Time Series Segmentation to Discover Behavior Switching in Complex Physical Systems

Zheng Han†, Haifeng Chen‡, Tan Yan†, and Geoff Jiang‡

†Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015
‡Autonomic Management, NEC Laboratories America, Princeton, NJ 08540

Email: zhh210@lehigh.edu, {haifeng, yan, gfj}@nec-labs.com

Abstract—An accurate and automated identification of operational behavior switching is critical to the autonomic management of complex systems. In this paper, we collect sensor readings from those systems, which are treated as time series, and propose a solution to discover switching behaviors by inferring the relationship changes among massive time series. The method first learns a sequence of local relationship models that can best fit the time series data, and then combines the changes of local relationships to identify the system level behavior switching. In the local relationship modeling, we formulate the underlying switching identification as a segmentation problem, and propose a sophisticated optimization algorithm to accurately discover different segments in time series. In addition, we develop a hierarchical optimization strategy to further improve the efficiency of segmentation. To unveil the system level behavior switching, we present a density estimation and mode search algorithm to effectively aggregate the segmented local relationships so that the global switch points can be captured. Our method has been evaluated on both synthetic data and datasets from real systems. Experimental results demonstrate that it can successfully discover behavior switching in different systems.

Keywords—Physical systems, time series, segmentation, optimization, ADMM

I. INTRODUCTION

With the decreasing hardware cost and increasing demand for autonomic management, most complex physical systems such as nuclear power plants and manufacturing systems are now equipped with a large network of sensors distributed across different parts of the system. The readings of sensors are continuously collected, which can be regarded as time series, to reflect the operational status of the system. Effectively modeling and discovering patterns from the sensor data is important to improve system operations and many management tasks such as anomaly detection[17], [23], and capacity planning [11], etc.

One important observation from physical systems is that their operations usually switch between different states. For instance, manufacturing systems usually follow certain production workflow, which will automatically switch to a new process after completing the previous process. In many cases system operators usually do not know the exact time that system behavior switches. It is desired to automatically discover system behavior switching based on the time series data from sensors.

Discovering the behavior switching essentially requires to segment the time series indices into regions, each of which represents a specific operational behavior or state. The boundaries of the regions are called switch points and time series segmentation is sometimes also called switch points identification. Traditional approaches for segmenting time series are based on either dynamic programming [2], [8], [10] or heuristics [12], [13] techniques. While dynamic programming based methods need to pre-determine the number of segments before the segmentation, heuristics based approaches rely on simple heuristics which are not robust to general situations. Recently, due to the fast advances in optimization research, optimization based approaches [1], [16], [19], [20], [24] received a lot of attention in time series segmentation. Although that technique mainly segments single time series in current literature, it opens a new opportunity for us to effectively discover complex system dynamics.

In this paper, we leverage the power of optimization to identify behavior switching in complex systems. Unlike previous methods that segment time series based on their shapes, we design an optimization-based method to segment an ensemble of models learned from time series. Those models come from our observation that the system state can be described by a set of dependency relationships between sensor readings. For instance, the invariant approach in [11] used pairwise dependencies between time series to reveal system conditions. When the system switches its operational behavior, some of the dependencies will change accordingly. In this paper, we propose an effective method that first segments each individual dependency model between time series, and then aggregates local segmentation results to identify the system behavior switching.

It is a challenge to segment each dependency model between time series. While the switch points are discovered from the transitions between dependency models along time, learning that model sequence requires the positions of behavior switching. To resolve this chicken-and-egg problem, we formulate the segmentation as an optimization problem, and propose an integrated approach to identify the switch points and learn a sequence of models simultaneously. Although our method does not depend on any specific models, we use the invariant model in [11] to segment time series. The invariants represent a set of highly correlated relationship between pairs of time series in the system. Given a set of time series, we begin with a sampling-based mechanism to select the right system attributes—in the form of time series—to build pairwise relationships. After that, for each selected invariant pair, we design an objective function to identify its behavior switching along time, which covers the accuracy, complexity, as well as the switching frequency to reflect the reality of real system operations. We follow the framework of alternating direction method of multipliers (ADMM) [3] to optimize the objective function. A novel primal-dual active set technique is proposed to efficiently optimize the internal steps of ADMM.

In most physical systems, the behavior switching may
The rest of the paper is organized as follows: we discuss the background of invariant modeling in Section II. The detail of the proposed method and its enhancement are presented in Section III and Section IV respectively. We evaluate our method in Section V and present the related work in Section VI. Finally, Section VII concludes this paper.

II. BACKGROUND

The Invariant Model: To model the (local) behavior/state of a system, the invariant model [11] considers a pairwise relationship between two attributes \(x(t)\) and \(y(t)\), which employs the AutoRegressive relationship with eXogenous inputs (ARX) [18]:

\[
y(t) + a_1 y(t-1) + \cdots + a_n y(t-n) = b_0 x(t) + \cdots + b_m x(t-m) + \xi_t,
\]

where \([n, m]\) is the order of the model that determines how many previous steps are affecting the current output. Additionally, \(a_i\) and \(b_j\) are the coefficient parameters that reflect how strongly a previous step is affecting the current output. The noise \(\xi_t\) are i.i.d and Gaussian with \(\xi_t \sim N(0, \sigma^2)\).

Denote:
\[
\theta = [a_1, \cdots, a_n, b_0, \cdots, b_m]^T,
\]
\[
\varphi(t) = [-y(t-1), \ldots, -y(t-n), x(t), \ldots, x(t-m)]^T,
\]

then Equation (1) can be rewritten as:

\[
y(t) = \varphi(t)^T \theta + \xi_t.
\]

The parameter \(\theta\) in Model (2) can be learned by minimizing the least squares error. After that, a fitness score [11] is given to evaluate how well the learned model \(\hat{y}(t)\) fits the measurement data, and a pair is considered as an invariant only when it has sufficiently high fitness score.

System with Multiple States: For each invariant pair of a physical system, their relationship can be modeled with a certain parameter \(\theta\) in each state. When the system state changes, their relationship in the previous state no longer holds, which results in a change of \(\theta\) in the new state. If we analyze the change of \(\theta\) for all the invariants and aggregate those switch points, we can get a strong clue about the global behavior switching in the physical system.

However, we need to address the following challenges to archive this goal: (1) Local relationship building: Since the system behaviors are unknown, it is difficult to select the right attributes to build local relationships; (2) Model complexity: Even if the correct attributes are known in advance, the relationship modeling generally requires more parameters than traditional time series, and thus leads to higher model complexity; (3) Efficient solver: By its nature, the number of established local relationships (e.g., pairwise) usually is several orders of magnitude greater than the number of time series, which explicitly calls for an efficient optimization algorithm to quickly identify the solution; (4) Result aggregation: We need a robust way to assemble thousands of local relationship models learned from different system components to reveal the behavior switching of the whole system. We will address those issues in the next section.

III. BEHAVIOR SWITCHING IDENTIFICATION

In this section, we first provide a sampling based strategy in Section III-A to select correct invariant pairs given unknown system states, which addresses the local relationship building...
challenge mentioned in the last section. After selecting all invariant pairs, Section III-B models their relationships, and segments each individual invariant pair according to the change of the relationship. We formulate the segmentation problem to an objective function that takes into account the accuracy, model complexity, and the frequency of behavior switching during the operation, which overcomes the model complexity challenge. To solve the optimization problem, we design an efficient iterative optimization algorithm in Section III-C and carefully implement each internal step to achieve the large-scale problem solving capability, which solves the efficient solver challenge. After segmenting all the invariants, Section III-D proposes an algorithm to effectively aggregate all the segments to reveal the system level switch points.

A. Pair Selection via Sampling

Given the time series data, only those pairs of time series with high fitness score are regarded as invariants and selected for segmentation. When the system has multiple states, the computation of fitness requires the knowledge of boundary of state switching, since it may get a low fitness if the computation is based on the whole time series. In this section, we introduce a simple but effective technique to select invariants without knowing the switch points. Our idea is to sample multiple segments of the time series, and compute a model fitness score for each segment to determine whether to treat them as invariants. More specifically, we use \( \kappa \) and \( M \) to denote the sample frequency and sample size, respectively. We generate \( \kappa \) random positions \( \{r_1, r_2, \ldots, r_\kappa\} \) within the length of the time series. For each two time series, each time we pick a segment \( \{r_i, r_i+1, \ldots, r_i+M-1\} \) and compute the fitness score of Model (2). We do this for all the \( \kappa \) segments of the two time series and choose the highest score as the final score for this pair of time series. We do this for all pairs of time series, and select those with high scores (e.g., final score > 0.7) as invariants.

Since the switch points cannot be known beforehand, our sampling mechanism tries the best effort to examine different segments of time series and filter out the pairs with low fitness score. Note that we do not expect the full accuracy of this step due to the nature of sampling as well as existence of independent pairs where a sampled segment accidentally has high fitness score. As in Section III-D, we use the majority voting to identify the system behavior switching, which can tolerate random errors caused by sampling. Nevertheless the experiments in Section V-A demonstrate the high accuracy of our sampling based invariants selection.

B. Objective Function Formulation

After selecting invariant pairs, this section proposes a probabilistic model to describe the switching behavior of dependency relationship between each pair of time series. Given two time series as \( x(t) \) and \( y(t) \), each containing \( N \) data points, we define the observed data set as \( \mathcal{D} = \{x_1, \ldots, x_N, y_1, \ldots, y_N\} \). We extend the ARX in Equation (1) to capture the switching behavior of the invariant. Rather than assuming the model parameter \( \theta \) to be a constant parameter as in Section II, we treat \( \{\theta_i\} \) for \( t = 1, \ldots, N \) as a stochastic sequence that evolves in a piecewise constant fashion and come up with an objective function of \( \{\theta_i\} \) whose solution reflects the evolution of system states. By optimizing that objective function, we can segment the time series and hence discover the switch points.

For the convenience of illustration, in the following the notation of some variables are defined by appending a subscript of time index \( t \) to emphasize their fixation on \( t \). We define \( \mathcal{D}_t := \{y_0^t, \ldots, y_t^0, x_t^0, \ldots, x_t^m\} \) where \( y_t^0 = y(t) \), \( x_t^0 = x(t) \), and the remaining variables are obtained from observations at \( t - 1 \) by \( y_t = y_{t-1}^0 \), \( x_t = x_{t-1}^0 \).

Note that the parameters \( m \) and \( n \) come from the ARX model in Equation (1).

Figure 1 gives the probabilistic relationship between \( \mathcal{D}_t \) and \( \theta_t \). Note that we use dash lines to represent deterministic transitions, and solid lines to represent probabilistic transitions where the relationship involves additional randomness. When behavior switching occurs, say at \( t \), \( \theta_t \) will be different with \( \theta_{t-1} \). Behavior switching essentially characterizes the situation that the underlying ARX relationship changes its parameter values. In Figure 1, the optional \( \lambda \) is a hyper parameter imposed on \( \theta_t \) to control the model complexity and we will describe its role after defining the distribution \( P(\theta_t|\lambda) \).

The probability that the sequence \( \theta_1, \ldots, \theta_N \) is observed from the data set \( \mathcal{D} \), i.e., the posterior probability, satisfies

\[
P(\theta_1, \ldots, \theta_N|\mathcal{D}) \propto P(\mathcal{D}|\theta_1, \ldots, \theta_N)P(\theta_1, \ldots, \theta_N). \quad (3)
\]

The first component of the right side of Equation (3) is the likelihood of the observation given the sequence, and the second part is the prior probability of the sequence, which carries our prior knowledge on \( \theta_1, \ldots, \theta_N \) such as the piecewise constant constraint. In the following we describe the expression of those two components in detail.

Likelihood function: By Figure 1, the likelihood function is expressed as

\[
P(\mathcal{D}|\theta_1, \ldots, \theta_N) = P(\mathcal{D}_1, \ldots, \mathcal{D}_N|\theta_1, \ldots, \theta_N) = P(\mathcal{D}_1|\theta_1) \prod_{t=2}^{N} P(\mathcal{D}_t|\theta_t, \mathcal{D}_{t-1}). \quad (4)
\]

According to Equation (1), given observations \( \mathcal{D}_{t-1} \), only \( y_t^0 \) is a random variable with distribution

\[
P(\mathcal{D}_t|\theta_t, \mathcal{D}_{t-1}) = P(y_t^0|\theta_t, \mathcal{D}_{t-1}, x_t^0) \sim N(\alpha^{T}_t \theta_t, \sigma^2),
\]

for \( t \geq 1 \), where \( \alpha_t = [-y_{t-1}^0, \ldots, -y_{t-n}^0, x_t^0]^T \). Note that \( \alpha_t \) is just a different representation of the vector \( \varphi(t) \) in Equation (2). As a result, the likelihood function (4) is represented as

\[
P(\mathcal{D}|\theta_1, \ldots, \theta_N) = \left( \frac{1}{2\pi\sigma^2} \right)^{N/2} \prod_{t=1}^{N} \exp \left\{ \frac{(y_t - \alpha^T_t \theta_t)^2}{\sigma^2} \right\}. \quad (5)
\]
Prior probability: As shown in Figure 1, the probability distribution of $\theta_t$ at time $t$ depends on its previous value $\theta_{t-1}$ as well as the parameter $\lambda$ that influences the model complexity via $\lambda$. We first formulate the prior distribution $P(\theta_1, \ldots, \theta_N)$ from a posterior of two distributions: the joint likelihood $P(\theta_t | \theta_{t-1})$, and the probability related to model complexity, i.e., the sparsity of each $\theta_t$, which is represented as $\prod_{t=1}^{N}P(\theta_t | \lambda)$. As a result, we obtain

$$P(\theta_1, \ldots, \theta_N) \propto P(\theta_1 | \lambda) \prod_{t=2}^{N} P(\theta_t | \theta_{t-1})P(\theta_t | \lambda).$$

(Note that $P(\theta_1)$ from the joint likelihood didn’t show up in the right side of equation (6) because it is treated as a constant in the model. There are now two types of probabilities on the right side of (6): $P(\theta_t | \theta_{t-1})$ and $P(\theta_t | \lambda)$. They embody different expectations on $\theta_t$ values as we will define immediately.)

Model complexity: The probability $P(\theta_t | \lambda)$ is optional and related to the model complexity. Specifically, let $\lambda > 0$ and we define $P(\theta_t | \lambda) = \lambda \exp\{-\lambda ||\theta_t||_1\}$ which encourages sparsity on $\theta_t$. Instead of controlling the model complexity explicitly via the parameter $\lambda$, one can also simply set $\lambda = 0$ which still seem to perform well in practice.

Switching frequency: As aforementioned, since we expect $\theta_t$ to be piecewise constant to reflect the state change in the real system operation, the probability density function $P(\theta_t | \theta_{t-1})$ should encourage similarities between consecutive $\theta$s. We use the exponential family to model such expectation by simply setting $P(\theta_t | \theta_{t-1}) = \exp\{-||\theta_t - \theta_{t-1}||\}$ but remark that the distribution function could also be parameterized without much change on our derivation results.

In summary, we have the prior probability in Equation (6), which reflects model complexity and switch frequency requirements, represented as

$$P(\theta_1, \ldots, \theta_N) \propto \lambda^N \prod_{i=1}^{N} \exp \{-\lambda ||\theta_i||_1\} \prod_{t=2}^{N} \exp \{-||\theta_t - \theta_{t-1}||\}.$$  

(7)

The Objective Function:

After formulating all the aforementioned requirements, we now assemble them together to derive the optimization model for identifying invariant relationship changes. The objective attempts to maximize logarithm posterior probability of $\theta_1, \ldots, \theta_N$ given the observation $D$, as defined in Equation (3). By plugging Equation (5) and (7) to the logarithm of Equation (3) and ignore the constant additive items, we have

$$\min_{\theta_1, \ldots, \theta_N} \frac{1}{2} \sum_{t=1}^{N} \left( y_t - \alpha_t^T \theta_t \right)^2 + \lambda_1 \sum_{t=1}^{N} ||\theta_t||_1 + \lambda_2 \sum_{t=2}^{N} ||\theta_t - \theta_{t-1}||,$$

where $\lambda_1 = \frac{\sigma^2}{T}$ and $\lambda_2 = \frac{\sigma^2}{T}$ can be seen as the regularization parameters.

The solution of Model (III-B) is an estimate of the real dynamics of $\{\theta_t\}$, $||\theta_t - \theta_{t-1}||$ measures the statistical significance of the change of $\theta_t$. By the definition of $P(\theta_t | \theta_{t-1})$, we conclude that for $||\theta_t - \theta_{t-1}|| \geq \epsilon$ where $\epsilon > 0$, we can claim with confidence of over $1 - e^{-\epsilon}$ that $\theta_t$ is different with $\theta_{t-1}$, i.e., $t$ is a switch point.

C. The Optimization Algorithm

In this section, we present an efficient and scalable optimization algorithm to solve Problem (III-B). We follow the framework of Alternating Direction Method of Multipliers (ADMM) [3], a popular iterative framework that is well-suited for large-scale convex optimization. We first describe the ADMM framework and its typical formulation. Then, we reformulate Problem (III-B) to an optimization problem with linear equality constraints involving two separable classes of variables, such that it can be solved under the setting of ADMM. After that, we state our algorithm that iteratively solves the problem, which includes a careful implementation to get the three generic steps of each ADMM iteration efficiently computed. Finally, we remark on possibilities to speed up the algorithm.

Adaption for ADMM:

ADMM is a framework to design efficient optimization algorithms, which achieves large-scale optimization capability by iteratively solving the problem in a decentralized way. A typical ADMM problem formulation is shown as follows:

$$\min_{x_1,x_2} \left\{ f(x_1) + g(x_2) : A_1 x_1 = A_2 x_2 \right\},$$  

(8)

where $f, g$ are convex functions, and $A_1, A_2$ are linear coefficient matrices. To solve such a problem, ADMM iteratively updates $x_1$ and $x_2$ in an alternating manner that steers $(x_1, x_2)$ progressively closer to the optimal solution.

Reformulation: We now adapt Problem (III-B) to the form of (8) so that ADMM can be applied on it. We first denote $A := [\alpha_1, \ldots, \alpha_N]^T \in \mathbb{R}^{N \times s}$, and $\theta := [\theta_1^T, \ldots, \theta_N^T]^T \in \mathbb{R}^{Ns}$, and introduce the auxiliary variable $\beta$ as

$$\beta := [\theta_2^T, \ldots, \theta_N^T]^T - [\theta_1^T, \ldots, \theta_{N-1}^T]^T \in \mathbb{R}^{(N-1)s}.$$  

(9)

Notice that $\beta$ is the block first-order difference of $\theta_t$ with respect to time $t$, which represents the parameter change between invariant models. The nonzero elements indicate the locations of the switch points thus are what we want to obtain. Moreover, later in this section, we show such block-wise formation of $\beta$ allows it to be computed in a distributed way, which greatly improves the scalability of the algorithm. By introducing $\beta$, we rewrite the original Problem (III-B) and formulate it compactly as

$$\min_{\theta, \beta} \left\{ \frac{1}{2} ||y - A\theta||^2 + \lambda_1 ||\theta||_1 + \lambda_2 ||\beta||_{2,1} : \beta = D\theta \right\},$$  

(10)

where $D \in \mathbb{R}^{(N-1)s \times s}$ is the linear difference operator defined according to Equation (9), and $||\beta||_{2,1} = \sum_{i=1}^{N-1} ||\beta_i||_2$ is the sum of 2-norms. Now it becomes clear that Problem (10) follows the form of Equation (8), and $(\beta, \theta)$ is the solution we want to compute.

The optimization algorithm to get $(\beta, \theta)$:

We follow the work of [3] that describes the ADMM framework for Problem (10) as minimizing its Augmented Lagrange defined by

$$L(\beta, \theta, \mu) := \frac{1}{2} ||y - A\theta||^2 + \lambda_1 ||\theta||_1 + \lambda_2 ||\beta||_{2,1} + \mu^T(\beta - D\theta) + \frac{\mu}{2} ||\beta - D\theta||^2.$$  

(11)
The parameter $\rho > 0$ can be arbitrary positive number or dynamically updated [3]. The ADMM algorithm specialized for Problem (10) is described in Algorithm 1. Generally speaking, we first estimate an initial value for $(\theta^0, \beta^0, \mu^0)$, set $\epsilon_{\text{opt}} > 0$ as the optimality tolerance and then iteratively update the solution until it is close enough to the optimal solution.¹

**Algorithm 1 The ADMM framework for problem (10)**

**Input:** An initial $(\theta^0, \beta^0, \mu^0)$, and $\epsilon_{\text{opt}} > 0$.

**Output:** $(\theta^{k+1}, \beta^{k+1}, \mu^{k+1})$ after $k + 1$ updates.

1: for $k = 0, 1, 2, \ldots$ do
2: Set $\theta^{k+1} \leftarrow \text{argmin}_{\theta} L(\theta, \beta, \mu)$.
3: Set $\beta^{k+1} \leftarrow \text{argmin}_{\beta} L(\theta^{k+1}, \beta, \mu^{k})$.
4: Set $\mu^{k+1} \leftarrow \mu^k + \rho(\beta^{k+1} - D\theta^{k+1})$.
5: if $\|\beta^{k+1} - D\theta^{k+1}\| < \epsilon_{\text{opt}}$ and $\|\beta^{k+1} - \beta^k\| < \epsilon_{\text{opt}}$, then return $(\theta^{k+1}, \beta^{k+1}, \mu^{k+1})$
6: end for

Notice that in Algorithm 1 each iteration involves three generic steps, i.e., Step 2–4. Step 4 being trivial, we describe in detail how to efficiently implement Step 2 and Step 3.

**Update $\theta$:** For brevity, we temporarily drop the iteration counter $k$ and the computation in Step 2 is

$$\arg\min_\theta L(\theta, \beta, \mu) = \arg\min_\theta \frac{1}{2}\|y - A\theta\|^2 - \mu^T D\theta + \frac{\lambda_1}{2}\|\beta - D\theta\|^2 + \lambda_1\|\theta\|_1$$

This is a convex quadratic function with $\ell_1$ regularization. To solve this problem, we adopt the primal-dual active-set method of [14], and explain briefly why this method is favored. It is observed that after several initial ADMM iterations, the subsequent update of $\theta$ is usually moderate, i.e., $\|\theta^{k+1} - \theta^k\|$ is relatively small. By serving $\theta^k$ as the input of Step 2, the method of [14] is able to rapidly yield $\theta^{k+1}$. Such an ability of utilizing a good initial point is called “warm-start” which is enjoyed by many active-set methods.

**Update $\beta$:** First, we can decompose the Augmented Lagrange (11) as

$$L(\beta, \theta, \mu) = \frac{1}{2}\|y - A\theta\|^2 + \lambda_1\|\theta\|_1 - \mu^T D\theta + \sum_{i=1}^{N-1} L_i(\beta_i, \theta, \mu_i),$$

where

$$L_i(\beta_i, \theta, \mu_i) = \lambda_2\|\beta_i\|_2 + \langle \mu_i, \beta_i \rangle + \frac{\rho}{2}\|\beta_i - D\theta_i\|^2.$$  

In Step 3, $\beta_i$ is separable thus is minimized individually for each $L_i$. Let $z := D\theta_i - \frac{\mu_i}{\rho}$, we immediately have

$$\arg\min_{\beta_i} L_i(\theta, \beta_i, \mu_i) = \arg\min_{\beta_i} \lambda_2\|\beta_i\|_2 + \langle \mu_i, \beta_i \rangle + \frac{\rho}{2}\|\beta_i - z_i\|^2.$$  

It turns out that this problem has a closed-form solution according to [21] as:

$$\beta_i = \begin{cases} 0, & \text{if } \|z_i\| \leq \lambda_2, \\ (1 - \frac{\lambda_2}{\|z_i\|^2}) \frac{\rho z_i}{\|z_i\|^2}, & \text{otherwise}. \end{cases}$$

Remarks: In Algorithm 1, for a large-scale problem, updating $\beta$ may be computationally intensive. However, according to our formulation of $L$, since each block $\beta_j$ is independent with other blocks, the update of $\beta$ can be carried out in a distributed fashion, which makes Algorithm 1 applicable in large-scale settings. Later in Section IV, we further improve the scalability of the solution by employing a hierarchical optimization strategy.

D. Identify the System Global Behavior Switching

The switch points from an invariant pair indicates the local behavior change of the system. In this section, we describe an algorithm to aggregate the identified switch points from all invariants to infer the global switch points of the whole system. Note that the aggregation is not trivial due to the noise and uncertainties in the local results. For example, the optimization algorithm may introduce extra switch points, i.e., false positives. On the other hand, since each invariant only represents a partial view of system behavior, its segmentation may not contain a complete set of global switch points. In addition, due to the operational delay between system components, the global switch points discovered by different local models are not necessarily well-aligned in time.

To address those issues, we propose a robust fusion process to discover global switch points from local results. Our algorithm is based on the observation that although there exist noise, delay, and partial impact issues in local results, the system behavior switching will still trigger a significant portion of local models to change parameters accordingly. If we combine all switch points from local models, we should see more points in the neighborhood of true system switching than in other regions. Therefore, if we build the density of those points, the modes of density distribution should correspond to the system level switching.

Our algorithm contains three main steps. First, we collect switch points from all segmented pairs and project them to the time axis. Figure 2(a) provides a scatter plot to illustrate how switch points are distributed across all the models. Note that we use the figure just to demonstrate the fusion process. In practice, the switch points from local models are not as dense as in Figure 2(a). Given the scatter plot, we project all the points to the $x$-axis, i.e., the time index domain. Next, we estimate the density distribution of the aggregated switch points using kernel density estimation, which regards the data points as sampled from a density distribution function and tries to learn that function from the data points, as shown by the blue curve in Figure 2(b). Finally, we extract the local maxima of the density distribution using the mean shift algorithm, and regard those modes (marked by red circles) as the global switch points.

In the following we briefly describe the kernel density estimation and the mean shift based mode search techniques.

**Kernel Density Estimation:** Let $x_i, i = 1, \ldots, n$, be the switch points aggregated from all local models, which are drawn from an arbitrary probability distribution $f(x)$. The kernel density estimate of this distribution $\hat{f}(x)$ (called the Parzen window estimate in pattern recognition), is obtained based on a kernel function $K(u)$ and a bandwidth $h$ as

$$\hat{f}(x) = \frac{1}{n\hat{h}} \sum_{i=1}^{n} K \left( \frac{x - x_i}{h} \right).$$

¹Closeness is measured via optimality condition. Due to the page limit, we omit the derivation of the KKT optimality condition for our problem.
The kernel functions considered here satisfy the following properties
\[ K(u) = K(-u) \geq 0 \quad K(0) \geq K(u) \quad \text{for} \ u \neq 0 \]
\[ K(u) = 0 \quad \text{for} \ |u| > 1 \quad \int_{-1}^{1} K(u) = 1. \quad (13) \]

There are a number of commonly used kernel functions such as the Gaussian kernel, the uniform kernel, the Epanechnikov kernel and so on. For details, see [25]. There also exist some plug-in rules [25, p.72] to determine the value of bandwidth \( h \) in equation (12).

**The Mean Shift Iteration:** The goal of mean shift is to discover the mode of density function (12), which corresponds to a zero of its gradient. The even symmetry of the kernel function allows us to define its profile, \( k(u) \) from \( \frac{1}{nh} K(u) = c_k k(u^2) \), where \( c_k \) is a normalization constant. Hence the gradient of (12) is
\[ \nabla f(x) = \frac{2c_k}{h^2} \sum_{i=1}^{n} (x - x_i) k' \left( \frac{\|x - x_i\|}{h} \right), \quad (14) \]

where \( k'(u) \) is the derivative of the kernel profile. By defining \( g(u) = -k'(u) \), we get
\[ \nabla f(x) = \frac{2c_k}{h^2} \sum_{i=1}^{n} g \left( \frac{x - x_i}{h} \right) \sum_{i=1}^{n} x_i g \left( \frac{\|x - x_i\|}{h} \right) - x. \quad (15) \]

The second term in Equation (15) is called the mean shift vector, because it is the difference between the weighted mean, using the \( g(\cdot) \) for weights, and \( x \), the center of the kernel window. Since the mean shift vector is proportional to the normalized function gradient, it provides an opportunity to locate the mode of the density function \( f \) in an iterative manner. From each point \( x^j \) in the domain of \( f \), a mean shift update would shift \( x^j \) to \( x^{j+1} \):
\[ x^{j+1} = \frac{\sum_{i=1}^{n} c_k G(x^j - x_i)}{\sum_{i=1}^{n} c_k G(x^j - x_i)} \quad (16) \]

where \( G(u) = c_k g(u^2) \). Note that if we chose \( G(u) \) as the Epanechnikov kernel function, the weighted mean in (16) becomes the regular mean of all points in the window. It has been proved that the update of Equation (16) will converge to a mode of the density function [4], [5].

**IV. Further Improving the Optimization Efficiency**

In Section III-C we present an efficient and scalable optimization algorithm, Algorithm 1, to obtain the switch points of each invariant pair. In Algorithm 1, the problem complexity grows linearly with the length of time series, which poses a great challenge for solving large-scale problems. In most physical systems, however, the behavior switching may not happen very frequently due to the constraints of work flow. That is, the system will stay in the same state for certain amount of time after entering into that state. As a result, the solution \( \beta \) in Equation (10) is usually very sparse. This provides an opportunity to further improve the efficiency of segmentation process. Ideally, if we somehow know in advance that \( \theta_i = \theta_{i+1} = \theta_{i+2} = \cdots = \theta_{i+K} \), then we could simply use \( \theta_i \) to replace such whole block of \( K+1 \) variables in Model (10), and thus undercut the problem size. When the switch points are rare, such a variable reduction approach drastically reduces the problem complexity.

**A. The Two-Phase Hierarchical Algorithm**

In the light of the above observation, we design a two-phase hierarchical algorithm to solve the problem. Phase-I divides the time indices into multiple blocks and reduce the variable number for each block. Some of the blocks may be falsely allocated if they contain a switch point. In such a situation, binding \( \theta_i \) on those blocks will cause unpredictable results and in practice the solution of such a block usually show suspiciously distinctness with its neighbors. In Phase-II we then zoom into the suspicious blocks to locate the switch points by building a point-wise model for each of those blocks.

**Phase-I:** Now we discuss Phase-I in detail. For simplicity, suppose that the index length \( N \) is divisible into \( \delta \) blocks—i.e., \( N = \delta \cdot b \)—where each block has equal size \( b \). Since we assume all the values of \( \theta_i \) within each block are identical, we use \( \gamma_i \in \mathbb{R}^s \) to represent the \( i \)th block, where \( s \) is the length of \( \theta_i \). Mathematically, this means \( \theta_i = \gamma_i \) for \( i = ib + 1, \ldots, ib + b \), which equivalents imposing extra constraints on (10). Let \( e_b \) be the \( b \)-dimensional all-one vector and \( I_s \in \mathbb{R}^{s \times s} \) the \( s \)-dimensional identity matrix, the linear constraints between \( \theta \) and \( \gamma \) is compactly expressed as
\[ \theta = \sum_{i=1}^{\delta} e_b \otimes I_s \gamma, \quad (17) \]

where \( \gamma \in \mathbb{R}^{\delta s} \) is the concatenation of \( \gamma_i \) for \( i = 1, \ldots, \delta \). Let \( M \) represent the linear transformation between \( \theta \) and \( \gamma \) of Equation (17), i.e., \( \theta = M \gamma \). It is easily verifiable that the reduced Model (10) could be formulated as
\[ \min_{\gamma, \beta} \left\{ \frac{1}{2} \|y - \tilde{A} \gamma\|^2 + \tilde{\lambda}_1 \|\gamma\|_1 + \tilde{\lambda}_2 \|\beta\|_{2,1} : \tilde{\beta} = \tilde{D} \gamma \right\}, \quad (18) \]

where \( \tilde{A} = AM, \tilde{\lambda}_1 = b \lambda_1, \tilde{\lambda}_2 = \lambda_2 \). Note that \( \tilde{D} \), just like \( D \) of Model (10), is the block difference operator but with reduced dimension. Notice that the reduced Model (18) is in the form of Model (10) hence the ADMM algorithm is applicable. The output of this step is suspicious blocks that need further search for switch points in Phase-II.
Phase-II: Phase-II of the hierarchical optimization strategy is a refinement of Phase-I. After identifying the suspicious blocks from Phase-I, it is natural to zoom into each identified block and pinpoint the switch points by the original point-wise Model (10). In practice we suggest building a point-wise model on indices belonging to both the suspicious block and its two adjacent neighboring blocks. Specifically, suppose the $i_{th}$ block is identified as a suspicious block by the solution of Model (18), a corresponding point-wise Model (10) is built on the time interval $t \in ((i-1)b+1,\ldots,i\ b,\ldots,(i+2)b)$. After such refinement, the output is the switch points identified by our hierarchical strategy.

Efficiency analysis: we close this section by providing some guidelines on the usage of Model (18). First, the hierarchical Model (18) has $(2\delta - 1)s$ variables, which is much fewer than the original Model (10) that has $(2N - 1)s$ variables, given $\delta \ll N$. We also remark that Phase-II can be solved in a distributed manner since the processes of building and solving piecewise models on suspicious blocks are independent. Finally, in our description the time indices are divided into blocks of equal size, this can be easily changed if one wants to set different block sizes by incorporating the prior knowledge.

V. EXPERIMENTS

In this section, we evaluate our proposed method with the following three objectives: (1) Testing the effectiveness of sampling algorithm in selecting invariant pairs; (2) Evaluating the accuracy in identifying the system switching behaviors, including both stagewise and global accuracy; (3) Testing the computational cost and the speedup of the hierarchical segmentation strategy. Note that no other existing methods provide such integrated solution as we do here thus only the results of our approach are reported. We apply our method in both synthetic and real datasets, and report their results in Section V-A and Section V-B, respectively. More specifically, we generate synthetic time series with different properties, whose invariant pairs and the switch points are known upfront. Our method consists of three stages: pair selection, pair segmentation, and global behavior switching identification. For each stage, we generate the data exclusively for it in such a way that it is isolated from being affected by other stages and thus we can evaluate the accuracy of the stage of interest.

After that, we employ two real datasets collected from large-scale physical systems to test the performance of our method in behavior switching identification in real complex systems. Even though the ground-truth switch points are unknown, applying the method on these data sets can still yield invariant pairs and consequently the switch points. We check with domain experts about the identified system behavior switching, and confirm the accuracy of our method.

Before conducting the experiments, we pre-process the data, such as normalizing data and discarding time series with constant values. We re-emphasize that $N$ and $M$ denotes the time series length and the sample size, respectively; $\kappa$ is employed to represent the number of sample times; $\mathcal{G}$ represents the ground-truth switch points in synthetic data. In building the optimization Problem (10), we choose $m = n = 4$ and for Algorithm I we set $\epsilon_{opt} = 10^{-3}$.

A. On Synthetic Data

In this section, multiple synthetic data sets are generated to test the effectiveness of sampling and the accuracy of segmentation. We start by introducing necessary concepts and explaining how the synthetic data is generated. We generate a group of time series, with a set of pre-specified ground-truth switch points $\mathcal{G}$ associated. Specifically, a seed time series $x_1(t)$ is initialized by arbitrary random data with length $N = 10,000$. All other time series within the group are then derived from $x_1(t)$ and $\mathcal{G}$ as follows:

For $i = 2, 3, \ldots$, and for $t$ in each block

1) Generate random parameters $a \in \mathbb{R}^2, b \in \mathbb{R}^3$;
2) Set $x_i(t)$ according to Equation (1) by $x_i(t) + \sum_{j=1}^{2} a_j x_i(t - j) = \sum_{j=0}^{\kappa} b_j x_1(t - j)$;
3) Add noise by $x_i(t) \leftarrow x_i(t) + \epsilon_t \sim \mathcal{N}(0, 10^{-3})$;

Pair Selection: In this section, we test the effectiveness of the sampling algorithm in pair selection using precision and recall. The precision is defined as the percent of selected true invariants over all the selected pairs, while the recall is defined as the percent of selected true invariant over all the true invariants. We generate 3 groups of time series, each of which is associated with 4 switch points. In total, this setting yields 1216 invariants out of 4950 pairs. We set the sample size $M \in [200,600,3000,4000,5000,9000]$, and sample each time series 30 times. The precision-recall curves of different sample sizes are shown in Figure 3.

Fig. 3. Precision and recall curves on synthetic data.

In this figure we can see that the overall accuracy of selecting true invariant pairs increases when the sample size increases from 200 to 600. It reaches a peak with high precision and recall once a certain value is reached, and begins to drop when the sample size further increases from 3000 to 9000. However, the results from the sampling method are always better than non-sampling method that fits the invariant model using the whole time series length. It shows that a well chosen $M$ can improve considerably the accuracy of pair selection. Although the optimal sample size is problem-dependent and may be unknown in advance, the superiority of sampling makes it a competitive alternative to the naïve non-sampling approach.

Pair Segmentation: We now present how an invariant pair is segmented and illustrate the segmentation results. As shown in Figure 4, we generate an invariant of two time series with switch points $\mathcal{G} := \{2000, 4000, 6000, 8000\}$. From the figure, we can hardly tell the switch points by merely looking at the curves of the time series, which is not surprising since
the definition of “invariant” is intrinsically abstract. To test the accuracy of the switch point identification, we choose \( \lambda_1 = 0 \) and \( \lambda_2 = 1 \) for segmentation, and generate the density distribution of the identified switch points in Figure 4(c), where the ground-truth switch points \( G \) are marked in green triangular, and the switch points identified by our method are marked red circles. From this figure we can see that the switch points identified by our method are very close to the ground-truth ones, indicating that our method has high accuracy in identifying the switch points.

**Global Behavior Switching Identification:** To test the global behavior switching identification accuracy, we set \( G := \{1653, 3639, 5923, 7918\} \), and generate a group of 42 time series that result in 861 invariants. In the optimization model for segmentation, we again set \( \lambda_1 = 0 \) and \( \lambda_2 = 1 \).

As described in the mean shift algorithm, to identify the global switch points, we aggregate the switch points of all segmented pairs. By aligning the switch points of each segmented pair row by row, we generate the scatter plot of all the switch points in Figure 5(a), and the density distribution of the aggregated switch points in Figure 5(b), where the green triangulars mark the ground-truth \( G \), and red circles mark the identified global switch points.

![Figure 4. An example of an invariant pair](image)

**Effectiveness of the Hierarchical Segmentation:** In Section IV, a hierarchical optimization strategy is proposed to speed up the original optimization model of Section III-B. To test the effectiveness of the hierarchical strategy, we apply both the hierarchical (block-wise) and non-hierarchical (point-wise) strategies to the dataset used in testing Global Behavior Switching Identification in the previous subsection, and adopt precision and recall to measure their identification accuracy shown in Table I. The average time of segmenting a pair is also reported. We use \( b \) to denote the block size, and \( b = 1 \) corresponds to the non-hierarchical strategy.

<table>
<thead>
<tr>
<th>( b )</th>
<th>precision</th>
<th>recall</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.840</td>
<td>0.988</td>
<td>101.78</td>
</tr>
<tr>
<td>20</td>
<td>0.817</td>
<td>0.915</td>
<td>31.52</td>
</tr>
<tr>
<td>50</td>
<td>0.832</td>
<td>0.917</td>
<td>24.07</td>
</tr>
<tr>
<td>100</td>
<td>0.839</td>
<td>0.925</td>
<td>27.70</td>
</tr>
<tr>
<td>150</td>
<td>0.861</td>
<td>0.943</td>
<td>35.69</td>
</tr>
<tr>
<td>200</td>
<td>0.864</td>
<td>0.953</td>
<td>42.30</td>
</tr>
<tr>
<td>250</td>
<td>0.868</td>
<td>0.958</td>
<td>50.40</td>
</tr>
<tr>
<td>300</td>
<td>0.875</td>
<td>0.963</td>
<td>57.19</td>
</tr>
</tbody>
</table>

Table I demonstrates the power of the hierarchical strategy, which is up to 4 times faster than the original non-hierarchical technique, and obtains similar precision and recall (by a tolerance proportional to the bandwidth for Mean Shift) as the non-hierarchical one in all the measurements. The running time increases slightly when the block size increases, but it also leads to improved precision and recall values. The superior running speed and comparable accuracy of the hierarchical strategy makes it favorable in practice.

**B. On Real Data**

After systematically evaluating the accuracy and computational cost of our segmentation method, we now apply it to real-world datasets to test its practical performance in identifying behavior switching. To do so, we employ two datasets collected from physical system A and physical system B. Due to the business privacy issue, we do not disclose the actual name of the systems, and use A and B to refer them. In such systems, hundreds of sensors are deployed to monitor the operation signal in different system components, and the monitoring duration ranges are several years, which contains phenomena of normal operations, state changes, noises, etc. More specifically, in system A, we have more than 100 sensors, each containing 3686 points, resulting in 1528 selected invariants; while in system B, we have about 400 sensors, each containing 9360 points, with 4336 selected invariants.

Due to the massiveness of data, it is impossible to investigate all invariants and their switch points by hands. We apply our segmentation method to each of the system, and consult the results with the system engineers to confirm the accuracy. According to their feedback, the identified switch points align well with their change of operation strategies, and has clear physical meanings. In pair selection, we set \( M = 500 \), \( \kappa = 30 \). In pair segmentation (Algorithm 1), we set \( \lambda_1 = 5 \), \( \lambda_2 = 2 \), and the number of blocks in hierarchical strategy \( b = 20 \).

**Physical System A:** In this dataset, each sensor monitors a distinct physical measure and consequently the collected data
show significant heterogeneity among different time series. To illustrate the effectiveness of our pair selection, we plot out an invariant pair returned by our pair selection algorithm in Figure 6.

![Invariant Pair](image1)

(a) Time series of an invariant

![Invariant Pair](image2)

(b) Time series of an invariant

![Identified Switch Points](image3)

(c) Identified switch points

Fig. 6. An example of an invariant pair

Even a single time series curve of Figure 6 shows some signs of state switches. Inside each state, the observed values show relatively steady trend either in constant or linear fashion. When the underlying system behavior switches, it is reflected by the abrupt curve value changes. For example, one of the time series see value surges at around index 1000 whereas the other see plummets and the invariant model before and after index 1000 does show distinct difference. To demonstrate this more clearly, we plot the density distribution of the identified switch points in Figure 6(c), which shows the abrupt value changes are well captured by the results of the segmentation since the switch points (red circles) coincide with the spikes of the curves.

Next, we report the identification results of system behavior switching. Figure 7 shows the aggregated segmentation results of all selected invariants.

![Aggregated Segmentation](image4)

(a) Scatter plot of switch points

![Aggregated Segmentation](image5)

(b) Density distribution of aggregated switch points

Fig. 7. Identification of global switch points of physical system A

In physical systems the switch of behaviors usually straddles several consecutive time indices. Consequently, we see patterns of bands rather than vertical lines in Figure 7(a). However, the identified global switch points (marked by red circles) in Figure 7(b) are very representative of the locations where the system behavior switching happens.

Physical System B: The physical system B data is more homogeneous than the physical system A data as it has more components with similar functionalities. Figure 8 demonstrates an invariant pair selected by our method. From this figure we can see there may be a major switch at around index 800. To confirm, we then segment this pair and identify several switch points from the density distribution as shown in Figure 8(c).

![Invariant Pair](image6)

(a) Time series of an invariant

![Invariant Pair](image7)

(b) Time series of an invariant

![Identified Switch Points](image8)

(c) Identified switch points

Fig. 8. An example of an invariant pair

To identify the global behavior switching, we aggregate the switch points of all segmented pairs and generate Figure 9. Similarly, the band patterns are observed and the identified global switch points again coincide with the observation value jumps or drops of Figure 8.

![Aggregated Segmentation](image9)

(a) Scatter plot of switch points

![Aggregated Segmentation](image10)

(b) Density distribution of aggregated switch points

Fig. 9. Identification of global switch points of physical system B

**Speedup of the Hierarchical Segmentation:** We close this section by reporting the speedup of the hierarchical optimization strategy over the non-hierarchical one in Table II, where \( b = 1 \) means non-hierarchical method. From this table we can see that the hierarchical strategy is able to undercut the total running time by up to 60% for physical system A data and 83% for physical system B data. Regardless of the block size \( b \) of our choice, the hierarchical optimization strategy is predominantly faster.

**VI. RELATED WORK**

In this section, we survey the relevant work on segmentation and invariants analysis in time series. Time series segmentation has been extensively studied, and the methods can be
TABLE II. RUNNING TIME OF HIERARCHICAL AND NON-HIERARCHICAL SEGMENTATION

<table>
<thead>
<tr>
<th>b</th>
<th>Physical System A time(s)</th>
<th>Physical System B time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24.27</td>
<td>184.40</td>
</tr>
<tr>
<td>20</td>
<td>7.73</td>
<td>31.66</td>
</tr>
<tr>
<td>50</td>
<td>8.44</td>
<td>42.85</td>
</tr>
<tr>
<td>100</td>
<td>10.20</td>
<td>48.41</td>
</tr>
<tr>
<td>150</td>
<td>10.84</td>
<td>51.84</td>
</tr>
<tr>
<td>200</td>
<td>10.61</td>
<td>53.78</td>
</tr>
<tr>
<td>250</td>
<td>12.91</td>
<td>57.47</td>
</tr>
<tr>
<td>300</td>
<td>12.96</td>
<td>65.24</td>
</tr>
</tbody>
</table>

divided into three categories. (1) The dynamic programming based methods [2], [10], [22] formulate the segmentation as a dynamic programming and then recursively segment the time series into pieces. The main weakness of those methods is that they require the prior knowledge about the number of system states in advance. (2) The heuristics based methods culminate in [12] where the classic top-down, bottom-up and sliding based search methods are compared and a hybrid version is proposed. Heuristic methods can be easily implemented, but the results are not stable. (3) The optimization based approaches [1], [16], [19], [20], [24] recently gained much favor for their flexibility and effectiveness in expressing the expected results of segmentation as a function objective. The optimization models in [16] segment time series into pieces of linear or constant segments, while those in [1], [20], [24] can segment more complicated relationship. [15] gives a nice summary of how the optimization model should be built to achieve desired properties.

Recent research finds that invariants are common in today’s complex distributed information systems and searching invariants from massive data can greatly strengthen our understanding of the system dynamics. The concept of flow intensity is used in [11] to quantify the invariant relationship which can effectively identify invariants. A pruning technique is proposed in [6] to speed up the search by utilizing an identification upper bound and successful applications of the invariant model are fault detection and localization [7], [23]. All these searching methods, however assume that the invariants are constant across the time range hence do not apply to our problem.

VII. CONCLUSION

We develop an efficient method to discover system behavior switching by inferring it from the relationship of system attributes. We formulate the objective of behavior switching discovery as an optimization problem, design several novel methods in both low and high levels to efficiently solve it, and further boost the solution efficiency with a hierarchical optimization strategy. A fusion mechanism is designed to aggregate the results from different system attributes to unveil the global system behavior switching with the consideration of noise, event lag, etc. Our method is evaluated in both synthetic and real datasets, which shows that it achieves high accuracy in identifying behavior switching in systems with a mixture of states and has low computational complexity.

REFERENCES


