldstein, T., Bordes, F., Bardes, A., Mialon, G., Tian, Y., et al., 2023. A cookbook of self-supervised learning. arXiv preprint arXiv:2304.12210.

Baltean-Lugojan, R., Bonami, P., Misener, R., Tramontani, A., 2019. Scoring positive semidefinite cutting planes for quadratic optimization via trained neural networks. preprint: http://www.optimization-online.org/DB_HTML/2018/11/6943.html.

Basso, S., Ceselli, A., 2023. A data driven Dantzig–Wolfe decomposition framework. Math. Prog. Comput. 15 (1), 153–194.

Basso, S., Ceselli, A., Tettamanzi, A., 2020. Random sampling and machine learning to understand good decompositions. Ann. Oper. Res. 284, 501–526.

Benders, J.F., 1962. Partitioning procedures for solving mixed-variables programming problems. Numer. Math. 4 (1), 238–252.

Bengio, Y., Lodi, A., Prouvost, A., 2021. Machine learning for combinatorial optimization: a methodological tour d'horizon. European J. Oper. Res. 290 (2), 405–421.

Bertsimas, D., Stellato, B., 2021. The voice of optimization. Mach. Learn. 110 (2), 249–277.

Bertsimas, D., Stellato, B., 2022. Online mixed-integer optimization in milliseconds. INFORMS J. Comput..

Biagioni, D., Graf, P., Zhang, X., Zamzam, A.S., Baker, K., King, J., 2020. Learning-accelerated ADMM for distributed DC optimal power flow. IEEE Control Syst. Lett. 6, 1–6.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al., 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. 3 (1), 1–122.

Chen, T., Chen, X., Chen, W., Heaton, H., Liu, J., Wang, Z., Yin, W., 2021. Learning to optimize: A primer and a benchmark. arXiv preprint arXiv:2103.12828.

Chen, W., Shao, Z., Wang, K., Chen, X., Biegler, L.T., 2011. Random sampling-based automatic parameter tuning for nonlinear programming solvers. Ind. Eng. Chem. Res. 50 (7), 3907–3918.

Chi, C., Aboussalah, A.M., Khalil, E.B., Wang, J., Sherkat-Masoumi, Z., 2022. A deep reinforcement learning framework for column generation. arXiv preprint arXiv:2206.02568.

Cohn, D., Atlas, L., Ladner, R., 1994. Improving generalization with active learning. Mach. Learn. 15, 201–221.

Conejo, A.J., Castillo, E., Minguez, R., Garcia-Bertrand, R., 2006. Decomposition Techniques in Mathematical Programming: Engineering and Science Applications. Springer.

Crainic, T.G., Rei, W., Hewitt, M., Maggioni, F., 2016. Partial Benders Decomposition Strategies for Two-Stage Stochastic Integer Programs, vol. 37, CIRRELT.

Di Liberto, G., Kadioglu, S., Leo, K., Malitsky, Y., 2016. Dash: Dynamic approach for switching heuristics. European J. Oper. Res. 248 (3), 943–953.

Ding, J.-Y., Zhang, C., Shen, L., Li, S., Wang, B., Xu, Y., Song, L., 2020. Accelerating primal solution findings for mixed integer programs based on solution prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34, pp. 1452–1459.

Eggensperger, K., Lindauer, M., Hutter, F., 2019. Pitfalls and best practices in algorithm configuration. J. Artificial Intelligence Res. 64, 861–893.

Geoffrion, A.M., 1972. Generalized benders decomposition. J. Optim. Theory Appl. 10, 237–260.

Gleixner, A., Hendel, G., Gamrath, G., Achterberg, T., Bastubbe, M., Berthold, T., Christophel, P., Jarck, K., Koch, T., Linderoth, J., et al., 2021. MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library. Math. Prog. Comput. 13 (3), 443–490.

Guignard, M., Kim, S., 1987. Lagrangean decomposition: A model yielding stronger Lagrangean bounds. Math. Prog. 39 (2), 215–228.

Gupta, P., Gasse, M., Khalil, E., Mudigonda, P., Lodi, A., Bengio, Y., 2020. Hybrid models for learning to branch. Adv. Neural Inf. Process. Syst. 33, 18087–18097.

Gupta, P., Khalil, E.B., Chetélat, D., Gasse, M., Bengio, Y., Lodi, A., Kumar, M.P., 2022. Lookback for learning to branch. arXiv preprint arXiv:2206.14987.

Gurobi Optimization, LLC, 2021. Gurobi optimizer reference manual. URL https://www.gurobi.com. Accessed: 2021-08-31.

Hart, W.E., Laird, C.D., Watson, J.-P., Woodruff, D.L., Hackebeil, G.A., Nicholson, B.L., Siirola, J.D., 2017. second ed. Pyomo–Optimization Modeling in Python, vol. 67, Springer Science & Business Media.

Huang, Z., Wang, K., Liu, F., Zhen, H.-L., Zhang, W., Yuan, M., Hao, J., Yu, Y., Wang, J., 2022. Learning to select cuts for efficient mixed-integer programming. Pattern Recognit. 123, 108353.

Hutter, F., Hoos, H.H., Leyton-Brown, K., 2010. Automated configuration of mixed integer programming solvers. In: International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming. Springer, pp. 186–202.

Hutter, F., Hoos, H.H., Leyton-Brown, K., 2011. Sequential model-based optimization for general algorithm configuration. In: International Conference on Learning and Intelligent Optimization. Springer, pp. 507–523.

Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T., 2009. ParamILS: an automatic algorithm configuration framework. J. Artificial Intelligence Res. 36, 267–306.

Hutter, F., Xu, L., Hoos, H.H., Leyton-Brown, K., 2014. Algorithm runtime prediction: Methods & evaluation. Artificial Intelligence 206, 79–111.

Iommazzo, G., d'Ambrosio, C., Frangioni, A., Liberti, L., 2020. Learning to configure mathematical programming solvers by mathematical programming. In: International Conference on Learning and Intelligent Optimization. Springer, pp. 377–389.

Jalving, J., Cao, Y., Zavala, V.M., 2018. Graph-based modeling and simulation of complex systems. arXiv preprint arXiv:1812.04983.

Jia, H., Shen, S., 2021. Benders cut classification via support vector machines for solving two-stage stochastic programs. INFORMS J. Comput. 3 (3), 278–297.

Khalil, E., Le Bodic, P., Song, L., Nemhauser, G., Dilkina, B., 2016. Learning to branch in mixed integer programming. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 30.

Larsen, E., Frejinger, E., Gendron, B., Lodi, A., 2023. Fast continuous and integer L-shaped heuristics through supervised learning. INFORMS J. Comput..

Lee, M., Ma, N., Yu, G., Dai, H., 2020. Accelerating generalized benders decomposition for wireless resource allocation. IEEE Trans. Wirel. Commun. 20 (2), 1233–1247.

Lewis, D.D., Gale, W.A., 1995. A sequential algorithm for training text classifiers: Corrigendum and additional data. In: ACM SIGIR Forum, vol. 29, ACM New York, NY, USA, pp. 13–19.

Linderoth, J., Wright, S., 2003. Decomposition algorithms for stochastic programming on a computational grid. Comput. Optim. Appl. 24 (2), 207–250.

Liu, D., Fischetti, M., Lodi, A., 2022. Learning to search in local branching. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 36, pp. 3796–3803.

Liu, J., Ploskas, N., Sahinidis, N.V., 2019. Tuning BARON using derivative-free optimization algorithms. J. Global Optim. 74 (4), 611–637.

Lodi, A., Zarpellon, G., 2017. On learning and branching: a survey. Top 25 (2), 207–236.

Magnanti, T.L., Wong, R.T., 1981. Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. Oper. Res. 29 (3), 464–484.

Marousi, A., Kokossis, A., 2022. On the acceleration of global optimization algorithms by coupling cutting plane decomposition algorithms with machine learning and advanced data analytics. Comput. Chem. Eng. 163, 107820.

2018. MINLPLib: A library of mixed-integer and continuous nonlinear programming instances. URL http://www.minlplib.org/.

Misra, S., Roald, L., Ng, Y., 2022. Learning for constrained optimization: Identifying optimal active constraint sets. INFORMS J. Comput. 34 (1), 463–480.

Mitrai, I., Daoutidis, P., 2020. Decomposition of integrated scheduling and dynamic optimization problems using community detection. J. Process Control 90, 63–74.

Mitrai, I., Daoutidis, P., 2021. Efficient solution of enterprise-wide optimization problems using nested stochastic blockmodeling. Ind. Eng. Chem. Res. http://dx.doi.org/10.1021/acs.iecr.1c01570.

Mitrai, I., Daoutidis, P., 2022a. An adaptive multi-cut decomposition based algorithm for integrated closed loop scheduling and control. In: Computer Aided Chemical Engineering, vol. 49, Elsevier, pp. 475–480.

Mitrai, I., Daoutidis, P., 2022b. A multicut generalized benders decomposition approach for the integration of process operations and dynamic optimization for continuous systems. Comput. Chem. Eng. 107859.

Mitrai, I., Daoutidis, P., 2023a. Computationally efficient solution of mixed integer model predictive control problems via machine learning aided benders decomposition. arXiv preprint arXiv:2309.16508.

Mitrai, I., Daoutidis, P., 2023b. Taking the human out of decomposition-based optimization via artificial intelligence: Part I. Learning when to decompose. Comput. Chem. Eng. http://dx.doi.org/10.1016/j.compchemeng.2024.108688.

Mitrai, I., Tang, W., Daoutidis, P., 2022. Stochastic blockmodeling for learning the structure of optimization problems. AIChE J. 68 (6), e17415.

Morabit, M., Desaulniers, G., Lodi, A., 2021. Machine-learning–based column selection for column generation. Transp. Sci. 55 (4), 815–831.

2022. NLP and MINLP test problems. URL https://minlp.com/nlp-and-minlp-test-problems.

Pacqueau, R., Soumis, F., Hoang, L.-N., 2012. A Fast and Accurate Algorithm for Stochastic Integer Programming, Applied to Stochastic Shift Scheduling. Groupe d'Études et de Recherche en Analyse des Décisions.

Paulus, M.B., Zarpellon, G., Krause, A., Charlin, L., Maddison, C., 2022. Learning to cut by looking ahead: Cutting plane selection via imitation learning. In: International Conference on Machine Learning. PMLR, pp. 17584–17600.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in python. J. Mach. Learn. Res. 12, 2825–2830.

Risbeck, M.J., Maravelias, C.T., Rawlings, J.B., 2019. Unification of closed-loop scheduling and control: State-space formulations, terminal constraints, and nominal theoretical properties. Comput. Chem. Eng. 129, 106496.

Ruszczyński, A., Świetanowski, A., 1997. Accelerating the regularized decomposition method for two stage stochastic linear problems. European J. Oper. Res. 101 (2), 328–342.

Saharidis, G.K., Boile, M., Theofanis, S., 2011. Initialization of the benders master problem using valid inequalities applied to fixed-charge network problems. Expert Syst. Appl. 38 (6), 6627–6636.

Saharidis, G.K., Ierapetritou, M.G., 2010. Improving benders decomposition using maximum feasible subsystem (MFS) cut generation strategy. Comput. Chem. Eng. 34 (8), 1237–1245.

Schede, E., Brandt, J., Tornede, A., Wever, M., Bengs, V., Hüllermeier, E., Tierney, K., 2022. A survey of methods for automated algorithm configuration. arXiv preprint arXiv:2202.01651.

Settles, B., 2009. Active learning literature survey.

Settles, B., Craven, M., Ray, S., 2007. Multiple-instance active learning. Adv. Neural Inf. Process. Syst. 20.

Seung, H.S., Opper, M., Sompolinsky, H., 1992. Query by committee. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. pp. 287–294.

Smith-Miles, K., Lopes, L., 2012. Measuring instance difficulty for combinatorial optimization problems. Comput. Oper. Res. 39 (5), 875–889.

Su, L., Tang, L., Grossmann, I.E., 2015. Computational strategies for improved MINLP algorithms. Comput. Chem. Eng. 75, 40–48.

Tang, Y., Agrawal, S., Faenza, Y., 2020. Reinforcement learning for integer programming: Learning to cut. In: International Conference on Machine Learning. PMLR, pp. 9367–9376.

Tang, W., Allman, A., Pourkargar, D.B., Daoutidis, P., 2018. Optimal decomposition for distributed optimization in nonlinear model predictive control through community detection. Comput. Chem. Eng. 111, 43–54.

Van Roy, T.J., 1983. Cross decomposition for mixed integer programming. Math. Prog. 25, 46–63.

Varelmann, T., Otashu, J.I., Seo, K., Lipow, A.W., Mitsos, A., Baldea, M., 2022. A decoupling strategy for protecting sensitive process information in cooperative optimization of power flow. AIChE J. 68 (1), e17429.

Wächter, A., Biegler, L.T., 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. 106 (1), 25–57.

Williams, C.K., Rasmussen, C.E., 2006. Gaussian Processes for Machine Learning, vol. 2, MIT Press, Cambridge, MA.

You, F., Grossmann, I.E., 2013. Multicut benders decomposition algorithm for process supply chain planning under uncertainty. Ann. Oper. Res. 210 (1), 191–211.

Zeng, S., Kody, A., Kim, Y., Kim, K., Molzahn, D.K., 2022. A reinforcement learning approach to parameter selection for distributed optimal power flow. Electr. Power Syst. Res. 212, 108546.