Household Electricity Consumption Data Cleansing

Guoming Tang  
University of Victoria, Victoria, BC, Canada  
guoming@uvic.ca  
Kui Wu  
University of Victoria, Victoria, BC, Canada  
wkui@uvic.ca  
Jian Pei  
Simon Fraser University, Burnaby, BC, China  
jpeii@cs.sfu.ca  
Jiuyang Tang  
National University of Defense Technology, China  
jiuyang_tang@nudt.edu.cn  
Jingsheng Lei  
Shanghai University of Electric Power, China  
jshlei@shiep.edu.cn

ABSTRACT
Load curve data in power systems refers to users’ electrical energy consumption data periodically collected with meters. It has become one of the most important assets for modern power systems. Many operational decisions are made based on the information discovered in the data. Load curve data, however, usually suffers from corruptions caused by various factors, such as data transmission errors or malfunctioning meters. To solve the problem, tremendous research efforts have been made on load curve data cleansing. Most existing approaches apply outlier detection methods from the supply side (i.e., electricity service providers), which may only have aggregated load data. In this paper, we propose to seek aid from the demand side (i.e., electricity service users). With the help of readily available knowledge on consumers’ appliances, we present a new appliance-driven approach to load curve data cleansing. This approach utilizes data generation rules and a Sequential Local Optimization Algorithm (SLOA) to solve the Corrupted Data Identification Problem (CDIP). We evaluate the performance of SLOA with real-world trace data and synthetic data. The results indicate that, comparing to existing load data cleansing methods, such as B-spline smoothing, our approach has an overall better performance and can effectively identify consecutive corrupted data. Experimental results also demonstrate that our method is robust in various tests. Our method provides a highly feasible and reliable solution to an emerging industry application.

1. INTRODUCTION
Electricity usage data, on the one hand, plays an important role in big data applications, and on the other hand, has been severely under explored. A recent news article appeared in Forbes [22] said, “But for the most part, utilities have yet to realize the potential of the flood of new data that has begun flowing to them from the power grid, . . . . And in some cases, they may not welcome it.” Yet, existing power grid is facing challenges related to efficiency, reliability, environmental impact, and sustainability. For instance, the low efficiency of current electric grid could lead to 8% of electric energy loss along its transmission lines, and the maximum generation capacity is in use only 5% of the time [13].

The emerging smart grid technology aspires to revolutionize traditional power grid with state-of-the-art information technologies in sensing, control, communications, data mining, and machine learning [6, 13]. Worldwide, significant research and development efforts and substantial investment are being committed to the necessary infrastructure to enable intelligent control of power systems, by installing advanced metering systems and establishing data communication networks throughout the grid. Consequently, power networks and data communication networks are envisioned to harmonize together to achieve highly efficient, flexible, and reliable power systems.

Among the various types of data transmitted over the smart grid, load curve data, which refers to the electric energy consumption periodically recorded by meters at points of interest across the power grid, has become the critical assets for utility companies to make right decisions on energy generation, billing, and smart grid operations. Load curve data, which “is beginning to give us a view of what the customer is actually experiencing, something that we’ve never ever seen before” [22], is precious user behavior data, and is an important type of big data.

Load curve data collected and reported from smart meters at end-users’ premises is especially important for both energy supply and energy demand sides. On the demand side, it has direct impact on customers’ energy bills and their trust on the still nascent smart grid technology. On the energy supply side, inaccurate load data may lead to large profit losses and wrong business decisions. In 2012, 126.8 million residential customers in the US used over 1,374 billion kWh, which counts to over 33% of the total electric energy in the US [11]. The importance of this huge amount of energy and its financial implication cannot be over emphasized.

Nevertheless, it is unavoidable that load curves contain corrupted data and missing data, caused by various factors, such as malfunctioning meters, data packet losses in wireless networks, unexpected interruption or shutdown in electricity use, and unscheduled maintenance [5]. Due to the huge volume of load curve data, it is hard for utilities to manually identify corrupted load curve data. Unfortunately, problems caused by corrupted data are usually realized only after it is too late, such as after a customer receiving a suspicious yet hard to rebut high energy bill.

As a concrete example, according to the news reports [19, 23], some customers in the province of British Columbia, Canada, were baffled by energy bills that are more than double what they were charged before the smart meter installation. While the problem could be identified by common sense and certain agreement might be reached by good faith negotiations [19, 23], fixing the ques-
tionable bill is another head-scratching and embarrassing issue to the utility. As a response to customer complaints, the utility normally took remedy actions, such as replacing the smart meters or taking back the smart meters for lab testing [23]. Such a remedy, however, can hardly be effective. According to CBC News [24], “Government estimates indicate there are about 60,000 smart meter holdouts (in the province).” Overall, the users and the utility company have the well-aligned interest and should work together to tackle this critical problem plaguing the electric power industry.

Techniques of load data cleansing have been proposed to deal with load data corruption problem recently [5]. Most existing load data cleansing methods are designed for the supply side (i.e., electricity service providers), to help the utility companies find the corrupted data and protect their profits. From the supply side, the collected load data is usually aggregated data, i.e., the energy consumption of a billing unit such as a house or a commercial building. When performing data cleansing on the supply side, due to the difficulty of obtaining extra knowledge behind the aggregated load data, most existing approaches apply outlier detection methods, i.e., the data that deviates remarkably from the regular pattern is identified as corrupted data. Various assumptions about the data generation mechanism are required for outlier detection, but due to limited information, those assumptions are usually based on empirical knowledge or statistic features of the data. Such outlier detection methods are oblivious of appliances’ various energy consumption models and may not be accurate or fair to customers. We call these methods appliance-oblivious. Such methods suffer from a few important deficiencies.

For example, the regression-based outlier detection methods find statistical patterns of load data and claim the data significantly deviating from the patterns as corrupted data. Nevertheless, such resulted outliers are not necessarily corrupted data. In addition, without the knowledge of appliances’ energy consumption models, some “hidden” corrupted data is hard to detect. To be specific, the energy consumption of a group of appliances in a house or a building is a stochastic process. The stochastic feature makes it hard to establish a fixed pattern. Turning on/off any high-power appliance may lead to a steep change in load curve. Using appliance-oblivious data cleansing methods, the data generated under such a condition is likely to be captured as outliers.

As another example, appliance-oblivious methods cannot deal with “hidden” corrupted data. Fig. 1 shows an example of three appliances, $A_1$, $A_2$, and $A_3$, which have power ranges of $[2,4]$, $[10,12]$ and $[30,32]$, respectively. The load data within some ranges such as $(4, 10)$, $(16, 30)$, and $(36, 40)$ cannot be generated by any combination of the three appliances. Nevertheless, such data may not be identified by existing outlier detection as corrupted data.

Recently, with the emergence of fine-grained in-house energy monitoring systems, customers now have the capability to monitor their own energy usage more closely and more accurately [27]. As such, users may possess more knowledge behind the data, e.g., the decomposition of total energy consumption according to main appliances. Even if in-house energy monitoring system is not available, users should know the rated power of appliances’, which are easily accessible from the appliances’ manual, technical specification or public websites, such as [12]. This knowledge presents new opportunities to perform load data cleansing on the customer side directly or on the supply side with auxiliary information from customers. This new angle of tackling the corrupted load data problem can greatly improve the quality of load data.

In this paper, we tackle the practical problem of corrupted load curve data identification in the industry context by developing an appliance-driven approach. Specifically, we make the following contributions:

- First, we define a new criterion in identification of corrupted load data. The criterion is aware of domain knowledge, including the power range of appliances and the physical laws behind valid load data.

- Second, we formally formulate the Corrupted Data Identification Problem (CDIP) and establish an optimization model to solve the problem. Furthermore, we introduce a new concept, called virtual appliance, in the objective function to help record corrupted data. Our empirical study in a proof-of-concept electricity usage test environment shows that a solution to the optimization problem is capable of precisely identifying the corrupted data, even without obtaining the exact on/off states of appliances. This nice feature indicates that our method is both effective and robust.

- We develop a sequential local optimization algorithm (SLOA) to approach CDIP efficiently. SLOA focuses on solving CDIP in a smaller time window, and considers the correlation between consecutive small windows. While we show that the original CDIP problem is NP-complete, our SLOA method offers an efficient heuristic solution and can achieve a very high detection precision. As an extra benefit, by applying the sequential optimization algorithm, we can easily handle consecutive corrupted data.

The rest of the paper is organized as follows. In Section 2, we review the related work. In Section 3, we formally formulate the corrupted data identification problem (CDIP). To solve CDIP, we develop an optimization model in Section 4. As CDIP is proven to be NP-complete, we develop SLOA to find an approximate solution in Section 5. We evaluate the performance of our method with real-world trace data and synthetic data in Section 6 and Section 7, respectively, and test the robustness of SLOA in Section 8. The paper is concluded in Section 9.

2. RELATED WORK

Most related literature treats the corrupted data the same as outliers in load pattern and focuses on outlier detection. A broad spectrum of techniques for outlier detection in load data have been developed, which include regression-based time series analysis, uni-
variante statistical methods, and data mining techniques. We review
them briefly here.

2.1 Regression-Based Time Series Analysis

Regression-based time series analysis is the most widely used
approach for outlier detection in load data [11, 5, 20, 24]. Mateos
and Giannakis [21] developed a nonparametric regression method
that approximates the regression function via \( \ell_0 \)-norm regularization.
Chen et al. [5] proposed a nonparametric regression method based
on B-spline and kernel smoothing to identify corrupted data.
Abraham and Chuang [1] analyzed residual patterns from some re-
gression models of time series and used the patterns to construct
outlier indicators. They also proposed a four-step procedure for
modeling time series in the presence of outliers. Greta et al. [20]
considered the estimation and detection of outliers in time series
generated by a Gaussian auto-regression moving average (ARMA)
process, and showed that the estimation of additive outliers is re-
lated to the estimation of missing observations. ARMA is also uti-
lized in [15, 2, 4, 26] as a fundamental model to identify outliers.

2.2 Univariate Statistical Methods

Univariate statistical methods are another type of techniques for
outlier detection. Univariate statistical methods deal with outliers
in load data by processing load data as one-dimensional real val-
ues [14, 5, 16, 9]. Most univariate methods for outlier detection
assume an underlying a priori distribution of data. The outlier de-
tection problem is then translated to finding those observations that
lie in the so-called outlier region of the assumed distribution, which
is defined by a confidence coefficient value [9]. Since the statistical
methods are susceptible to the number of exemplars, a simple but
effective method named Boxplot or IQR is proposed in [28] to deal
with small-sized exemplars.

2.3 Data Mining Techniques

In addition to the above methods, data mining techniques are also
applied to identify outliers, such as k-nearest neighbor [25, 17], k-
means, k-medoids [4], and DBSCAN [18]. As a type of clustering
methods, they group data with similar features, and identify data
items that do not strongly belong to any cluster or far from other
clusters as outliers. Recently, Aggarwal [3] provided a thorough
survey on outlier detection.

Nevertheless, all the above methods do not consider the special
physical laws behind the load data. Regression-based methods as-
sume that the data follows a certain pattern, which can be modeled
by a function governed by a set of parameters; univariate methods
assume that the data is sampled from a certain known distribution;
clustering methods assume that the data is well structured as clus-
ters and the corrupted data deviates significantly from the normal
structure. Obviously, the underlying assumptions in the existing
methods are quite general and do not capture the specific features
of load data well. Our paper fills the gap and differs from the ex-
isting literature by offering a completely new angle to address the
load data corruption problem.

3. THE CORRUPTED DATA IDENTIFICA-
TION PROBLEM

In this section, we present a formal problem definition. Before
that, we first describe an energy consumption model and discuss
the generation rules of load data.

3.1 Energy Consumption Model

Load data is time series data that records users’ energy consump-
tion. It is collected by smart meters periodically at a certain sam-
pling frequency. Without loss of generality, we assume that the time
is slotted, with each timeslot equal to the sampling interval time. In
the rest of the paper, we thus use the terms “time”, “timeslot” and
“sampling interval” interchangeably.

We denote the load data from timeslot \( t = 1 \) to timeslot \( t = n \)
in a column vector as

\[
Y \equiv [y_1, y_2, \ldots, y_n]^T,
\]

where each value \( y_i \) in the vector represents the aggregated energy
consumption of all appliances in a property, say a house at timeslot
\( i \). The energy consumption at a time instant depends on the appli-
cances’ on-off states and their individual power level.

We assume that a house includes \( m \) appliances in total, and the
power of the \( k \)-th appliance is \( p_k \) (watts). At any time instant, if
we record the power level of each individual appliance, we can de-
fine an \( m \) dimensional column power vector to capture energy
consumption of the house:

\[
P \equiv [p_1, p_2, \ldots, p_m]^T.
\]

Note that the power level of an appliance normally does not re-
maintain at a fixed value but changes in a certain range. For this reason,
we define two \( m \) dimensional column vectors, denoted as \( P_l \) and
\( P_u \), respectively:

\[
P_l = [l_1, l_2, \ldots, l_m]^T\quad (3)
\]

\[
P_u = [u_1, u_2, \ldots, u_m]^T,\quad (4)
\]

where \( l_i \) and \( u_i \) represent the lower and upper bounds of the power
level of the \( i \)-th appliance, respectively. A power vector \( P \) is called
valid if, for each value \( p_i \) in \( P \), \( l_i \leq p_i \leq u_i \).

At any instant, the state of an appliance may be either on or off.
We use an \( n \times m \) 0-1 state matrix, \( S = [S_{ij}]_{n \times m} \), to record the
states of the \( m \) appliances from time \( t = 1 \) to \( t = n \), where
\( S_{i,k} = 1 \) indicates that the \( k \)-th appliance is on at time \( i \), and 0
otherwise. In addition, we call the \( i \)-th row of \( S \) a state vector at
time \( i \), denoted by:

\[
S_i \equiv [S_{i,1}, S_{i,2}, \ldots, S_{i,m}].
\]

3.2 Generation Rules of Load Data

We have the following observations. First, a valid load data ele-
ment \( y_i \) (in watt-hours) should be equal to the inner product of the
state vector and the power vector at \( t = i \), multiplied by the sam-
pling interval time. This is a basic physical law for load curve data
generation. Second, since the sampling interval is typically small
(e.g., 10 seconds), we assume that the probability that an appliance
has more than one on-off switch events during a timeslots is negli-
gible. In addition, the total number of on-off state switches of all
appliances during a timeslot should be small. This feature is called
the temporal sparsity of on-off state switching events. Intuitively,
this feature means that in normal operation it is unlikely that the
household turns on-off many appliances in a short time. Based on
the above observations, we can define the generation rules of load
data.

**Definition 1 (Generation Rules). Assume that the ini-
tial state of appliances is \( S_0 \). We claim that each valid load data,
\( y_i \), must satisfy the following rules:

\[
\begin{cases}
S_i \cdot P_l / f \leq y_i \leq S_i \cdot P_u / f \\
\|S_i - S_{i-1}\|_1 \leq \delta,
\end{cases}
\]

\footnote{Residential smart meters can support sampling rate as high as 1 sample per second [29].}
where $f$ is data sampling frequency, $1 \leq i \leq n$, and $\delta$ is the upper bound on the total number of on-off state switches for $m$ appliances during a sampling interval.

Note that the energy consumption value (watt-hours) is calculated with power value (watt) multiplied by time $1/f$ (hour). To keep our discussion simple, we assume that a valid initial state vector $S_0$ is given at this moment. We will relax this assumption later and show that the impact of an inaccurate initial state vector quickly becomes negligible as long as the system runs for just a little while (Section 5).

### 3.3 Problem Definition

Based on the above generation rules, corrupted data is the values that break any of the rules.

**Definition 2 (CDIP).** The corrupted data identification problem (CDIP) is, given load data $Y = \{y_1, y_2, \cdots, y_n\}$, power bound vectors $P_1, P_n$, and a sampling frequency $f$, find corrupted data items that violate any of the generation rules, i.e., $C = \{y_i : y_i$ violates (Equation 6), for $1 \leq i \leq n\}$.

### 4. AN IMPORTANT STEP TOWARDS SOLVING CDIP

To solve CDIP, a naive idea is to find all the solutions satisfying the constraints in (Equation 6), by brute-force, exhaustive search for all possible appliance states. This method is very time-consuming. Even for a small data set it is very costly to find the answer. Since the generation rules can be considered as constraints in an optimization problem, we will show how the problem can be transformed to an optimization problem, which sheds light on a fast solution to an approximate problem.

**Definition 3 (Virtual Appliance).** Besides the real appliances, we introduce a virtual appliance into the system. Its associated power is called virtual power, and we record the values of virtual power from time $t = 1$ to $t = n$ in a virtual power vector

$$V = [v_1, v_2, \cdots, v_n]^T,$$

where $v_i \in (-\infty, +\infty)$ denotes the virtual power at time $t = i$.

By introducing the virtual appliance, we can develop the following optimization model to solve CDIP:

$$\min_{S_i, v_i} \|V\|_1$$

subject to

$$\frac{(S_i \cdot P_i + v_i)}{f} \leq y_i \leq \frac{(S_i \cdot P_n + v_i)}{f}$$

$$\|S_i - S_{i-1}\|_1 \leq \delta$$

$$S_{i,j} \in \{0, 1\}$$

$$1 \leq i \leq n$$

$$1 \leq j \leq m$$

To understand the rationale behind the formulation of Equation 6, it is worthwhile to point out that $v_i \in V$ will come into play whenever $S_i$ cannot satisfy the generation rules, i.e., the virtual appliance is “turned on” when the load data $y_i$ is corrupted. Thus, $v_i$ essentially makes a record to the corrupted data. After obtaining the final solution to Equation 6, the $v_i$ variables with non-zero values indicate the corrupted data, i.e.,

$$C = \{y_i : v_i \neq 0 \text{ for } 1 \leq i \leq n\}.$$  

We try to minimize the $\ell_1$-norm, because it is proven that for most large under-determined systems of linear equations the minimal $\ell_1$-norm solution is also the sparsest solution (i.e., resulting in the minimal number of non-zero values of $v_i$) [10]. In addition, a larger $v_i$ value means that a corrupted $y_i$ is farther away from a valid range. In this sense, the value of $v_i$ can be also regarded as the corrupted degree of $y_i$.

The problem can be proven NP-complete (refer to Appendix A). By investigating the special structure of the problem, however, we can develop an effective heuristic algorithm introduced in the next section.

### 5. SEQUENTIAL LOCAL OPTIMIZATION ALGORITHM

In this section, we propose a Sequential Local Optimization Algorithm (SLOA) and develop a quantitative strategy to estimate the minimum local window size.

#### 5.1 SLOA

The temporal sparsity of corrupted load data suggests that we can perform optimization in a smaller, local time window. By considering the correlation between consecutive timeslots, we design a Sequential Local Optimization Algorithm (SLOA). Without loss of generality, we take a load data from time $t = 1$ to $t = n$ as an example to show the major steps of SLOA.

**Step 1.** Consider a small time window with size of $w$, $1 \leq w < n$, which starts from time $k$ to time $k + w - 1$. Given the state vector at time $k - 1$, i.e., $S_{k-1}$, we consider the following optimization problem:

$$\min_{S_i, v_i} \sum_{i=k}^{k+w-1} |v_i|$$

subject to

$$\frac{(S_i \cdot P_i + v_i)}{f} \leq y_i \leq \frac{(S_i \cdot P_n + v_i)}{f}$$

$$\|S_i - S_{i-1}\|_1 \leq \delta$$

$$S_{i,j} \in \{0, 1\}$$

$$1 \leq i \leq k + w - 1$$

$$1 \leq j \leq m$$

By setting $w \ll n$, we can significantly reduce the search space. Actually, we can show that the computational complexity to solve the above problem is $O(M^w)$, where $M = 2^m + 2^m + \cdots + 2^m$ (refer to Appendix B). Since $m$ is the total number of appliances and $\delta \ll m$, the problem can be solved quickly using tools such as CVX 2.0 with a Gurobi engine [7].

**Step 2.** For the $k$-th time window that starts from time $k$, we use the following strategy to handle consecutive corrupted data: if the data point at time $k$ is identified to be corrupted, i.e., $v_k \neq 0$, recover the current state vector from the previous one, i.e., set $S_k = S_{k-1}$.

**Step 3.** Repeat Step 1 and Step 2 from $k = 1$ to $k = n$, and solve problems in form of Equation 10 sequentially. After $n$ iterations, we can get a sequential solution $v_1, v_2, \cdots, v_n$. Thus, the corrupted data set is $C = \{y_i : v_i \neq 0, 1 \leq i \leq n\}$, in which $v_i$ is the corrupted degree of load data $y_i$.

Algorithm 1 shows the pseudo code of SLOA.

#### 5.2 Estimation of Minimum Local Window Size

Clearly, one key question in SLOA is to determine a suitable size of the local window. In principle, we want the size to be as small
Output: Corrupted data set \( C \), corrupted degree \( v_i, 1 \leq i \leq n \)
1: \( v_0 = 0 \)
2: \( C = \emptyset \)
3: for \( k = 1 : n \) do
4: Solve Problem (Equation 10), and obtain \( v_i \) and \( S_i \) where \( k \leq i \leq k + w - 1 \)
5: if \( v_k \neq 0 \) then
6: \( C = C \cup \{ y_k \} \)
7: \( S_k = S_{k-1} \)
8: end if
9: end for
10: return \( C, \{ v_1, v_2, \ldots, v_n \} \)

Algorithm 1 Sequential Local Optimization Algorithm

Input: Load data \( \{ y_1, y_2, \ldots, y_n \} \), power bounds \( P_l, P_u \), initial state \( S_0 \), sampling frequency \( f \), local time window size \( w \).

Output: Corrupted data set \( C \), corrupted degree \( v_i, 1 \leq i \leq n \)

as possible to speed up the calculation, but a size too small may result in a poor solution largely deviating from the global optimal one. For example, in the extreme case of \( w = 1 \), SLOA becomes a simple greedy search algorithm, where the final solution may not be good. On the other hand, if \( w = n \), the problem becomes the same as Equation (8), which is hard to solve. What is the minimum local window size that leads to a nearly global optimal solution?

Since it is hard to obtain a strict proof that the local optimal solutions together lead to the global optimal solution, we use the following heuristics to estimate the minimum local window size: within the local window, it should be possible that one state vector can be transited to any other state vectors in the vector space. In other words, within the local window, we should cover all possible state vectors in the search. This heuristics sheds light on finding the minimum local window size, as formulated below.

Definition 4 (Overlap Index). Consider \( m \) appliances denoted by a set \( \{ R_1, R_2, \ldots, R_m \} \), where \( R_i \equiv [l_i, u_i] \subset \mathbb{R} \) and represents the \( i \)-th appliance’s power range. The overlap index of \( m \) appliances is defined as:

\[
O \equiv \frac{\sum_{i=1}^{m} \int_{p_{\min}}^{p_{\max}} I(R_i \cap \{ x \})dx}{\int_{p_{\min}}^{p_{\max}} I(\cup_{i=1}^{m} R_i \cap \{ x \})dx} \tag{11}
\]

where \( p_{\max} \) and \( p_{\min} \) stand for the maximum and minimum power of all appliances, respectively, and \( I(x) \) is an indicator function

\[
I(x) = \begin{cases} 
1, & x \neq \emptyset \\
0, & x = \emptyset 
\end{cases} \tag{12}
\]

Note that the denominator \( \int_{p_{\min}}^{p_{\max}} I(\cup_{i=1}^{m} R_i \cap \{ x \})dx \) includes all valid power values, i.e., the ones that can be covered by at least one appliance’s power range. We can see that the overlap index represents the number of appliances whose power range covers a valid power value, averaged over the whole power range of all appliances. In particular, \( O = 1 \) indicates that no pair of appliances have overlapped power, and \( O = m \) means that all appliances have the same power range. Intuitively, when \( O \) is large, there is a good chance of finding multiple local optimal solutions to Equation (10), since there are multiple equivalent choices to turn on/off appliances in each iteration.

With the heuristics in estimating the minimum local window size, we have the following result.

Lemma 1. Given the overlap index \( O \) of \( m \) appliances and the upper bound \( \delta \) on the total number of on-off state switches in a timeslot, in order to get the nearly global optimal solution to Equation (9) via Equation (10), the minimum local window size \( w \) is

\[
w \geq \max\left\{ \left\lfloor \frac{m}{\delta} \right\rfloor, 1 \right\}. \tag{13}
\]

Proof. We prove the following condition holds: within the local window, we can cover all possible state vectors in the search.

First, the value of \( w \) relates to upper bound \( \delta (\leq m) \) on the total number of on-off switches in one timeslot. It is obvious that from time \( t = i \) to \( t = i + 1 \), the state vector \( S_i \) can only change to another state vector \( S_{i+1} \), where \( \| S_{i+1} - S_i \|_1 = \delta \). If \( \delta = m \), then within one step, a state vector is allowed to change to any other state vector. On the other hand, if \( \delta = 1 \), within one step, a state vector can only change one value in the vector. In other words, from one state vector, it requires at least \( \left\lceil \frac{m}{\delta} \right\rceil \) timeslots to reach any other state vector in the state vector space, i.e., \( w \geq \left\lceil \frac{m}{\delta} \right\rceil \).

Second, the overlap index \( O \) can reduce the value of \( w \). Based on the meaning of \( O \), \( m \) appliances with overlap index \( O \) are equivalent to \( m/O \) appliances without overlapped power. Replace \( m \) with \( m/O \), we can get \( w \geq \left\lceil \frac{m}{\delta \cdot O} \right\rceil \). Considering \( w \geq 1 \), we conclude:

\[
w \geq \max\left\{ \left\lfloor \frac{m}{\delta \cdot O} \right\rfloor, 1 \right\}. \tag{13}
\]

Since we want \( w \) to be as small as possible, \( w = \max\left\{ \left\lfloor \frac{m}{\delta \cdot O} \right\rfloor, 1 \right\} \).

Please note that, although the minimum local window size obtained above is an estimation, it works effectively in our experiments with real-world data as well as with synthetic data.

5.3 Algorithm Analysis

Given \( n \) load values, \( m \) appliances, and the upper bound \( \delta (\leq m) \) on the total number of on-off state switches in a timeslot, the computational complexity of the original problem (Equation 8) is \( O(M^{m}) \), where \( M = \binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{m} \). Using SLOA, solving the optimization problem (Equation 10) for \( n \) times results in the time complexity of \( O(n \cdot M^{m}) \), where \( w \in \mathbb{Z}^+ \) and \( w \ll n \) (refer to Appendix B). Please note that the appliance number \( m \) is a constant value and \( w \) is also a small constant.

Obviously, the larger the value of \( w \), the higher the computational complexity. Fortunately, the overlap index of appliances in a house or a building is usually high, as observed in our real-world experiment tested. This fact allows us to select a small local window size following Lemma 1. Therefore, in the application scenarios, SLOA can approach the NP-complete problem heuristically and efficiently. We will show that this algorithm indeed can provide a good solution with abundant experimental results in the following sections.

6. EXPERIMENTAL EVALUATION ON REAL DATA

We evaluate our method with real-world trace data from a real-world energy monitoring platform. We monitored the appliances’ energy consumption of a typical laboratory and a lounge room on the fifth floor of the Engineering/Computer Science Building at the University of Victoria (UVic) for two months. The real-time power of laptops, desktops and some household appliances was recorded. Each appliance’s power level was measured every 10 seconds and the measurement results were transmitted with ZigBee radio to a server that stores the data. The monitored appliances and their regular power values shown in Fig. 9. An appliance’s regular power is an approximate range around the rated power where this appliance works.
We test the data day by day over the two-month period. Fig. 3 demonstrates one-week and one-day load data collected by our platform. For clear illustration, we only show one-day data as an example. Note that even in a lab setting like ours, there indeed exists some apparent corrupted data, indicated by the dashed red dots in Fig. 3.

In order to introduce more corrupted data, we asked three students to distort the load data with “falsification”, i.e., they were asked to arbitrarily modify the aggregated load data within the range of $[0, \infty)$. These changed data together with the original corrupted ones were labeled and used as the ground-truth to verify the performance of our method.

Since the existing appliance-oblivious load data cleansing methods, such as B-spline smoothing, detect outliers and consider outliers as corrupted data, we use the terms “outliers” and “corrupted data” interchangeably hereafter. For outlier detection, four statistical results can be obtained: (1) true positive ($TP$), the number of points that are identified correctly as outliers; (2) false positive ($FP$), the number of points that are normal but are identified as outliers; (3) true negative ($TN$), the number of points that are normal and are not identified as outliers; (4) false negative ($FN$), the number of points that are outliers but are not identified. Using $TP, FP, TN$ and $FN$, we evaluate the following three broadly-used performance metrics: precision, recall, and F-measure. Precision is the ratio of the number of correctly detected corrupted values over the total number of detected values; recall is the ratio of the number of correctly detected values over the number of pre-labeled corrupted values; and the F-measure is a harmonic mean of precision and recall, i.e.,

$$F\text{-}measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}.$$

For comparison, we implement and test an appliance-oblivious data cleansing method, B-spline smoothing, which is introduced in [5] to identify corrupted load data. In the B-spline smoothing method, we set the confidence coefficient $\alpha = 0.05$, which results in a confidence interval of 95%. We treat the degree of freedom ($df$) as a variable, whose value is trained when smoothing the load curve data. For our method, the overlap index is obtained as $O \approx 2$, and the upper bound of on-off switching events of appliances within the sampling interval is set to 2, i.e., $\delta = 2$. According to Equation (13), the local window size, i.e., the value of $w$ in Algorithm 1, is set to 3. Since the value of local window size is an estimation, in order to obtain more comprehensive performance evaluation for our method, we also vary the local window size in a range.

Table I summarizes some of the results from the two methods. Furthermore, Fig. 4 and Fig. 5 illustrate one of the outcomes from our appliance-driven method and the B-spline smoothing method, respectively.

From the results, we have the following interesting observations.

- Comparing to B-spline smoothing, our method performs much better in Precision, but worse in Recall. This shows that our method can identify the corrupted data more accurately, even though our output does not cover the completed set of all corrupted data. In addition, our appliance-driven method achieves a higher F-measure. F-measure reflects a balanced mean between precision and recall, indicating that our method has overall better performance.

- The performance of our method remains roughly the same when the local window size is beyond the minimum value estimated using Lemma 4. Further increase of the local window size does not bring clear performance gain but with a cost in longer running time. This suggests that our previous estimation on the minimum local window size for SLOA is appropriate.

7. Evaluation on Synthetic Data

To thoroughly test our method, we evaluate its performance using large-scale synthetic data that simulates a large number of appliances and much diverse energy patterns. With different synthetic datasets, we can also test the robustness of our method.

7.1 Load Data Generation via Monte Carlo Simulation

There is no standard model for the load curve data of a house, since the data actually results from a complex process related to human activities. We thus use the Monte Carlo simulation to generate the load data using the following method:

- Given the lowest appliance power ($P_{\text{min}}$) and the highest appliance power ($P_{\text{max}}$), the lower bound of an appliance ($p_l$) is a random variable uniformly distributed between $P_{\text{min}}$ and $P_{\text{max}}$. The upper bound of the appliance ($p_u$) is determined by a parameter called power range ratio ($r$) and is calculated by $p_u = \min\{p_l + \text{random}([0, rP_l]), P_{\text{max}}\}$, where random([0, rP_l]) returns a random number uniformly distributed in the range $[0, rP_l]$.
At a given sampling frequency, each appliance reports its current power value, which is a random number uniformly distributed between the appliance’s lower power bound and upper power bound. It reports 0 if its state is off.

In a sampling interval, the number of total on-off switch events follows a Poisson distribution with parameter λ. At the end of each sampling interval, the load data of the house is recorded as the aggregated power value of all appliances (i.e., the sum of all appliances’ load values).

To introduce some corrupted data and test the effectiveness of our method, we “corrupt” some data values by replacing them with random values uniformly distributed between the appliance’s lower power bound and upper power bound. Corrupted data are assumed to follow an exponential distribution with parameter μ. To introduce some corrupted data and test the effectiveness of our method, we “corrupt” some data values by replacing them with random values uniformly distributed between the appliance’s lower power bound and upper power bound. Corrupted data are assumed to follow an exponential distribution with parameter μ. To introduce some corrupted data and test the effectiveness of our method, we “corrupt” some data values by replacing them with random values uniformly distributed between the appliance’s lower power bound and upper power bound. Corrupted data are assumed to follow an exponential distribution with parameter μ. To introduce some corrupted data and test the effectiveness of our method, we “corrupt” some data values by replacing them with random values uniformly distributed between the appliance’s lower power bound and upper power bound. Corrupted data are assumed to follow an exponential distribution with parameter μ.

### 7.2 Corrupted Data Identification on Large-Scale Appliances

The parameters used to generate the synthetic data and the corrupted data are listed in Table 2.

We treat the bound on the total number of on-off switches in a sampling interval δ as a variable. To speed up the processing, we set the local window size to 1. The small local window size may not lead to the best performance of SLOA. However, as shown in our experimental results, even with this setting, our method already performs better than B-spline smoothing. For the B-spline smoothing method, the degree of freedom (df) is set as a variable and is trained when smoothing the synthetic data.

The performance results of our method and the B-spline smoothing method are summarized in Table 2. Fig. 4 and Fig. 5 illustrate one of the outcomes from our method and the B-spline smoothing method, respectively.

From the results, we can see that the our method works effectively on large-scale synthetic data. In particular, we find that the precision of our method increases with increase of δ, and can even reach 100%. This result indicates that our method can provide excellent correct identification when δ is large enough. Regarding the overall performance in view of F-measure, our method works better with a smaller δ value and outperforms B-spline smoothing clearly.

### 7.3 Identification of Consecutive Corrupted Data

In practice, we often meet the situation that all data within a
small time interval are corrupted or lost. Consecutive corrupted data poses a big challenge to regression-based methods, as will be illustrated in this subsection.

To introduce consecutive corrupted data, we replace the load data in a small time window as 0s, as shown in the upper part of Fig. 8. We then use our method and the B-spline smoothing method to test the data. Fig. 8 and Fig. 9 illustrate one outcome from our appliance-driven approach and the B-spline smoothing method, respectively.

From the results, we can see that our method does much better than B-spline smoothing for consecutive corrupted data identification. With δ = 5, our method can correctly identify all the corrupted data. On the other hand, even though we regulate the parameters for B-spline smoothing, it almost failed every time to identify even half of the corrupted data.

An interesting phenomenon can be found around the consecutive corrupted data in Fig. 9. There is an apparent trend with B-spline smoothing to fit the corrupted data. This is mainly because the B-spline smoothing method tries to fit the curve pattern and reduce the total bias error with global optimization, indicating that it cannot deal with consecutive corrupted data well.

8. ROBUSTNESS TESTING

One may question whether the performance of SLOA relies on a correct initial state vector, accurate information regarding appliances power ranges, and an accurate estimation on appliances’ on-off states. all of such information may be hard to obtain in practice. To answer this question, we test the robustness of SLOA. We use the synthetic data created using the same parameters in Table 2. We first disclose the test results and then explain the reasons.

8.1 Impact of the Initial State

For this test, we change the initial state of an appliance to a random 0-1 value, and perform multiple tests. Fig. 10 shows one of the outcomes. We find that, even with an incorrect initial state, our method can always recover to correct load data after a few steps. This result indicates that our SLOA method is robust against inaccurate initial power state setting.

8.2 Impact of Power Ranges
In practice, we may not precisely know the power ranges of appliances. Based on this consideration, we run extra simulations to test the robustness of our method when the power range information of appliances is inaccurate. We carry out two kinds of tests as follow.

- **Widen power range**: each appliance’s power range is widened by 5% or 10%, respectively, with the center power value, i.e., (upper bound − lower bound)/2, unchanged.

- **Shift & widen power range**: each appliance’s lower power bound is increased by 5% or 10%, and upper bound is increased by 5% or 10%, respectively. Note that the above operations will shift the center power value as well as widen the power range.

We do not consider the situation where the appliances’ power ranges are narrowed, since intuitively a user can always widen an appliance’s power range if s/he is not sure about the right values.

The test results are summarized in Table 4. The results clearly indicate that, with inaccurate or even wrong power ranges of appliances, our method can still manage to identify corrupted data with high precision.

### Table 4: Robustness tests with incorrect power ranges of appliances

|                  | Widen (5%) | Widen (10%) | Shift & Widen (3%) | Shift & Widen (10%) |
|------------------|------------|-------------|--------------------|---------------------|
| **Precision**    | m = 50, δ = 5 | m = 50, δ = 5 | m = 50, δ = 5      | m = 50, δ = 5       |
|                  | 83.55%     | 87.64%      | 91.33%             | 94.12%              |
| **Recall**       | 76.52%     | 65.26%      | 65.66%             | 42.11%              |
| **F-measure**    | 84.06%     | 67.74%      | 68.84%             | 58.18%              |

In real life, a lot of appliances are with similar or overlapped power range. In this sense, we indeed do not need to know the exact state for similar appliances as long as we can give a good approximation for their total consumption. In addition, due to the temporal sparsity of on-off switch events in the short sampling interval and the fact that only some appliances are on at any time instant, the negative impact of inaccurate power range estimation on one appliance can be offset by the negative impact of incorrect state estimation of another appliance. The offsetting is enforced au-

### Figure 10: Fast recovery of estimated load starting from a random initial state

### Figure 11: Difference between estimated state $S_e$ and real state $S_r$

#### 8.3 Impact of State Vector

We have seen that our method can give correct bounds for energy consumption most of the time. Accordingly, we might infer that the estimated states of the appliances should be the same with the real situation, or at least quite close.

In order to verify this conjecture, we calculate the difference (one-norm distance) between the estimated state $S_e$ and the corresponding real state $S_r$ at each time instance. Fig. 11 shows the result.

To our surprise, the estimated states are not close to the real states, and actually deviate remarkably from their real states. This shows that the solution to the CDIP problem is not unique but multiple, and our method can provide the right load data without need to always find the right states of appliances.

#### 8.4 Why Is SLOA Robust?

In real life, a lot of appliances are with similar or overlapped power range. In this sense, we indeed do not need to know the exact state for similar appliances as long as we can give a good approximation for their total consumption. In addition, due to the temporal sparsity of on-off switch events in the short sampling interval and the fact that only some appliances are on at any time instant, the negative impact of inaccurate power range estimation on one appliance can be offset by the negative impact of incorrect state estimation of another appliance. The offsetting is enforced au-
tomatically by the optimization objective function that minimizes the gap between the actual load data and the estimated value.

9. CONCLUSION AND FUTURE WORK

To answer the industrial call for improving quality of load data, we developed a new appliance-driven approach for corrupted data identification that particularly takes advantage of information available on the demand side. Our appliance-driven approach considers the operating ranges of appliances that are readily available from users’ manual, technical specification, or public websites. It identifies corrupted data by solving a carefully-designed optimization problem. To solve the problem efficiently, we developed a sequential local optimization algorithm (SLOA) that practically approach the original NP-complete problem approximately by solving an optimization problem in polynomial time.

We evaluated our method using both real trace data from a real-world energy monitoring system and large-scale synthetic data. Test results indicate that our method can precisely capture corrupted data. In addition, SLOA is robust under various test scenarios, and its performance is resilient to inaccurate power range information or inaccurate power state estimation.

Our method greatly augments the arsenal of existing load data cleansing tools to minimize human effort in identifying corrupted data. Yet, we ignore the privacy issues in this study. Even if the customers and the utility companies have an aligned common goal for accurate load data, some customers may be reluctant to collaborate due to privacy concerns. Our future research is to enhance our appliance-driven approach by developing privacy-preserving load data cleansing methods. In addition, how to replacing aberrant values and missing values is out of the focus of this paper, because this issue is relevant to utilities’ internal rules and thus requires human interaction. Considering various policies and methods for load data imputation will be our future work.

10. REFERENCES

[1] B. Abraham and A. Chuang. Outlier detection and time series modeling. Technometrics, 31(2):241–248, 1989.
[2] B. Abraham and N. Yatawara. A score test for detection of time series outliers. Journal of time series analysis, 9(2):109–119, 1988.
[3] C. C. Aggarwal. Outlier Analysis. Springer, 2013.
[4] R. J. Bolton, D. J. Hand, et al. Unsupervised profiling methods for fraud detection. Credit Scoring and Credit Control VII, pages 253–255, 2001.
[5] J. Chen, W. Li, A. Lau, J. Cao, and K. Wang. Automated load curve data cleansing in power systems. IEEE Transactions on Smart Grid, 1(2):213–221, 2010.
[6] S.-y. Chen, S.-f. Song, L. Li, and J. Shen. Survey on smart grid technology. Power System Technology, 33(8):1–7, 2009.
[7] CVX. Matlab software for disciplined convex programming. cvxr.com/cvx, accessed in July 2013.
[8] H. David. Robust estimation in the presence of outliers. Journal of the American Statistical Association, 88(423):782–792, 1993.
[9] D. L. Donoho. For most large underdetermined systems of linear equations the minimal norm solution is also the sparsest solution. Communications on pure and applied mathematics, 59(6):797–829, 2006.
[10] EIA. Electric power annual 2012. http://www.eia.gov/electricity/annual/yrsrc=Electricity-f4, accessed in Jan. 2014.
[11] EPA. A tool of home product finder by energy star. http://www.energystar.gov/productfinder/, 2013.
[12] H. Farhangi. The path of the smart grid. Power and Energy Magazine, IEEE, 8(1):18–28, 2010.
[13] T. S. Ferguson. On the rejection of outliers. In Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, volume 1, pages 253–287, 1961.
[14] A. J. Fox. Outliers in time series. Journal of the Royal Statistical Society. Series B (Methodological), pages 350–363, 1972.
[15] U. Gather. Testing for multisource contamination in location/scale families. Communications in Statistics-Theory and Methods, 18(1):1–34, 1989.
[16] E. M. Knox and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In Proceedings of the International Conference on Very Large Data Bases, 1998.
[17] H.-P. Kriegel and M. Pfeifle. Density-based clustering of uncertain data. In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, pages 672–677. ACM, 2005.
[18] J. Lang. A bad case of smart meter regret. http://www.cloverdalereporter.com/news/189923521.html, accessed in Jan. 2014.
[19] G. M. Ljung. On outlier detection in time series. Journal of the Royal Statistical Society. Series B (Methodological), pages 559–567, 1993.
[20] G. Mateos and G. B. Giannakis. Robust nonparametric regression via sparsity control with application to load curve data cleansing. IEEE Transactions on Signal Processing, 60(4):1571–1584, 2012.
[21] J. McMahon. Big data from smart grid tells utilities more than they want to know. Forbes, September 2013.
[22] S. Moneo. Bc hydro pulls smart meters for testing. http://www.theglobeandmail.com/news/british-columbia/bc-hydro-pulls-smart-meters-for-testing/article534384/, accessed in Jan. 2014.
[23] C. News. BC Hydro outlines smart meter refusal cost. September 13, 2013.
[24] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In ACM SIGMOD Record, pages 427–438. ACM, 2000.
[25] W. Schmid. The multiple outlier problem in time series analysis. Australian Journal of Statistics, 28(3):400–413, 1986.
[26] J. Stragier, L. Hauttekeete, and L. De Marez. Introducing smart grids appliances. In Innovative Technologies for an Efficient and Reliable Power System Technology, pages 190–197, 2012.
[27] J. W. Tukey. Exploratory data analysis. Reading, MA, 231, 1977.
[28] M. Weiss, A. Helfenstein, F. Mattern, and T. Staake. Leveraging smart meter data to recognize home appliances. In 2012 IEEE International Conference on Pervasive Computing and Communications (PerCom), pages 190–197, 2012.

Appendix A: Proof of NP-Completeness of CDIP

10.1 Preparation

First, we introduce a tree structure \( T = (M, N) \), which is a complete \( M \)-ary tree with height of \( N \), i.e., every internal node has exactly \( M \) children and all leaves have the same depth of \( N \). Furthermore, each edge \((i, j)\) of \( T \) has a non-negative cost \( c(i, j) \), which will be defined later.

Assume that load data \( \{y_1, y_2, \ldots, y_n\} \) is generated by \( m \) appliances with the initial state vector \( S_0 \). Also assume that the upper bound on the total number of on-off switches within a sampling interval is \( \delta < m \). We can build the following tree:

Step 1. Set \( S_0 \) as the root of the tree.
Step 2. Set the children of root as all possible states that can be transited from $S_0$, with the constraint that the total number of on-off switches is no larger than $\delta$. Therefore, we can add $M$ children to the root, where $M = C_{m}^0 + C_{m}^1 + \cdots + C_{m}^3$.

Step 3. For each node of the tree, set its children as all possible states ($M$ states) that can be transited from it.

Step 4. Repeat Step 3) from $t = 1$ to $n$. At the end, we obtain $T = (M, N)$, where $N = n$.

Step 5. Set the cost of edge $(i, j)$ as $c(i, j) = |v|$, where $v$ is obtained by solving the optimization problem:

$$\text{minimize } |v|, \text{ subject to } (P^T_i S_j - v) / f \leq v_i \leq (P^T_i S_j + v) / f.$$ 

Thus, we can translate CDIP into the problem of finding the minimum-cost path in $T(M, N)$ from the root to a leaf. Equivalently, we need to answer the following question: given a constant $k$, is there a path in $T(M, N)$ from the root to a leaf with total cost no larger than $k$? In the following, we call a path from the root to a leaf in the tree as a full path.

With the notation above, CDIP can be re-formulated as

$$\text{CDIP} = \{(T, c, k) : T = (M, N), c \text{ is the cost function}, k \in \mathbb{R}^+ \text{, and } T \text{ has a full path with cost } \leq k \}.$$ 

We next reduce a well-known NP-complete problem, the Traveling Salesperson Problem (TSP) to CDIP. TSP can be formulated as

$$\text{TSP} = \{ (G, c', k) : G = (V, E) \text{ is a complete graph}, c' \text{ is the cost function}, k \in \mathbb{R}^+ \text{, and } G \text{ has a Hamiltonian cycle with cost } \leq k \}.$$ 

### 10.2 Proof

We complete the proof in two steps: firstly we show that CDIP is NP; then, we prove that CDIP is NP-hard by showing $\text{TSP} \leq_p \text{CDIP}$, i.e., there exists a reduction from TSP to CDIP.

- **a. CDIP is NP**
  - **Certificate:** A path of $T$.
  - **Algorithm:**
    - Check that the path is full, i.e., the path starts from the root and ends at a leaf.
    - Sum up the edge costs along the path and check if it is no larger than $k$.
  - **Polynomial Time:** We need $N$ steps to check the fullness of path and obtain the total cost.

- **b. CDIP is NP-hard**
  - Firstly, we develop an algorithm $F : (G, c', k) \rightarrow (T, c, k)$, i.e., $G$ and $c'$ in TSP can be transferred to $T$ and $c$ in CDIP as follow:
  
  **Step 1.** Choose any node of $G$ as the root of $T$;

  **Step 2.** For each leaf node of the current tree, add its children as all the other nodes of $G$. Since $G$ is a complete graph, we can add $|V| - 1$ children to each leaf node, where $|V|$ is the number of nodes in $G$.

  **Step 3.** Repeat Step 2 for $|V|$ times. At the end, we build $T = (M, N)$, where $M = |V| - 1$ and $N = |V|$.

  **Step 4.** Set the cost of edge $(i, j)$ in $T$, $c(i, j)$, as follows:
  
  (a) Initialization: $c(i, j) = c(i, j')$, where $c(i, j')$ is the edge cost in $G$.

  (b) For each edge $(i, j)$ of $T$ where $j$ is a non-leaf node, if $j$ has appeared in the path from the root (including the root) to $i$, i.e., $j$ is an ancestor of $i$ in the tree already, replace $c(i, j) = \infty$.

  (c) For each edge $(i, j)$ of $T$ where $j$ is a leaf node, if $j$ is not the same as the root node, replace $c(i, j) = \infty$.

  To help understand the construction of $T$ with $G$, Fig. 12 show an example with three nodes in $G$.

  - Secondly, it is easy to see that $F$ takes $O(N^2)$ running time.

  - Thirdly, we show that

  $$\langle G, c', k \rangle \in \text{TSP} \iff \langle T, c, k \rangle \in \text{CDIP}.$$ 

  $$\text{\((\Rightarrow )\)}$$

  $G$ has a Hamiltonian cycle with cost $\leq k$.

  $\Rightarrow$ there exists a full path in tree $T$ with cost $\leq k$.

  (Note that there will be no internal node along the path occurring more than once, otherwise the cost will be infinite with operation in $\text{Step 4}$)

  $$\text{\((\Leftarrow )\)}$$

  $T$ has a full path with cost $\leq k$.

  $\Rightarrow$ there exists a traverse instance in its corresponding graph $G$ with cost $\leq k$.

  (Note that based on the tree construction procedure, only the full paths starting and ending at the same node can have a cost no larger than $k$, because other paths have a cost of infinity.)

  $\Rightarrow$ so $G$ has a Hamiltonian cycle with cost $\leq k$.

### Appendix B: Computational Complexity

For problem (Equation 5), the second constraint means that during a sampling interval, there are at most $\delta$ out of $m$ appliances that can change their states. This results in $\binom{m}{0} + \binom{m}{1} + \cdots + \binom{m}{\delta}$ feasible solutions. Therefore, the total number of possible state sequences is $M^m$, where $M = \binom{m}{0} + \binom{m}{1} + \cdots + \binom{m}{\delta}$. Thus, the computational complexity of problem (Equation 8) is $O(M^m)$, which is exponential.

As to problem (Equation 10), the whole search space is split into $n$ local windows of size $w$, and the optimization is confined within the local window. Given $S_{i-1}$, there are at most $\binom{m}{0} + \binom{m}{1} + \cdots + \binom{m}{\delta}$ instances. We have to traverse all the instances to find the one that minimizes $v_i$ in each step of a local optimization. Therefore, the computational complexity to find a local optimal solution with $w$ steps is $O(M^m)$, where $M = \binom{m}{0} + \binom{m}{1} + \cdots + \binom{m}{\delta}$.

Hence, after applying SLOA in each of the $n$ local windows, the total computational complexity to obtain the final solution is $O(n \cdot M^m)$. Considering that the number of appliance $m$ is a constant value and $w$ is also a small constant, SLOA cuts down the computational complexity of the original problem from exponential to polynomial.