Regression Optimization for System-level Production Control

Dzung T. Phan[1], **Lam M. Nguyen**[1], **Pavankumar Murali**[1],
Nhan H. Pham[2], **Hongsheng Liu**[2], **Jayant R. Kalagnanam**[1]
[1]IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA
[2]The University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, USA
phandu@us.ibm.com, LamNguyen.MLTD@ibm.com, pavanm@us.ibm.com,
nhanph@live.unc.edu, hsliuustc@gmail.com, jayant@us.ibm.com

***Abstract*— We propose a novel generalized prediction-optimization framework to optimize set point controls for a network of processes in a production plant, wherein a regression model is used to capture the physical representation of each process's behavior and the relationship between its inputs and outputs. We introduce a nonlinear optimization problem to model the optimal set-point problem. For piece-wise linear regressors, we reformulate the problem into a mixed-integer linear program. For highly nonlinear models such as deep neural networks, we propose a decomposition primal-dual algorithm for solving it. Using a real-world use case of oil sands processing, we show the benefit of our approach by the ability to efficiently identify a set of feasible control variables, while giving a high production output.**

I. Introduction

Process industries such as those related to crude oil production and refining, steel or aluminum manufacturing, cement manufacturing etc. frequently consist of many complex processes. Within each process, a byproduct from an upstream process undergoes a chemical and/or a physical transformation and is transported to a downstream process. Set points that control the performance of each process, measured in terms of throughput or quality, may themselves be dependent on the performance of an upstream process, as well as external factors such as ambient temperature, mineral content in the raw material etc. To maintain consistency in the overall performance, plant managers monitor and control set points. Processing plants instrumented with senors that capture fine grained process-related data, coupled with novel AI techniques could help aid this need.

A fundamental premise that has driven the infusion of AI technologies into process planning and management for heavy industries is the need for a Cognitive Advisor that can be built for each enterprise function. For example, a Cognitive Plant Advisor that is designed to consume historical and real-time data may be able to predict process performance and support a plant operator to (i) control process behavior over a time horizon by manipulating set points (advisory control) in order to navigate through normal operations with scheduled maintenance, and (ii) chart out a recovery plan in terms of set points to use to minimize disruption due to an unforeseen breakdown. This necessitates the development of a high fidelity regression model built using the available sensor data, and an optimization model, that operates in the neighborhood of set points with sufficient support, to optimize the desired business objective. Since the optimization uses a data-driven regression model as a representation of each plant, its run-time complexity, scalability and solution quality guarantees could depend on the nature of the model, e.g. whether we use a piece-wise linear model, nonlinear and non-convex deep neural network or a black-box ensemble model. This paper presents a novel prediction-optimization framework combining machine learning and optimization techniques for recommending optimal set points in a complex production plant.

The system-wide set point optimization spanning multiple plants and processes has received attention in the literature [1], [2], [3]. However, a common feature in prior art is the use of nonlinear physical first-principles models reflecting physical laws such as energy balance, heat transfer relations and mass balance for each process, coupled via material flow balance equations. Existing work has previously combined physical models and data-driven models [4]. A machine learning-based approach has been proposed in [5], but it lacks generality for applying in various manufacturing plants and prediction model selection.

The end-to-end learning methods in [6], [7], [8] are devoted to a single process, and they can handle specific classes of problems due to a need for solving a two-stage stochastic programming problem. In our case, since the sensor data for processes come from different temporal resolutions, these frameworks are unsuitable to build a similar end-to-end solution for our system-wide problem. There are existing works using surrogate statistical models for complex processes [9], but mainly as an experimental design to estimate a response surface model [10] or optimize a flowsheet structure [11], [12] or set points for a single process [13], [14]. In our setting, we rely only on historical plant sensor data, and do not have access to a plant simulator.

II. System-level Prediction Optimization

To fully understand the need for prediction-optimization models in real applications, we consider a real-world case study from the oil sands processing plant, whose graphical representation is given in Figure 1.

In this context, optimization involves devising set point trajectories over a time horizon that maximize Synthetic Crude Oil (SCO) production. Starting with mined ore, the first step is to extract bitumen as froth and store it in a

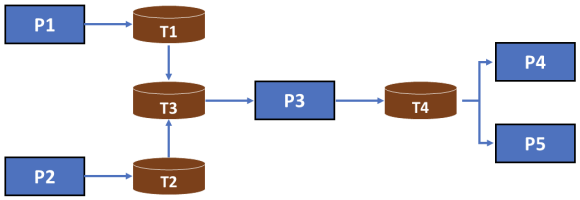

Fig. 1: An oil sands production plant

storage tank. Processes P1 and P2 are two parallel froth and diluted bitumen production plants. Three storage tanks T1, T2, and T3 store diluted bitumen. Diluted bitumen is then passed through another stage of extraction, P3, to produce low-quality SCO that gets stored in tank T4. This is further processed in processes P4 and P5. They are parallel upgraders that produce different grades of synthetic crude oil. We are given historical data covering a span of three years containing sensors measurements and production outputs for every processes. Under such scenarios, an optimization model could provide a set of recommendations on control set-points for the plant operator, such as mine tonnage rates and upgrading feed rates, that optimize the SCO production.

Our system-wide prediction optimization problem for a production plant is devoted to maximizing the flow throughput of end products by seeking an optimal production schedule with operational constraints, e.g., maintaining the levels of intermediate products in storage inventories and economic targets. The use of machine learning to model complex process from data is in large part motivated by a desire to improve operations measure in terms of productivity, throughput, efficiency and/or resource utilization. In such a setting the pipelines for analysis need to include a machine learning step to build prediction models for every output followed by an optimization procedure to derive the optimal set points for the process.

A plant can be modeled using multiple lower-level process nodes that connect to make up the plant. In this section, we propose a mathematical model to capture the interactions between processes (e.g., processes P1, $\ldots$ P5 in Figure 1) and auxiliary sub-systems (e.g, tank inventories T1, $\ldots$, T4) when they operate at a stationary point. We model process flows for the system of processes as a directed multi-layer network of sub-systems as in Figure 2.

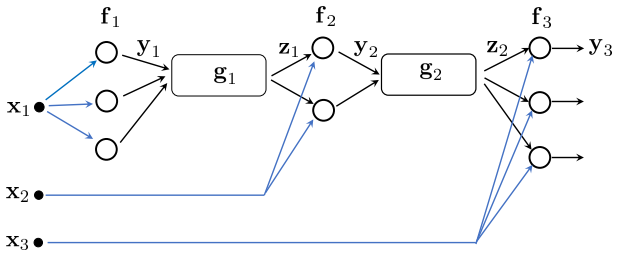

Fig. 2: Network representation for a system of plants

We view the plant as a hierarchy structure with L layers having upstream-downstream operation relations. A circular node in a layer of the network represents a process, where a regression function is built. A rectangular node represents all operational constraints such as mass balance equations, maintaining inventory levels, and limits on the adjustment of variables from the preceding time period. The relationships in these rectangular nodes are assumed to be linear.

We assume that each vector-valued regression function

$$f_l : (z_{l\ 1}; x_l) \in \mathbb{R}^{k_{l-1}+m_l} \ ! \ y_l \in \mathbb{R}^{n_l}$$

for processes positioned at the l-th layer has been learned from historical data. The trained model is available to produce a liable prediction output. Here we denote $x_l \in \mathbb{R}^{m_l}$ and $z_{l\ 1} \in \mathbb{R}^{k_{l-1}}$ by the controllable variables (e.g., set points) and uncontrollable variables (e.g., the inflow from the previous process) for f_l. The vector $y_l \in \mathbb{R}^{n_l}$ is the outflow. We assume $k_0 = 0$; that is, the decision variables for the first layer are only set points. Let be the objective function for the system of processes. The main goal is to find optimal set points x_l and flow rates (y_l, z_l) to maximize production or some target variable. We express the optimal control model as the following constrained optimization problem

$$\begin{aligned} \min_{X} \quad & (y_L) \\ \text{s.t.} \quad & y_1 = f_1(x_1); \\ & y_l = f_l(z_{l\ 1}; x_l); \ \forall l = 2, \ldots, L; \\ & A_l y_l + B_l z_l \quad b_l; \ \forall l = 1, \ldots, L \quad 1; \\ & \underline{x}_l \quad x_l \quad \overline{x}_l; \ \forall l = 1, \ldots, L; \\ & \underline{y}_l \quad y_l \quad \overline{y}_l; \ \forall l = 1, \ldots, L; \\ & \underline{z}_l \quad z_l \quad \overline{z}_l; \ \forall l = 1, \ldots, L \quad 1; \end{aligned} \tag{1}$$

where $X = (x_1, \ldots, x_L, y_1, \ldots, y_L, z_1, \ldots, z_{L\ 1})$ and the objective function (y_L) depends on the output at the last layer. Define $\underline{x}_l$ and $\overline{x}_l$ as upper and lower bounds on control set-points, $\underline{z}_l$ and $\overline{z}_l$ are limits for infows of processes. The operational limits for processes are $\underline{y}_l$ and $\overline{y}_l$.

The linear constraint $A_l y_l + B_l z_l \quad b_l$ captures mass balance and material flow rate equations between the l-th process layer, represented by the rectangular nodes g_i. An example for operational constraints in the linear constraint is the storage limit for T4 (i.e., $l = 2$) in Figure 1:

$$y_2 \quad z_2^1 \quad z_2^2 \quad s_2 = 0;$$

where s_2 is the tank level for T4; $\underline{s}_2$ and $\overline{s}_2$ are storage limits.

III. Mixed-integer Formulation for Piecewise Linear Regressors

The relationship between inputs and an output for an industrial plant is often complex, a non-linear prediction model should be used to capture the complexity rather than a linear model. It gives rise to a nonlinear regression f_l; consequently the problem (1) is a nonconvex program. Most nonlinear optimization algorithms devoted to a *nonconvex* program such as augmented Lagrangian method and interior-point methods do not guarantee to generate a *global* minimizer, they can get stuck at a *local* minimizer. In some industries such as oil and gas, a small improvement to solution quality

can have significant economic impact; for example it can be measured in billions of dollars per year for a 5% increase in control efficiency in the area. By using a mixed-integer linear program (MILP), we can obtain a *globally* optimal solution to the *nonconvex* (possibly nonsmooth) problem in a reasonable running time with the help of exact methods such as branch-and-bound and branch-and-cut for some classes of regression functions. We make use of recent advances in terms of computational strength and flexibility of state-of-the-art MILP solvers (e.g. CPLEX [15]), which leads to an optimally-tractable MILP formulation.

In this section, we show that for certain partition regressions based on piece-wise linear approach such as decision trees, multivariate adaptive regression splines (MARS), and decision lists [16], we can formulate Problem (1) as a mixed-integer linear program. The main idea is to use a mixed-integer linear representation for each regression function f_l. In the following, we present a MILP model for two popular models: oblique decision trees and MARS, but we can extend the technique for other piece-wise linear regressions.

Decision trees: An oblique decision tree regression $y = h(\mathbf{x})$ is characterized by sets of leaf nodes and branching nodes. Denote L by the set of leaf nodes, and B by the set of branching nodes. For each leaf node $\ell \in L$, a linear regression $r_\ell(\mathbf{x}) = \mathbf{w}_\ell^\top \mathbf{x} + c_\ell$ has been learned from the training data based on the points routed to the leaf node. A branching node $\ell \in B$ is represented by a hyperplane $\mathbf{a}_\ell^\top \mathbf{x} + b_\ell$, where if $\mathbf{a}_\ell^\top \mathbf{x} + b_\ell < 0$ then the point $\mathbf{x}$ will follow the left branch from the node, otherwise it splits into the right branch. Since the topology of the tree is fixed, for each feature vector $\mathbf{x}$, there is a unique path leading to a leaf node $\ell \in L$ from the root of the tree. Let $N_L(\ell)$ denote the ancestor nodes of ℓ where the left branch is followed on the path from the root to ℓ, and let $N_R(\ell)$ denote the set of right branch ancestors on the path. The binary variable $e_\ell \in \{0,1\}, \ell \in L$, indicates if $\mathbf{x}$ is assigned to leaf node ℓ then $e_\ell = 1$.

Assigning a data point $\mathbf{x}$ to a leaf node is modeled as

$$\sum_{\ell \in L} e_\ell = 1. \tag{2}$$

To determine the unique path routing to a leaf node, with the help of the indicator variable e_ℓ, the following constraints are enforced for modeling the splitting at branching nodes

$$\begin{aligned} &\mathbf{a}_k^\top \mathbf{x} + b_k < M(1 - e_\ell), \ \forall \ell \in L, k \in N_L(\ell) \\ &\mathbf{a}_k^\top \mathbf{x} + b_k \geq M(1 - e_\ell), \ \forall \ell \in L, k \in N_R(\ell), \end{aligned} \tag{3}$$

where M is the big-M parameter. The decision tree regression $y = h(\mathbf{x})$ can be represented as a mixed-integer bi-linear model

$$\begin{aligned} y = \ & \sum_{\ell \in L} e_\ell (\mathbf{w}_\ell^\top \mathbf{x} + c_\ell) \\ \text{s.t.} \ & \text{Eqs. (2), (3)} \\ & e_\ell \in \{0,1\}, \ \forall \ell \in L. \end{aligned} \tag{4}$$

We now linearize the bilinear term $e_\ell(\mathbf{w}_\ell^\top \mathbf{x} + c_\ell)$. Assume $y_\ell^L \leq \mathbf{w}_\ell^\top \mathbf{x} + c_\ell \leq y_\ell^U$ for some constants y_ℓ^L and y_ℓ^U. This assumption is reasonable because $\mathbf{x}$ is usually bounded; for example, the training data are normalized to the $[0, 1]$ range, and $\mathbf{w}_\ell, c_\ell$ are fixed model parameters. Then we have that $y_\ell = e_\ell(\mathbf{w}_\ell^\top \mathbf{x} + c_\ell)$ is equivalent to

$$\begin{aligned} & y_\ell^L e_\ell \leq y_\ell \leq y_\ell^U e_\ell \\ & \mathbf{w}_\ell^\top \mathbf{x} + c_\ell - y_\ell^U(1 - e_\ell) \leq y_\ell \leq \mathbf{w}_\ell^\top \mathbf{x} + c_\ell - y_\ell^L(1 - e_\ell). \end{aligned} \tag{5}$$

Hence a mixed-integer linear representation for the decision tree regression $y = h(\mathbf{x})$ is

$$\begin{aligned} h(\mathbf{x}) = \ & \sum_{\ell \in L} y_\ell \\ \text{s.t.} \ & \text{Eqs. (2), (3), (5)} \\ & e_\ell \in \{0,1\}, \ \forall \ell \in L. \end{aligned} \tag{6}$$

Multivariate adaptive regression splines: We consider the regression spine fitting of degree 1: $h(\mathbf{x}) = \beta_0 + \sum_{i=1}^N \beta_i h_i(\mathbf{x})$, where β_i are scalars and $h_i(\mathbf{x}) = \max\{\mathbf{w}_i^\top \mathbf{x} + c_i, 0\}$. A linear representation is

$$\begin{aligned} h(\mathbf{x}) = \ & \beta_0 + \sum_{i=1}^N \beta_i y_i \\ \text{s.t.} \ & y_i \geq \mathbf{w}_i^\top \mathbf{x} + c_i, i = 1, \ldots, N \\ & y_i \leq (\mathbf{w}_i^\top \mathbf{x} + c_i) + M_1 e_i, i = 1, \ldots, N \\ & y_i \leq M_2(1 - e_i), i = 1, \ldots, N \\ & e_i \in \{0,1\}, y_i \geq 0, i = 1, \ldots, N, \end{aligned} \tag{7}$$

where M_1 and M_2 are large numbers.

Obviously, when the linear-based representations (6) and (7) are plugged into the problem (1), we get a MILP formulation provided that Ω is linear, which can be efficiently solved to optimality by a MILP solver.

IV. Nonlinear Optimization Algorithm

In this section, we present a decomposition optimization algorithm for solving the problem (1) when the regression functions are highly nonlinear and cannot be linearized as in Section III. We assume that the gradient of f_l is readily available, but expensive to compute such as deep neural networks. We propose a two-level augmented Lagrangian algorithm for (1). We decompose the problem and solve a sequence of subproblems at process-level.

A. Two-level Augmented Lagrangian Method

We propose to solve (1) by the two-level augmented Lagrangian method when gradients of f_l are available. We treat *nonlinear* local constraints $\mathbf{y}_\ell = f_\ell(\cdot)$ and the *linear* coupling constraints $\mathbf{A}_l \mathbf{y}_l + \mathbf{B}_l \mathbf{z}_l \leq \mathbf{b}_l$ differently. The nonlinear ones are taken care by the augmented Lagrangian method (ALM) in the outer loop, while the linear ones are handled by ADMM in the inner level.

In order to apply a multi-block ADMM [17], we reformulate the inequality constraint into an equality constraint $\mathbf{A}_l \mathbf{y}_l + \mathbf{B}_l \mathbf{z}_l + \mathbf{v}_l - \mathbf{b}_l = 0, \mathbf{v}_l \geq 0$. However, as explained in Section 2.3 of [17], a multi-block ADMM has some intrinsic limitations, which prevents its application to a nonconvex problem. A crucial condition for convergence guarantee related to the images of coupling matrices is not satisfied in many settings, including our formulation (see Condition

1 in [17]). To overcome this issue, we add a slack variable u_l for the linear constraint

$$A_l y_l + B_l z_l + v_l \quad b_l + u_l = 0;$$
$$u_l = 0:$$

Now, we deal with both constraints $y_l = f_l(z_{l-1}; x_l)$ and $u_l = 0$ by the augmented Lagrangian method, and $A_l y_l + B_l z_l + v_l \quad b_l + u_l = 0$ by a multi-block ADMM. Specifically, we consider the following subproblem in ALM:

$$\begin{aligned}
\min \quad & F(\hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}, \hat{\lambda}^k, \hat{\nu}^k; \beta^k, \gamma^k) = \\
& \phi(\mathbf{y}_L) + (\lambda_1^k)^T(\mathbf{y}_1 - f_1(\mathbf{x}_1)) + \frac{\beta^k}{2}\|\mathbf{y}_1 - f_1(\mathbf{x}_1)\|^2 \\
& + \sum_{l=2}^{L}(\lambda_l^k)^T(\mathbf{y}_l - f_l(\mathbf{z}_{l-1}, \mathbf{x}_l)) + \frac{\beta^k}{2}\sum_{l=2}^{L}\|\mathbf{y}_l - f_l(\mathbf{z}_{l-1}, \mathbf{x}_l))\|^2 \\
& + \sum_{l=1}^{L-1}(\nu_l^k)^T\mathbf{u}_l + \frac{\gamma^k}{2}\sum_{l=1}^{L-1}\|\mathbf{u}_l\|^2 \\
\text{s.t.} \quad & \mathbf{A}_l\mathbf{y}_l + \mathbf{B}_l\mathbf{z}_l + \mathbf{v}_l - \mathbf{b}_l + \mathbf{u}_l = \mathbf{0}, \quad \forall\, l = 1, \ldots, L-1, \\
& \underline{\mathbf{x}}_l \le \mathbf{x}_l \le \overline{\mathbf{x}}_l,\ \underline{\mathbf{y}}_l \le \mathbf{y}_l \le \overline{\mathbf{y}}_l, \quad \forall\, l = 1, \ldots, L, \\
& \underline{\mathbf{z}}_l \le \mathbf{z}_l \le \overline{\mathbf{z}}_l,\ \mathbf{v}_l \ge \mathbf{0}, \quad \forall\, l = 1, \ldots, L-1.
\end{aligned} \tag{8}$$

The two-level augmented Lagrangian method is given in Algorithm 1, where $h(\hat{x}; \hat{y}; \hat{z}) = [y_1 \quad f_1(y_1); y_2 \quad f_2(z_1; y_2); \quad ; y_L \quad f_L(z_{L-1}; y_L)]$:

Algorithm 1 : Two-level ALM

Initialize starting points $(\hat{\mathbf{u}}^0, \hat{\mathbf{v}}^0, \hat{\mathbf{x}}^0, \hat{\mathbf{y}}^0, \hat{\mathbf{z}}^0)$; $\beta^1, \gamma^1 > 0$, $\omega \in [0, 1)$, $\tau > 1$; index $k \leftarrow 1$;
while some stopping criterion is not satisfied **do**
  /* Inner level problem */
  Solve (8) by a multi-block ADMM to get $(\hat{\mathbf{u}}^k, \hat{\mathbf{v}}^k, \hat{\mathbf{x}}^k, \hat{\mathbf{y}}^k, \hat{\mathbf{z}}^k, \mu^k)$ for $(\hat{\lambda}^k, \beta^k)$ and $(\hat{\nu}^k, \gamma^k)$;
  if $\|h(\hat{\mathbf{x}}^k, \hat{\mathbf{y}}^k, \hat{\mathbf{z}}^k)\| \le \omega\|h(\hat{\mathbf{x}}^{k-1}, \hat{\mathbf{y}}^{k-1}, \hat{\mathbf{z}}^{k-1})\|$ **then**
    $\hat{\lambda}^{k+1} \leftarrow \hat{\lambda}^k + \beta^k h(\hat{\mathbf{x}}^k, \hat{\mathbf{y}}^k, \hat{\mathbf{z}}^k)$, $\beta^{k+1} \leftarrow \beta^k$;
  else
    $\hat{\lambda}^{k+1} \leftarrow \hat{\lambda}^k$, $\beta^{k+1} \leftarrow \tau\beta^k$;
  end if
  if $\|\hat{\mathbf{u}}^k\| \le \omega\|\hat{\mathbf{u}}^{k-1}\|$ **then**
    $\hat{\nu}^{k+1} \leftarrow \hat{\nu}^k + \gamma^k\hat{\mathbf{u}}^k$, $\gamma^{k+1} \leftarrow \gamma^k$;
  else
    $\hat{\nu}^{k+1} \leftarrow \hat{\nu}^k$, $\gamma^{k+1} \leftarrow \tau\gamma^k$;
  end if
  $k \leftarrow k + 1$;
end while

B. Convergence Analysis

Next, we show a theoretical convergence result for the two-level ALM algorithm when combining with a multi-block ADMM for solving subproblems. To achieve the goal, we make use of the following assumptions.

Assumption 1: $F(\hat{u}; \hat{v}; \hat{x}; \hat{y}; \hat{z}; \hat{\lambda}^k; \hat{\nu}^k; \quad ^k; \quad ^k)$ is continuously differentiable.

Assumption 2: The inner level ADMM outputs an approximate solution $f(\hat{u}^k; \hat{v}^k; \hat{x}^k; \hat{y}^k; \hat{z}^k; \quad ^k)g$ to the following conditions

$$\begin{aligned}
& d_1^k \ 2 \ r f(\hat{x}^k; \hat{y}^k; \hat{z}^k) + N_X(\hat{x}^k; \hat{y}^k; \hat{z}^k) + C^T \quad ^k + \\
& \quad + r h(\hat{x}^k; \hat{y}^k; \hat{z}^k)^T(\hat{\lambda}^k + \quad ^k h(\hat{x}^k; \hat{y}^k; \hat{z}^k)) \\
& d_2^k \ 2 \quad ^k + N_X(\hat{v}^k); \\
& d_3^k = A\hat{y}^k + B\hat{z}^k + \hat{v}^k \quad b + \hat{u}^k; \\
& \hat{\lambda}^k + \quad ^k\hat{u}^k + \quad ^k = 0;
\end{aligned}$$

where $\lim_{k \to 1} kd_i^k k = 0$ for $i = 1; 2; 3$, $f(\hat{x}; \hat{y}; \hat{z}) = \quad (y_L)$, $C = 0 \quad A \quad B$ is the matrix associate with $(\hat{x}; \hat{y}; \hat{z})$ in the linear constraint. and are the box constraints for $(\hat{x}; \hat{y}; \hat{z})$ and $\hat{v}$ respectively.

We notice that for the last block of inner level ADMM, the linear constraint related to $\hat{u}$ is an identity matrix and the objective function is a convex and quadratic function. Under mild conditions on the functions f, h and the matrices A, B, the multi-block ADMM algorithm can converge (subsequently) to a stationary point of the subproblem (8) and therefore Assumption 2 can be satisfied. The readers can refer to [18] for details.

Theorem 1: Suppose that Assumptions 1-2 hold. Let $(\hat{u}^*; \hat{v}^*; \hat{x}^*; \hat{y}^*; \hat{z}^*)$ be a limit point of outer-level iterates $f(\hat{u}^k; \hat{v}^k; \hat{x}^k; \hat{y}^k; \hat{z}^k)g$ generated by Algorithm 1. Assume that the sequences $f\hat{\lambda}^k g$ and $f\hat{\nu}^k g$ are bounded. If $f(\hat{\lambda}^k; \quad ^k)g$ has a limit point $(\hat{\lambda}^*; \quad ^*)$ along the subsequence converging to $(\hat{u}^*; \hat{v}^*; \hat{x}^*; \hat{y}^*; \hat{z}^*)$. Then $(\hat{v}^*; \hat{x}^*; \hat{y}^*; \hat{z}^*; \hat{\lambda}^*; \quad ^*)$ is a stationary point.

Proof: By adding nonnegative slack variables $(v_1; \ldots; v_{L-1})$ in problem (1), the model can be transformed equivalently as

$$\begin{aligned}
\min \quad & (y_L) \\
\text{s.t.} \quad & y_1 = f_1(x_1); \\
& y_l = f_l(z_{l-1}; x_l); \quad 8l = 2; \ldots; L; \\
& A_l y_l + B_l z_l + v_l \quad b_l = 0; \quad 8l = 1; \ldots; L \quad 1; \\
& \underline{x}_l \quad x_l \quad \overline{x}_l; \quad 8l = 1; \ldots; L; \\
& \underline{z}_l \quad z_l \quad \overline{z}_l; v_l \quad 0; \quad 8l = 1; \ldots; L \quad 1:
\end{aligned} \tag{10}$$

We note that $(\hat{v}^*; \hat{x}^*; \hat{y}^*; \hat{z}^*; \hat{\lambda}^*; \hat{\lambda}^*)$ is a stationary point of problem (10) if it satisfies the following condition

$$\begin{aligned}
& 0 \ 2 \ r f(\hat{x}^*; \hat{y}^*; \hat{z}^*) + r h(\hat{x}^*; \hat{y}^*; \hat{z}^*)^T\hat{\lambda}^* + \\
& N_X(\hat{x}^*; \hat{y}^*; \hat{z}^*) + C^T \\
& 0 \ 2 \quad + N_X(\hat{v}^*) \\
& A\hat{y}^* + B\hat{z}^* + \hat{v}^* \quad b = 0; \\
& h(\hat{x}^*; \hat{y}^*; \hat{z}^*) = 0:
\end{aligned} \tag{11}$$

Under Assumption 2, we only need to show $\hat{u}^* = 0$ and $h(\hat{x}^*; \hat{y}^*; \hat{z}^*) = 0$ to complete primal feasibility.

For the first part, if ^{k} is bounded, then we have $\hat{u}^k \ ! \ 0$ so $\hat{u}^* = 0$. If ^{k} is unbounded, by taking limits on both sides of

$$\frac{\hat{\lambda}^k}{^k} + \hat{u}^k + \frac{^k}{^k} = 0; \tag{12}$$

we also have $\hat{u} = 0$, because $\hat{\lambda}^k$ is bounded and ρ^k converges to ∞.

For the second part, if ρ^k is bounded, then we have $h(\hat{x}^k, \hat{y}^k, \hat{z}^k) \to 0$ so $h(\hat{x}, \hat{y}, \hat{z}) = 0$. By contradiction argument, if ρ^k is unbounded and suppose that $\lim_{k \to \infty} \|h(\hat{x}^k, \hat{y}^k, \hat{z}^k)\| / \rho^k = \|h(\hat{x}, \hat{y}, \hat{z})\| / \rho^k > 0$. For any feasible solution $(\bar{x}, \bar{y}, \bar{z})$ to problem (1), it follows that

$$
\begin{aligned}
& L_\rho(\hat{u}^k, \hat{v}^k, \hat{x}^k, \hat{y}^k, \hat{z}^t, \rho^k) \\
&= f(\hat{x}^k, \hat{y}^k, \hat{z}^k) + \hat{\lambda}^T h(\hat{x}^k, \hat{y}^k, \hat{z}^k) \\
&+ \frac{\rho}{2} \|h(\hat{x}^k, \hat{y}^k, \hat{z}^k)\|^2 + (\hat{\lambda}^k)^T \hat{u}^k + \frac{\rho}{2} \|\hat{u}^k\|^2 \\
&+ \sum_{l=1}^{L} (\gamma_l^k)^T (A_l y_l^k + B_l z_l^k + v_l^k - b_l + u_l^k) \\
&+ \frac{\rho^k}{2} \sum_{l=1}^{L} \|A_l y_l^k + B_l z_l^k + v_l^k - b_l + u_l^k\|^2 \\
&\quad f(\bar{x}, \bar{y}, \bar{z}).
\end{aligned}
$$

Notice that $\hat{\lambda}^k$ and $\hat{\lambda}^k$ are bounded and $\lim_{k \to \infty} \hat{u}^k = 0$, $\lim_{k \to \infty} f(\hat{x}^k, \hat{y}^k, \hat{z}^k) \to f^* := \min_{(\bar{x}, \bar{y}, \bar{z}) \in X} f(\bar{x}, \bar{y}, \bar{z})$. Taking $k \to \infty$, the above inequality will fail to hold. As a result, $h(\hat{x}, \hat{y}, \hat{z}) = 0$, which finishes the proof. ■

V. Case Study: Oil Sands Production

We present a case study for a real system from a major Canadian oil-sands company. We use the proposed system-wide prediction-optimization framework to improve crude oil production under various asset capacity constraints. The details and the graphical representation have been described in Sect. 2. The objective of the production optimization is to maximize SCO production and maintain the levels of intermediate products in storage tanks under both normal operations and planned maintenance.

Data set. The dataset were collected from over $100,000$ sensors or tags that record measurements taken every five minutes and covering a historical span of three years. There are two types of tags: raw and calculated. Raw tags capture physical measurements such as flow rate, density, temperature, pressure, and vibration at various points in the plant. Calculated tags, such as laboratory results on product quality and chemical composition, are available every twelve hours. The dataset is ordered by time and then split into train, test and validation sets. Each split is time-wise contiguous, i.e. it contains data for set of consecutive time stamps. Based on expert input and feature extraction and engineering, covariates to be used for each regression model are identified.

Modeling. Regression models are used to represent the relationship between inflows and outflows for each of process P_i. We used the IBM AutoAI toolkit for automated machine learning to search for the right learning algorithm and optimize its hyperparameters [19]. We restrict out machine learning models to: MARS, decision tree, random forests, and fully connected deep neural networks.

A. Optimization Algorithm Performance

First, we compare *solution quality* and *running time* between our optimization algorithms (MILP-based models (6), (7), two-level ALM over the popular baseline methods (sequential least squares programming algorithm-`SLSQP` and conventional ALM [20]) for solving the problem (1) with different regression function scenarios.

Mixed-Integer Linear Programming: Table I compares MILP for MARS and the decision tree regression (TreeReg) models with the conventional ALM for solving (1). We run both algorithms with 200 different initializations, count the number of runs that achieve a *global* solution and compute the average running time. We use IBM CPLEX 12.9 to solve MILP problems. For solved cases, MILP always found the globally optimal solution with speed about 50x faster than ALM. Because of discontinuity for the regression tree model, ALM often got stuck at an infeasible point for TreeReg. To show solution quality for the MARS model, Figure 3 plots the percentage difference (from the global optimal value) $p = \frac{|v - v_{opt}|}{\max\{1, |v_{opt}|\}} \cdot 100\%$, where v_{opt} is the global optimal value, and v is the objective function value obtained from ALM or MILP. We can see that ALM converges to local minimums instead of the global minimum in 26 out of 200 cases with more than 3% away from the optimal value. The MILP formulation shows significant advantages in both solution qualities and computational time when dealing with piece-wise linear regressors.

	Optimality found (%)	Average time (s)
MARS-MILP	100%	0.14
MARS-ALM	87%	6.58
TreeReg-MILP	100%	0.12
TreeReg-ALM	65%	7.42

TABLE I: Comparison between MILP and ALM

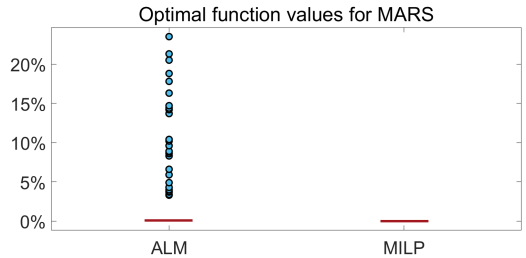

Fig. 3: Comparison between MILP and ALM

Nonlinear Models: Here, we use the two-level ALM in Sect. IV to solve (1) when the regression functions f_l are pre-trained feed forward neural networks. Figure 4 shows the comparison among SLSQP, two-level ALM, and conventional ALM. We randomly generate 200 initialization points and run all algorithms until convergence. Clearly, we can see that SLSQP performs poorly, while two-level ALM and ALM achieve essentially the same optimal value. The two-level ALM converged faster than ALM; that is, 8.6(s) versus 9.1(s) on average.