LEARNING TO INITIALIZE GENERALIZED BENDERS DECOMPOSITION VIA ACTIVE LEARNING

Ilias Mitrai [a] and Prodromos Daoutidis [a,1]

[a] Department of Chemical Engineering and Materials Science, University of Minnesota, Minneapolis, MN 55414

Abstract

The repeated solution of large scale mathematical optimization problems arises frequently in the operation of process systems. Generalized Benders Decomposition (GBD) can be used to reduce the computational time, however its implementation is nontrivial. In this work we aim at developing a machine learning approach to accelerate the solution of large scale optimization problems by optimally initializing GBD. We train surrogate models to estimate the computational time for different number of cuts for given values of the parameters of the optimization problem. The surrogate models are used to find the optimal number of cuts that should be added in the master problem in the first iteration such that the computational time is minimized. Additionally, we improve the computational performance of the proposed approach using active learning. We apply this approach to a case study on the integration of production scheduling and dynamic optimization for continuous systems and analyze the computational performance.

Keywords

Algorithm configuration, Decomposition based solution algorithm, Machine learning, Active learning

Introduction

Generalized Benders Decomposition (GBD) has been widely used to solve large scale optimization problems that arise in the design and operation of process systems (Conejo et al. (2006); Grossmann (2005); Rahmaniani et al. (2017); Chu and You (2013); Nie et al. (2012); Mitrai and Daoutidis (2021)). In this approach, the original problem is decomposed into two problems; a master problem which contains the discrete variables and a subproblem which contains the continuous variables and depends on the values of the variables of the master problem. The coordination between the master problem and the subproblem is done via the addition of Benders optimality/feasibility cuts (Geoffrion (1972)). Since the algorithm alternates between the solution of the master problem and the subproblem, the computational performance depends on the complexity of the master problem and subproblem, the number of infeasible subproblems that must be solved and the quality of cuts (in terms of tightness) that are generated during the solution.

We can argue that 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)), initialization using valid inequalities (Saharidis et al. (2011)), multicut implementation (You and Grossmann (2013)), cut generation and management (Magnanti and Wong (1981); Su et al. (2015); Saharidis and Ierapetritou (2010); Pacqueau et al. (2012)), and regularization/stabilization of the master problem (Ruszczyński and Świetanowski (1997); Linderoth and Wright (2003)). In the second approach machine learning is used to substitute the heuristic or computationally expensive steps of the algorithm (Bengio et al. (2021)). Typical example is the addition of cuts for the solution of two stage stochastic optimization problems (Jia and Shen (2021); Lee et al. (2020)) and the approximation of the solution of the subproblem (Larsen et al. (2022)).

In this work, we will focus on the *initialization* of GBD regarding the number of cuts that should be added in the first iteration of the algorithm. We will consider the case where an optimization problem must be solved repeatedly given updated values of the parameters and the task is to decide the optimal number of cuts to add in the first iteration such that the CPU time is minimized. Since the effect of the number of cuts on the solution time is not known a-priori, we propose a supervised learning approach to estimate the effect of cuts on the CPU time for given values of the parameters. Next, we show that the efficiency of the proposed approach can be improved using active learning. We apply the proposed ap-

[1] Corresponding author. Email: daout001@umn.edu.

proach to a case study on the integration of scheduling and dynamic optimization. The results show that the proposed approach can lead to significant reduction in CPU time (up to 70%).

Generalized Benders Decomposition

We will assume that the following problem (denoted as $\mathcal{P}$) must be solved:

$$\begin{aligned}\mathcal{P}(\mathbf{p}) := \underset{z,x,y}{\text{minimize}} \quad & f_1(z,x;p_1)+f_2(x,y;p_4)\\ \text{subject to} \quad & g_1(z,x;p_2)\le 0\\ & g_2(x,y;p_5)\le 0\\ & h_1(z,x;p_3)=0\\ & h_2(x,y;p_6)=0\\ & z\in\mathbb{Z}^{n_z}, x\in\mathbb{R}^{n_x^c}\times\mathbb{Z}^{n_x^d}, y\in\mathbb{R}^{n_y},\end{aligned} \tag{1}$$

where $\mathbf{p}=[p_1...p_6]^\top$ 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 $\mathbf{p}$. The complicating variables are the x variables and we can solve the problem using GBD. The subproblem is

$$\begin{aligned}\mathcal{S}(x,p_4,p_5,p_6) := \underset{\bar{x},y}{\text{minimize}} \quad & f_2(\bar{x},y;p_4)\\ \text{subject to} \quad & g_2(\bar{x},y;p_5)\le 0\\ & h_2(\bar{x},y;p_6)=0\\ & \bar{x}=x \;:\; \lambda\\ & \bar{x}\in\mathbb{R}^{n_x^c+n_x^d}, y\in\mathbb{R}^{n_y}.\end{aligned} \tag{2}$$

The solution of this problem depends on the values of the complicating variables x. The master problem is:

$$\begin{aligned}\mathcal{M}(\cdot) := \underset{z,x,\eta}{\text{minimize}} \quad & f_1(z,x;p_1)+\eta\\ \text{subject to} \quad & g_1(z,x;p_2)\le 0\\ & h_1(z,x;p_3)=0\\ & \eta\ge f_2(\bar{x}^l,\bar{y}^l;p_4)-\lambda^l(x-\bar{x}^l)\;\forall l\in\mathcal{L}\\ & z\in\mathbb{Z}^{n_z}, x\in\mathbb{R}^{n_x^c}\times\mathbb{Z}^{n_x^d},\end{aligned} \tag{3}$$

where $\mathcal{M}(\cdot)=\mathcal{M}(p_1,p_2,p_3,\mathcal{L})$, 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.

Algorithm 1 Generalized Benders Decomposition

Require: Optimization problem
1: Set $UB=\infty, LB=-\infty$
2: Set tolerance and optimality gap (tol)
3: Initialize the algorithm
4: **while** $(UB-LB)/LB \ge$ tol/100 **do**
5: Solve the master problem (Eq. 3) and obtain LB, x
6: Solve the subproblem (Eq. 2)
7: Add Benders cut
8: Update upper bound $UB = f_1(x) + f_2(x,y)$
9: **end while**
10: **return** Upper, lower bound and variable values

Problem definition and proposed solution approach

We will assume that problem $\mathcal{P}$ must be solved repeatedly given new values of the parameters $\mathbf{p}$. The problem that we will address is the following: *Given the new value of the parameters $\mathbf{p}$, add the optimal number of cuts in the master problem in the first iteration of the algorithm such that the CPU time is minimized.*

The number of cuts added in the fist iteration can be considered as a hyper-parameter of GBD. Therefore this is a hyper-parameter tuning problem, which is known in the literature as the algorithm configuration problem (Hutter et al. (2011)). In general the solution of the above problem is challenging. The effect of a cut on the solution time is not known a priory, the complicating variables can be both discrete and continuous, and all the parameters of the problem might change. For simplicity we will assume that the parameters of the subproblem do not change, all the complicating variables are continuous and 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.

Given these assumptions we can compute the cuts once and add them in the master problem every time the problem must be solved with updated values of the parameters. However, the addition of a large number of cuts will increase the complexity of the master problem, which might increase the computational time. Therefore for given values of the parameters $\mathbf{p}$ we must determine the optimal number of cuts to be added by solving the following problem:

$$n^*_{cuts}\in\arg\min_{n_{cuts}} f(n_{cuts},\phi(\mathcal{P}(\mathbf{p}))), \tag{4}$$

where f captures the effect of the number of cuts (for given parameters in the problem) on the solution time and ϕ represent features of the problem which affect the solution time. Since function f is not known, we can approximate it with a surrogate model. The steps used to approximate f are presented in Algorithm 2.

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

Require: Optimization problem, Number of data points N_{data}, number of discretization points N_{cuts}, upper and lower bounds for x variables (x^{lb}, x^{ub}), $\check{p}=[p_4,p_5,p_6]$
1: $i=1$
2: **while** i $\le N_{data}$ **do**
3: Generate parameters $p_1,p_2,p_3 \rightarrow p_i=[p_1,p_2,p_3,\check{p}]$
4: **for** $j=2{:}N_{cuts}$ **do**
5: Solve problem $\mathcal{P}(p_i)$ using j cuts for each x
6: Obtain CPU time y_j
7: Obtain features of the problem $s_j=(\phi(\mathcal{P}(p_i)),j)$
8: append data $\{s_j,y_j\}$
9: **end for**
10: i = i+1
11: **end while**
12: Using the data $\{s_i,y_i\}_{i=1}^{(N_{data}N_{cuts})}$ learn the parameters of a surrogate model $\hat{f}$
13: **return** Approximate function $\hat{f}$

This algorithm returns a surrogate $\hat{f}$ which estimates the CPU time for given values of the features of the problem and number of cuts. Given this function, we can find the optimal number of cuts by solving the following problem:

$$n^*_{cuts} = \arg \min_{n \in \{2, N_{cuts}\}} \hat{f}(n, \phi(\mathcal{P}(\mathbf{p}))). \tag{5}$$

The steps that are followed to initialize Generalized Benders decomposition based on the learnt surrogate model are presented in Algorithm 3.

Algorithm 3 Regression based initalization of Generalized Benders decomposition based on Algorithm 2

Require: Surrogate model $\hat{f}$, optimization problem, values of parameters $\mathbf{p}$

1: Compute the features of the problem $\phi(\mathbf{p})$
2: Determine the optimal number of cuts n_{cuts} per transition to be added $n^*_{cuts} = \arg\min_{n \in \{2, N_{cuts}\}} \hat{f}(\phi(\mathbf{p}), n)$
3: Add n^*_{cuts} cuts in master problem
4: Solve problem using Algorithm 1
5: **return** Problem solution

Application to closed loop scheduling and control

In this section we will consider the problem of integration of scheduling and dynamic optimization. We will use the optimization problem proposed in Mitrai and Daoutidis (2022a). We will assume that the system is an isothermal CSTR where 5 products must be manufactured, and the time horizon is 24 hours discretized into 5 slots. The dynamic behavior of the reactor is described by the following differential equation

$$\frac{dc(t)}{dt} = \frac{F(t)}{V}(c_{in} - c(t)) - kc(t)^3, \tag{6}$$

where c (mol/L) is the concentration in the reactor, F (L) is the inlet flowrate, C_{in} (mol/L) is the inlet concentration and $V = 5000\ L, k = 2\ L^2/(hr\ mol^2)$ is the volume and reaction constant respectively.

At each time point different disturbances, i.e. change in inlet concentration or change in demand, can affect the system. Given a disturbance the integrated problem must be resolved given the updated process information. In this case two types of transitions can be performed; either between products or between an intermediate state and a product. The integrated problem has the following general form:

$$\begin{aligned} \max \quad & \Phi_1(w) - \sum_{ijk} Z_{ijk} f^{ijk}_{dyn}(\omega_{ijk}, \theta_{ijk}) - \sum_i \hat{Z}_i f^i_{dyn}(\hat{\omega}_i, \hat{\theta}_i) \\ \text{s.t.} \quad & g_{sched}(w, \theta_{ijk}, \hat{\theta}_i) \le 0 \\ & g_{dyn}(\theta_{ijk}, \omega_{ijk}) \le 0 \ \forall i, j, k \\ & \hat{g}_{dyn}(\hat{\theta}_i, \hat{\omega}_{ijk}) \le 0 \ \forall i \\ & Z_{ijk} \in \{0,1\}, \hat{Z}_i \in \{0,1\}, \end{aligned} \tag{7}$$

where w are scheduling variables, ω_{ijk} are variables associated with the dynamic behavior of the system for a transition from product i to product j in slot k, $\hat{\omega}_i$ are variables associated with the dynamic behavior of the system for a transition from the intermediate state to product i, θ_{ijk} is the transition time for a transition from product i to product j in slot k, and $\hat{\theta}_i$ is the transition time from the intermediate state to product i. The variable Z_{ijk} is equal to 1 if a transition occurs between product i and j in slot k and zero otherwise and the variable $\hat{Z}_i$ is equal to 1 if a transition occurs between a intermediate state and product i and zero otherwise. The objective function has three terms; the first represents the profit, the second the transition cost for a transition between products, and the last term represents the transition cost for a transition from a intermediate state to the products. This problem is a large scale MINLP whose monolithic solution is intractable.

If we fix the scheduling variables 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

$$\begin{aligned} \min \quad & f^{ijk}_{dyn}(\omega_{ijk}, \tilde{\theta}_{ijk}) \\ \text{s.t.} \quad & g_{dyn}(\tilde{\theta}_{ijk}, \omega_{ijk}) \le 0 \\ & \tilde{\theta}_{ijk} = \theta_{ijk} \quad : \lambda_{ijk}. \end{aligned} \tag{8}$$

where λ_{ijk} is the Lagrange 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

$$\begin{aligned} \min \quad & f^{ijk}_{dyn}(\hat{\omega}_i, \check{\theta}_i) \\ \text{s.t.} \quad & g_{dyn}(\check{\theta}_i, \hat{\omega}_i) \le 0 \\ & \check{\theta}_i = \hat{\theta}_i \quad : \hat{\lambda}_i. \end{aligned} \tag{9}$$

The value functions $\phi_{ijk}, \hat{\phi}_i$ can be approximated with Benders cuts given by the following equations (Geoffrion (1972))

$$\begin{aligned} \eta_{ijk} \ge & \phi^l_{ijk}(\bar{\theta}^l_{ijk}) - \lambda^l_{ijk}(\theta_{ijk} - \bar{\theta}^l_{ijk}) \\ \hat{\eta}_i \ge & \hat{\phi}_i(\bar{\theta}_i) - \hat{\lambda}^l_i(\hat{\theta}_i - \bar{\hat{\theta}}^l_i), \end{aligned} \tag{10}$$

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

$$\begin{aligned} \max \quad & \Phi_1(w) - \sum_{ijk} Z_{ijk}\eta_{ijk} - \sum_i \hat{Z}_i \hat{\eta}_i \\ \text{s.t.} \quad & g_{sched}(w, \theta_{ijk}, \hat{\theta}_i) \le 0 \\ & \eta_{ijk} \ge \phi^l_{ijk}(\bar{\theta}^l_{ijk}) - \lambda^l_{ijk}(\theta_{ijk} - \bar{\theta}^l_{ijk}) \ \forall i, j, k, l \\ & \hat{\eta}_i \ge \hat{\phi}_i(\bar{\theta}_i) - \hat{\lambda}^l_i(\hat{\theta}_i - \bar{\hat{\theta}}^l_i) \ \forall i, l. \end{aligned} \tag{11}$$

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.

Training the surrogate models

We assume that at a random time point t the demand of all the products and the inlet concentration of the reactor change simultaneously. The statistics of the demand are presented in Table 1. We assume that the inlet concentration follows a uniform distribution with low value of 0.8 and high value 1.2. The values of the other parameters can be found in Mitrai and Daoutidis (2022a). Given the updated process information, we solve the problem for different number of cuts and obtain the CPU time. The features of the problem in this case are the time point t, the concentration in the reactor c, the inlet flowrate Q, 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, and number of cuts added. Overall the features s_i for data point i are:

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

and the label y_i is the CPU time. For the training, we discretize the domain of the transition times using $N_{cuts} = 50$ and we generate 1000 random disturbances. Overall we obtain 49000 data-points $\{s_i, y_i\}_{i=1}^{49000}$.

Table 1: 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

We train three surrogate models; 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 parameters. 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.

Initialization of GBD based on the surrogate models

Once the surrogate models were trained, we generated 100 random disturbances that change simultaneously the demands and the inlet concentration. We solve the problem initializing the hybrid multicut GBD (Mitrai and Daoutidis (2022b)) using Algorithm 3. The total CPU time for the different disturbances is presented in Fig. 1 and the solution time statistics in Table 2. From the results we observe that the average total CPU time without the addition of cuts (No cuts) is 14.7 seconds. The proposed approach leads to 70% reduction in CPU time. From the three surrogate models, the neural network shows the maximum improvement in total CPU time. Furthermore, for this case study, the time to determine the optimal number of cuts is in the order of 10^{-2} seconds for the decision tree and the neural network and in the order of 10^{-1} seconds for the random forest (see Fig. 2).

Table 2: Computational time for the proposed approach for different surrogate models

Solution statistics	Initalization strategy			
	No cuts	Neural networks	Random forests	Decision trees
Average CPU time (sec)	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

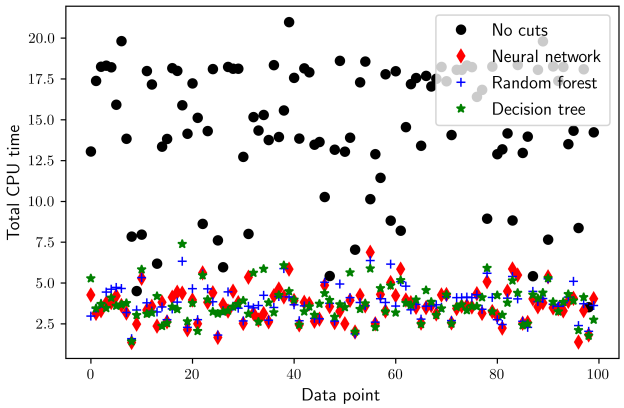

Figure 1: Solution time of the proposed approach with different surrogate models.

Learning to initialize via active learning

Motivation

The main limitation of the above approach is the computational time required to generate the training data to approximate the CPU time for different values of parameters and cuts. For the previous case study, the computational time to obtain the data was 100 hours. For problems with larger number of product the computational time of this step will be a bottleneck. To resolve this we use active learning (Settles (2009)).

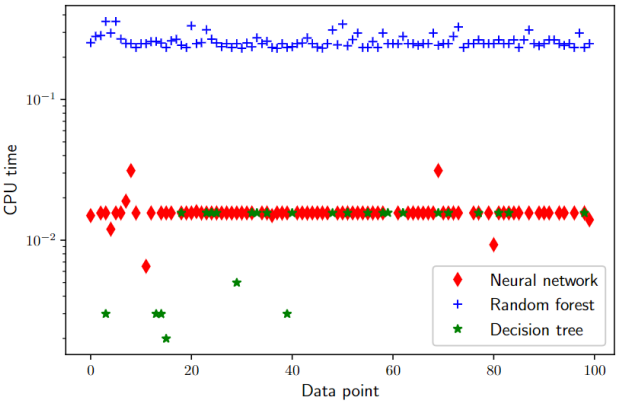

Figure 2: CPU time to determine the optimal number of cuts to add for the different surrogate models.

Algorithm 4 Active learning

Require: Number of evaluations N, initial dataset $C = \{s, y\}$, Pool of labels C_p, testing set C_{test}, Surrogate model $\check{f}$

Train surrogate model on 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

$$s = \arg\max_{s \in C_p} \sigma(s)$$

  Evaluate the label y for s
  Append data $\{s, y\}$ in set $C = C \cup \{s, y\}$
  Remove data-point s from pool $C_p = C_p \setminus \{s, y\}$
  Train surrogate model using set C
end while
return Surrogate model $\check{f}$

Active learning approach

Active learning (AL) is commonly used in supervised machine learning tasks where the features of the data are known but obtaining their label can be costly or time consuming. In the case study considered in this section we can generate a large number of features (pool of features), i.e. random disturbances and number of cuts, but we do not know the labels, i.e. CPU time, for these features. The active learning strategy will determine for which features from the pool we should evaluate the CPU time. This strategy is known as pool based active learning. The surrogate model is a Gaussian Process with Matern kernel (Williams and Rasmussen (2006)) and we will use uncertainty based sampling where we evaluate the CPU time of the data-point whose uncertainty σ is maximum. The main steps for the active learning strategy are presented in Algorithm 4.

For the application of active learning, first we generate random disturbances and obtain the features of the problem s_i. These features form the pool $\mathcal{C}_p = \{s_i\}_{i=1}^{N_{pool}}$ of data points. Next we generate a small number of data points $(N_{initial})$ and evaluate the CPU time. This is the initial training set $\mathcal{C} = \{x_i, y_i\}_{i=1}^{N_{initial}}$. We also generate N_{test} data points and evaluate the CPU and these are the testing data $\mathcal{C}_{test} = \{x_l, y_l\}_{l=1}^{N_{test}}$. Given the initial training set $\mathcal{C}$ and the pool $\mathcal{C}_p$ sets, in each iteration the active learning algorithm will provide the data point that we should evaluate.

The initial size of the training set is $N_{initial} = 10$, the size of the testing set is $N_{test} = 50$, the size of the pool is $N_{pool} = 49000$, and we allow $N = 100$ evaluations. The computational time for obtaining the 100 labels is 726 seconds. We compare the proposed active learning approach, denoted as GP-AL, with 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 normalized mean squared error (NMSE) for the different approaches is presented in Table 3. From these results we observe that the active learning approach leads to a surrogate model with lower NMSE.

Table 3: Comparison of normalized mean squared error for the different approaches

Method	NMSE
Gaussian Process - AL	5.94
Gaussian Process	16.01
Neural Network	30.84
Decision Tree	29.46
Random Forest	21.48

Finally we evaluate the active learning approach on optimally initializing GBD. We compare the Gaussian process obtained via active learning with the surrogate models trained via random sampling of 110 data points from the pool. The average solution time for the different surrogate models is presented in Table 4. From these results we observe that the surrogate model obtained via active learning outperforms the other surrogates. Specifically, the active learning approach leads on average to 66.5% reduction in CPU time, whereas the Gaussian process, neural network, random forest and decision tree lead to 53%, 43%, 51% and 33% 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. This is not true for the other surrogate models, for example using the random forest as surrogate can lead up to 26% increase in CPU time compared to the solution of the problem without the addition of cuts.

Table 4: 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	Initalization 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

Conclusions

Generalized Benders Decomposition has been widely used to solve large scale optimization problems. However the implementation of the algorithm is nontrivial. In this work we considered the optimal initialization of GBD for the repeated solution of problems where the parameters of the subproblem do not change. First we proposed a supervised learning approach to learn a surrogate model that estimates the CPU for given parameters of the problem and number of cuts. Although this approach can lead to 70% reduction in CPU time, the generation of the dataset to train the surrogate model can be a bottleneck. To overcome this obstacle we proposed the application of active learning. The results show that active learning can be used to efficiently train a surrogate model to estimate the effect of cuts on the solution time.

Acknowledgement

This work was supported by National Science Foundation (NSF- CBET, award number 1926303) and a Doctoral Dis-

sertation Fellowship (DDF) of University of Minnesota for Ilias Mitrai.

References

Bengio, Y., A. Lodi, and A. Prouvost (2021). Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research 290*(2), 405–421.

Chu, Y. and F. You (2013). Integrated scheduling and dynamic optimization of complex batch processes with general network structure using a generalized benders decomposition approach. *Industrial & Engineering Chemistry Research 52*(23), 7867–7885.

Conejo, A. J., E. Castillo, R. Minguez, and R. Garcia-Bertrand (2006). *Decomposition techniques in mathematical programming: engineering and science applications*. Springer.

Crainic, T. G., W. Rei, M. Hewitt, and F. Maggioni (2016). *Partial Benders decomposition strategies for two-stage stochastic integer programs*, Volume 37. CIRRELT.

Geoffrion, A. M. (1972). Generalized benders decomposition. *Journal of optimization theory and applications 10*, 237–260.

Grossmann, I. (2005). Enterprise-wide optimization: A new frontier in process systems engineering. *AIChE Journal 51*(7), 1846–1857.

Hutter, F., H. H. Hoos, and K. Leyton-Brown (2011). Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pp. 507–523. Springer.

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

Larsen, E., E. Frejinger, B. Gendron, and A. Lodi (2022). Fast continuous and integer l-shaped heuristics through supervised learning. *arXiv preprint arXiv:2205.00897*.

Lee, M., N. Ma, G. Yu, and H. Dai (2020). Accelerating generalized benders decomposition for wireless resource allocation. *IEEE Transactions on Wireless Communications 20*(2), 1233–1247.

Linderoth, J. and S. Wright (2003). Decomposition algorithms for stochastic programming on a computational grid. *Computational Optimization and Applications 24*(2), 207–250.

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

Mitrai, I. and P. Daoutidis (2021). Efficient solution of enterprise-wide optimization problems using nested stochastic blockmodeling. *Ind. Eng. Chem. Res.*.

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

Mitrai, I. and P. Daoutidis (2022b). A multicut generalized benders decomposition approach for the integration of process operations and dynamic optimization for continuous systems. *Computers & Chemical Engineering*, 107859.

Nie, Y., L. T. Biegler, and J. M. Wassick (2012). Integrated scheduling and dynamic optimization of batch processes using state equipment networks. *AIChE Journal 58*(11), 3416–3432.

Pacqueau, R., F. Soumis, and L.-N. Hoang (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.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12*, 2825–2830.

Rahmaniani, R., T. G. Crainic, M. Gendreau, and W. Rei (2017). The benders decomposition algorithm: A literature review. *European Journal of Operational Research 259*(3), 801–817.

Ruszczyński, A. and A. Świetanowski (1997). Accelerating the regularized decomposition method for two stage stochastic linear problems. *European Journal of Operational Research 101*(2), 328–342.

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

Saharidis, G. K. and M. G. Ierapetritou (2010). Improving benders decomposition using maximum feasible subsystem (mfs) cut generation strategy. *Computers & chemical engineering 34*(8), 1237–1245.

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

Su, L., L. Tang, and I. E. Grossmann (2015). Computational strategies for improved minlp algorithms. *Computers & Chemical Engineering 75*, 40–48.

Williams, C. K. and C. E. Rasmussen (2006). *Gaussian processes for machine learning*, Volume 2. MIT press Cambridge, MA.

You, F. and I. E. Grossmann (2013). Multicut benders decomposition algorithm for process supply chain planning under uncertainty. *Annals of Operations Research 210*(1), 191–211.