An Improvement of Controlling Quality of Large Scale Water-Level Data in Thailand

Nuttapon Pattanavijit
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Phayathai Road, Pathumwan
Bangkok, Thailand 10330
Email: nuttapon.p@student.chula.ac.th

Peerapon Vateekul
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Phayathai Road, Pathumwan
Bangkok, Thailand 10330
Email: peerapon.v@chula.ac.th

Kanoksri Sarinnapakorn
Hydro and Agro Informatics Institute
Ministry of Science and Technology
Thailand, 10400
Email: kanoksri@haii.or.th

***Abstract*—Extremely change in precipitation level such as water level can cause severe damage. In order to acknowledge changes, Hydro and Agro Informatics Institute has installed telemetry system across Thailand to collect and analyze precipitation level. However, to use its data in researching, incorrect data must be filtered. In previous work, we successfully detect various problems in water level data accurately. Still, missing pattern algorithm and outliers algorithm proposed in the previous work have run-time complexity up to $O(n^2)$, which are not quite suitable for large-scale data. In this paper, we aim to improve these algorithms, mainly focus on run-time complexity. As a result, we can speed up the outlier algorithm to $O(n)$ and the missing pattern to $O(n \log n)$. Moreover, compared with the previous work, we measure actual running time of our algorithms and found that they significantly help reduce the running time.**

***Index Terms*—Purus, Vehicula, Ullamcorper, Dolor, Sollicitudin**

I. Introduction

Thailand has faced many dreadful climate-related disaster including floods, droughts, and tropical cyclones. They have happened year after year and cause severe loss. For example, In 2011, seasonal flooding results in US$ 45.7 billion damage to Thailands economy, which is 1.1% of the countrys GDP [1]. Also, the climate-related disaster bring difficulties to agriculture and industry, which are backbones of Thailand economy.

To handle these disaster, the government established many organization to cooperatively counteract with it. One of the organization is Hydro and Agro Informatics Institute (HAII). HAIIs main focus is to research and utilize knowledge in agricultural and water resource management to confront the climate disaster [2].

In order to conduct researches, they have collected precipitation data by installing telemetry systems across Thailand. The telemetry system is a device that is used to collect physical, chemical, and biological data from its various sensors [3]. For instance, rivers water level, rainfall level, humidity, and temperature can be measured. Today, HAII have already installed over 800 telemetry systems. The data collected from the telemetry systems is sent back to HAIIs server by using GPRS cellular network [4].

Sometimes, incorrect data is reported from the telemetry systems such as minus value of cumulative rainfall level, or rivers water level suddenly changed from one level to another, which are not possible. In addition, data loss can also be occurred due to poor cellular network in rural area. These inconsistent data is not suitable to be used in researches since it might lead to inaccurate results.

To filtered out incorrect data efficiently, our previous work called Controlling Quality of Water-Level Data in Thailand proposed algorithms to detect anomalies in water level data, which includes outliers, inhomogeneity, and missing pattern algorithm [5]. Example of anomaly data detected by these algorithms is illustrated in figure 1, figure 2, and figure 3. However, when we implement this algorithm to use with real precipitation level data, a problem arise. Since HAII use 'R' as a main programming languages for all data analytic tasks, implementing outliers and missing pattern algorithm may leads to $O(n^2)$ in complexity. This inefficiency is originated from two main points. First, The bottleneck of both algorithms is occurred from clustering algorithm. The previous work use *DBSCAN* [6] as the clustering algorithm which have $O(n^2)$ complexity if it is a naive implementation and $O(n \log n)$ if it is implemented using special data structures that support fast region query such as *R*-Tree* or *k-d Tree* , which seem to be much more complex. Second, Our last work is implemented in HAII using `R` library called `fpc` which contains *DBSCAN* library that have $O(n^2)$ complexity [7]. Thus, the overall algorithms complexity become $O(n^2)$.

This paper aim to propose an improvement in outliers and missing pattern algorithms for large-scale water level data. Our main goal is to refine both algorithm to be simple, yet expeditious, while bring out the same results as our previous work. We have reduce running time of outliers algorithm from $O(n^2)$ or $O(n \log n)$, depends on data structures, to $O(n)$. Also, missing pattern is decresed from $O(n^2)$ or $O(n \log^2 n)$ to $O(n \log n)$. This can be done by imitating the clustering result with more simple and fast approach which we will discuss later. The experimental result revealed that our approach significantly cuts down the overall running time in large-scale data.

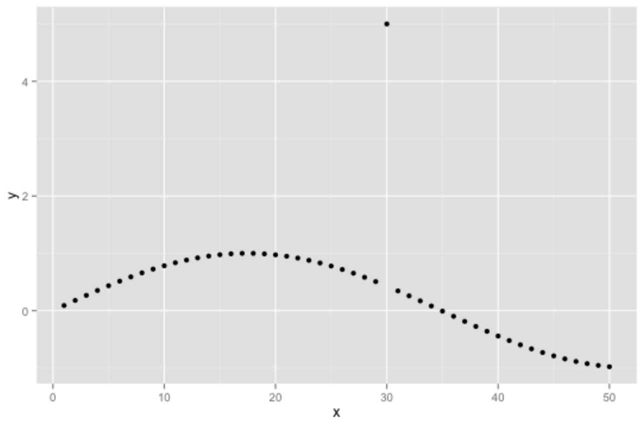

Fig. 1. Example of data with outliers.

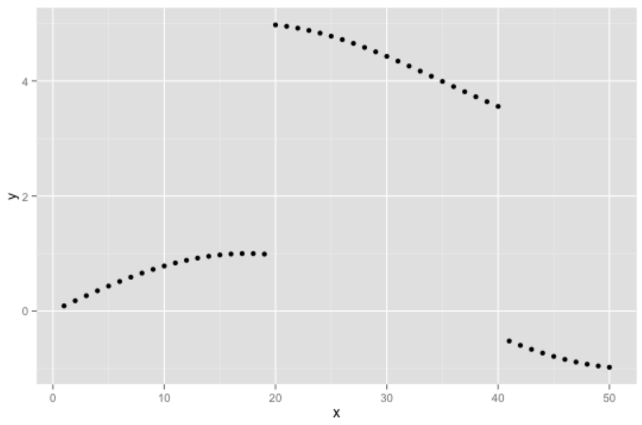

Fig. 2. Example of data with inhomogeneity.

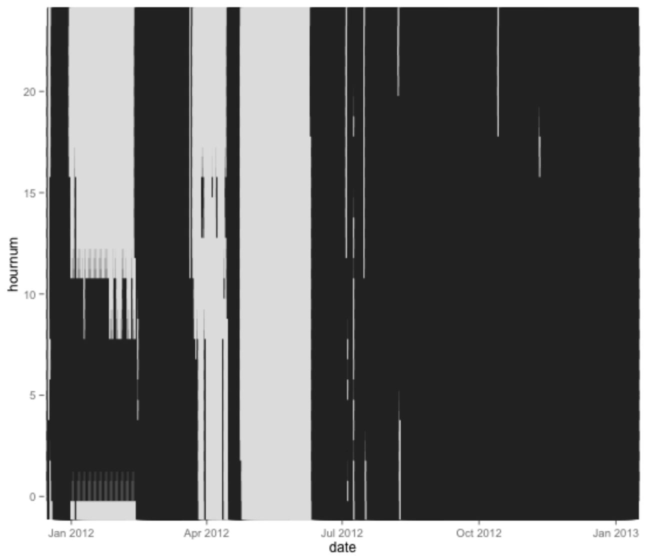

Fig. 3. Example of data with missing pattern. X-axis represent date of the data. Y-axis indicate hour of day of the data. The lighter area shows date and hour where data was missing.

The paper is organized as follows. Section 2 describes how data was collected at HAII. Section 3 recaps our previous work and points out problems. Our approach to refine the both algorithms are illustrated in Section 4. Experiment results are shown in Section 5. Last, Section 6 delivers a conclusion.

II. Large-Scale Water Level Data

Currently, Hydro and Agro Informatics Institute (HAII) have already installed over 800 telemetry system across Thailand. **Add Map Figure!**Figure 4 shows example of location where telemetry systems were installed. Each system send data from its many sensors back to central database every 10 minutes via cellular network. From these numbers, we can estimate that there are 3.45 million records of data added to database every month. Since HAII have collected precipitation data for over 5 years, we can assume that we are dealing with approximately 207 million records. In previous paper, we use water level data for both outliers and missing pattern algorithms. Moreover, because of the nature of water level in the river, water level's value should be continuous without sudden change. Moreover, data is stored in timestamp order.

III. Previous Data Quality Management

Our previous work proposed two algorithms to control quality of water-level data, outliers and missing pattern. Both of the algorithms apply clustering technique in order to detect anomaly data and pattern of data. DBSCAN was selected as a clustering algorithms. Note that for the time dimension, 10 minutes will be counted as a distance of 1 unit because data points is captured in 10 minutes interval.

A. Outliers Detection Algorithm

In previous work, we detect outliers by directly applies *DBSCAN* to water-level data. *DBSCAN*'s required parameters, minimum number of points *minpts* and distance epsilon *eps*, is set to 3 and 1.05 respectively in order to classify data which have extreme difference in value compared to adjacent data as a noise. Figure 4 illustrates pseudocode of outliers detection algorithm.

```
1: procedure DETECT-OUTLIERS($d$)
2:     $minpts \leftarrow 3$
3:     $eps \leftarrow 1.05$
4:     $C \leftarrow \emptyset$                    ▷ Set of cluster of data index
5:     $N \leftarrow \emptyset$                    ▷ Set of noise data index
6:     $C, N \leftarrow$DBSCAN($d, minpts, eps$)
7:     return $N$
8: end procedure
```

Fig. 4. Pseudocode of outliers detection algorithm.

From pseudocode in Figure 4, it is obvious that running time of DETECT-OUTLIERS is based on DBSCAN. Thus, we can conclude that DETECT-OUTLIERS complexity is $O(n^2)$ or $O(n \log n)$ – depends on the data structure inside DBSCAN.

B. *Missing Pattern Algorithm*

Proposed in previous work, missing pattern is used to detect pattern of missing data. **Add some conjunction sentencs here!** First, it convert water-level data into frequency domain, which count how many data is missing during each hour $(f_0, f_1, \ldots, f_{23})$. Second, we use DBSCAN to analyze the frequency. If there are more than one cluster detected, there must be at least two hours having significantly different missing frequency (f_i). Then, we ... . Next, divided-and-conquer technique was applied to find missing pattern in subset of data. Last, we merge overlapped missing result together.

To be easier to explain, we rewrite its pseudocode from previous work in figure 5. Result of MISSING-PATTERN procedure is returned as a list of tuples indicating start date and end date of pattern. Also, list is always sorted by start date of tuples.

```
1:  procedure MISSING-PATTERN($d$)
2:      if $end - start \le minDataReq$ then
3:          return $\emptyset$                                  ▷ Too few data
4:      end if
5:      $f \leftarrow$ HOURLY-MISSING-FREQUENCY($d$)
6:      $minpts \leftarrow 1$
7:      $eps \leftarrow \frac{|d|}{2}$
8:      $C, N \leftarrow$ DBSCAN($f, minpts, eps$)
9:
10:     $haveOverallMP \leftarrow |C| \ge 2$
11:
12:     $P_l \leftarrow$ MISSING-PATTERN($d_1 \ldots d_{\frac{|d|}{2}}$)
13:     $P_r \leftarrow$ MISSING-PATTERN($d_{\frac{|d|}{2}+1} \ldots d_{|d|}$)
14:     $P \leftarrow$ COMBINE-RESULT($P_l, P_r, haveOverallPattern$)
15:
16:     $mergeGap \leftarrow 15$                                ▷ 15 days
17:     return MERGE-OVERLAP-PATTERN($P, mergeGap$)
18: end procedure
```

Fig. 5. Pseudocode of missing pattern algorithm.

From figure 5, HOURLY-MISSING-FREQUENCY on line 5 can be done in $O(|d|)$ by using counting sort with 24 bucket representing missing frequency of each hour of day $(f_0, \ldots, f_{23})$. COMBINE-RESULT on line 14 is done in $O(1)$ by checking 8 possible cases. And, MERGE-OVERLAP-PATTERN could be executed in $O(|d|)$. Since the tuples in P is sorted and not literally overlap, we can sequentially check each adjacent pair of tuples whether it should be merged. We can conclude that DBSCAN is the bottleneck in each recursive call.

Since it uses divide-and-conquer, We can compute the complexity of by using *master method*, which the complexity of algorithm can be represent in:

$$T(n) = aT(\frac{n}{b}) + f(n)$$

If DBSCAN have $O(n^2)$ complexity, the complexity of MISSING-PATTERN would be:

$$T(n) = 2T(\frac{n}{2}) + O(n^2)$$

By applying *master theorem*, MISSING-PATTERN's complexity become $O(n^2)$.

On the other hand, If DBSCAN have $O(n \log n)$ complexity, the complexity of MISSING-PATTERN would be:

$$T(n) = 2T(\frac{n}{2}) + O(n \log n)$$

By applying *master theorem*, MISSING-PATTERN's complexity become $O(n \log^2 n)$.

IV. PROPOSED IMPROVED ALGORITHMS

Our proposed algorithms aim to solve the bottleneck in both algorithms in order to make them faster. To do that, we try to create a new clustering algorithm which can imitate *DBSCAN*'s result but with more efficiency, by using some facts about precipitation data and the algorithms themselves.

First, since the data was captured from a sensor, we can assume that at time t, there will be only one d_t. This fact implies that there is no need to search for adjacent points in two dimension.

Second, outliers algorithm and missing pattern algorithm produce best result when *minpts* $= 3$ and *minpts* $= 1$ respectively. Let C_k be the set of cluster of data index. We can say that if *minpts* ≤ 3, following properties are always true:

1) if i is in C_k (k^{th} cluster) and $|d_{i+1} - d_i| \le$ *eps*, then $i + 1$ is in C_k too.
2) $|C_k| \ge$ *minpts* for all k

The first property is true because if *minpts* $= 1$, at least d_i can form its own cluster and add d_{i+1} to its cluster. If *minpts* $= 2$ and i is in C_k, there must be another data point $j \in C_k, j < i$ within distance *eps*. So i can be add freely into C_k. Last, if *minpts* $= 3$, since i is already in C_k, there must be another data point $j \in C_k, j < i$ within distance *eps*. So, if $|d_{i+1} - d_i| \le$ *eps*, d_i will satisfy *minpts* $= 3$ (including itself) condition and $i + 1$ can be added into C_k.

For the second property, assume that C_k is the smallest cluster possible, C_k must have at least one core data point in order to form a cluster. Let d_k be the only core data point. d_k must have at least *minpts* data point within distance *eps* (including itself). So, we can conclude that $|C_k| \ge$ *minpts*.

With this two properties, we can create an algorithm which can imitate *DBSCAN*'s result for *minpts* ≤ 3. Figure 6 illustrates its pseudocode.

```
procedure LINEAR-CLUSTER($d, minpts, eps$)
    $C \leftarrow \emptyset$
    $N \leftarrow \emptyset$
    $c \leftarrow \{1\}$                                   ▷ Temporary Cluster
    for $i \leftarrow 2, |d|$ do
        if $|d_i - d_{i-1}| < eps$ then
            $c \leftarrow c \cup \{i\}$
        else
            if $|c| \geq minpts$ then
                $C \leftarrow C \cup \{c\}$
            else
                $N \leftarrow N \cup c$
            end if
            $c \leftarrow \emptyset$
        end if
    end for

    if $|c| \geq minpts$ then
        $C \leftarrow C \cup \{c\}$
    else
        $N \leftarrow N \cup c$
    end if

    return $C, N$
end procedure
```

Fig. 6. Pseudocode of new clustering algorithm.

From figure 6, the pseudocode work by following steps. First, it creates a temporary cluster an assume that d_i is in cluster (start from d_1). Then, it checks the distance to its next adjacent data point (d_{i+1}). If the distance is less than eps, put d_{i+1} in the same temporary cluster. Otherwise, create new temporary cluster and assume that d_{i+1} is in it. Before creating new temporary cluster, we determine whether the previous temporary cluster can be formed into real cluster or not by counting data points inside it. If it has more than $minpts$ points, it can be formed as a real cluster. If not, all data point inside it should be considered as noise. Thus, the algorithm obviously has $O(n)$ complexity since it contains only one for-loop from 2 to $|d|$.

Therefore, we can replace *DBSCAN* in outliers algorithm and missing pattern algorithm with LINEAR-CLUSTER to reduce it's overall complexity. The outliers algorithm is become $O(n)$ in complexity.

V. EXPERIMENTS

Lorem Ipsum

VI. CONCLUSION

Lorem Ipsum

REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.