Similar Data Detection for Cooperative Spectrum Monitoring in Space-Ground Integrated Networks

Zhijuan Hu, Danyang Wang, Qifan Fu, and Zan Li
State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, 710071, China
Email:zjhu@stu.xidian.edu.cn; dywang@xidian.edu.cn; fuqifanfqf@163.com; zanli@xidian.edu.cn

Abstract*—Space-ground aided cooperative spectrum monitoring, which combines the benefits of satellite components and terrestrial components for improving monitoring accuracy and enlarging monitoring area, has been becoming an emerging application of the space-ground integrated networks (SGIN). However, a short transmission window is usually provided for satellite components to connect with ground gateway, which means only a limited transmission time is allowed for the satellite component to upload the collected spectrum data. On the other hand, lots of redundancy may exist among the spectrum data collected by a single sensor during one collection period, which may further reduce the data uploading efficiency. In this paper, we investigate the similar data detection which is a matching problem for comparing two data, and it is important to the following data compression for improving data uploading efficiency. Firstly, the definition of the sharing fragment set is given. Then a metric method is presented to measure the redundancy of one data with respect to another data. We propose a Sharing Fragment Set (SFS) algorithm that can select a good sharing fragment set. Theoretical analysis proves that the proposed SFS algorithm is well suited to determine the redundancy between datas. In addition, we conduct an experiment based on the randomly produced synthetic dataset. Numerical results shows that the SFS algorithm performs better in selecting sharing fragment set compared with the Greedy-String-Tiling (GST) and simple greedy algorithm.

I. Introduction

With the rapid development of wireless communications, terrestrial networks (e.g. Long-term Evolution (LTE) networks) can provide high-speed and high-reliability services for the users in urban area. However, in rural areas, there still exist large populations that can not be served by cellular networks due to economic and technological reasons. The space-ground integrated networks (SGIN) are expected to exploit the benefits of satellites, high-altitude platforms (HAPs) as well as the terrestrial wireless communication networks, for providing wide-range and seamless networking services [1]–[3]. Unfortunately, radio spectrum resources in space-air networks are often scarce. Efficient utilization and management of spectrum resources between the space network and terrestrial network can help to release the pressure of the spectrum scarcity. As known to all, spectrum monitoring is the premise foundation to facilitate the efficient spectrum management and utilization.

The original task of spectrum monitoring techniques is to help the cognitive systems to be aware of whether the spectrum band is vacant or not. While the current demand for spectrum monitoring expands from a single frequency element to frequency, time, space, signal, power and other multi-dimensional elements, the spectrum monitoring has changed from the traditional data collection to data analysis [4], signal process [5], signal localization and tracking [6], etc. With the proliferation of satellite communication system, space-ground cooperative spectrum monitoring system has become the inevitable trend in the future [7]–[9], where the sensors equipped on satellite can provide a wide monitoring coverage and the rich deployed sensors on ground can provide an accurate monitoring performance. Unfortunately, a short transmission window is usually allowed for satellite sensors to connect with ground gateway, which means only a limited transmission time can be used for the satellite sensor to upload the spectrum data. On the other hand, there exist redundancy among the spectrum data collected within a data collection period by a single sensor. In order to obtain valuable spectral information efficiently, the similar data detection seems especially urgent. A good method of similar data detection can find more sharing fragments so as to compress file compactly and improve the data transmission rate effectually.

In the exist literatures, three typical algorithms have been studied to find out the sharing fragments exactly between two file [10], [11] in byte level. Greedy-String-Tiling (GST) algorithm [12], [13] is usually used in similarity detection systems to find the matching fragments, which consists of two phases [14]. In the first phase, the longest contiguous sharing fragments between two file are searched. In the second phase, all the sharing fragments longer than or equal to the maximal length are marked. Then the marked fragments are forbidden in the next iteration. If an unmarked fragment is repetitive with part of a marked fragment, it will be ignored, so GST algorithm can not find a good set consisting of sharing fragments. Running Karp-Rabin Greedy String Tiling (RKR-GST) [15] is an improvement of GST as it imports a rolling hash function. The computational complexity of RKR-GST algorithm has reduced dramatically compared to GST algorithm, while both two algorithms have the same ability in finding sharing fragments. A simple greedy algorithm of differential compression [16]–[18], based on block move model [19], can also detect sharing fragments. Since the simple greedy algorithm pays more attention to the longest sharing fragments and loses the short ones, it is not good at determining sharing fragment set between two data.

In this paper, we study the similar data detection method, to construct a good sharing fragment set from two data D_i^t and D_i^{t+1}. According to the location, two fragments in a

978-1-7281-0960-2/19/$31.00 ©2019 IEEE

data are separated, overlapped or contained. Let LS denote a sharing fragment set, and we define $Len(LS)$ as the total length of all separated sharing fragments in D_i^{t+1} to evaluate a sharing fragment set. Usually there are not only one LS, so $Len(LS)$ have various values. A good sharing fragment set must have the maximal $Len(LS)$. Then we propose the Sharing Fragment Set (SFS) algorithm to extract a good LS of D_i^{t+1} with respect to D_i^t. The proposed SFS algorithm has three parts: i) finding a sharing fragment set LS whose sharing fragments in D_i^{t+1} are separated or overlapped; ii) merging each overlapped sequences in D_i^{t+1} in terms of LS and then finding a best choice ; and iii) making a sharing fragment set of all the separated sharing fragment and best choices. We prove that a good redundancy of D_i^{t+1} with reference to D_i^t can achieved by leveraging the SFS algorithm. At last, we conduct an experiment by randomly producing D_i^t and D_i^{t+1}. By comparing the results from SFS algorithm with that from the existing algorithms, we can obtain that SFS algorithm can pick out better sharing fragment set between D_i^t and D_i^{t+1}.

The main contributions in this manuscript are summarized as follows:

- We define sharing fragment set rigorously, and use location to identify fragment uniquely so that we can understand sharing fragment set better.
- SFS algorithm is proposed to identify a good sharing fragment set between two data. The evaluation for the redundancy of D_i^{t+1} in reference to D_i^t is conducted by adding the length of the separated sharing fragments in D_i^{t+1}. In order to compute incrementally and produce fewer collisions, Karp-Rabin function is used to create a hash table. Theoretical analysis reveals that the SFS algorithm performs well in measuring inter-data redundancy.
- At last, SFS algorithm is realized by programming. The synthetic dataset for experiment is produced randomly. Numerical results indicate that the proposed SFS algorithm can find better sharing fragment set than the GST and simple greedy algorithm.

The rest of this paper is organized as follows. Section II presents a system structure of space-ground aided cooperative spectrum monitoring. Section III proposes the measurement of similar data detection algorithm and the outline of SFS algorithm and then analyses it in theory. The numerical results from experiments carrying on synthetic D_i^t and D_i^{t+1} are provided in Section IV. Finally, the conclusion of the paper is presented in Section V.

II. System Structure

We consider a cooperative spectrum monitoring system which combines the space components and ground components together, as described in Fig. 1. Detailed, the system network structure is composed of two main fragments: the space network and ground network. These two fragments can work cooperatively, due to the integrating heterogeneous networks among these two fragments, from which a hierarchical wireless network can be easily built as follows.

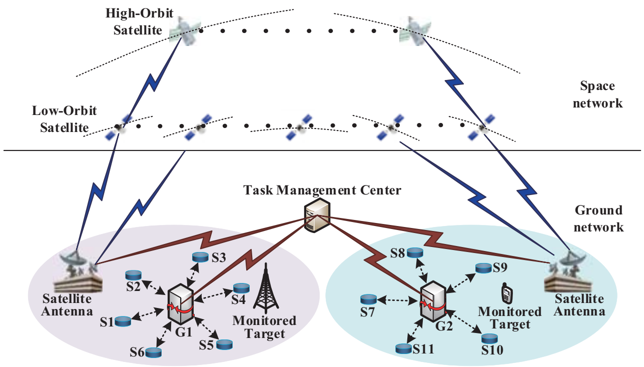


Fig. 1. Space-ground cooperative spectrum monitoring scenario.

- Space Network: The space network comprises high and low orbit satellites. These satellites can collect spectrum data and transmit them to satellite antennas belonging to the ground network. The satellites are tasked to provide a wide monitoring coverage.
- Ground Network: The ground network consists of task management center, gateway nodes and sensor nodes. The ground sensor nodes can be deployed densely due to the low cost, and these rich deployed sensors on ground are tasked to provide an accurate monitoring performance.

The gateway nodes and satellite antennas receive spectrum data and compress data and then transmit to the task management center. At last, task management center realizes human-machine interaction and displaying. By using a variety of network interconnection, this system network has flexible access to different networks at the same time inter-operationally, without limitation of a particular network paralysis.

The similar data detection is carried out at gateway node as shown in Fig. 2. Usually, the data perceived by the same sensor node in the previous moment and the current have a lot of correlation. That is, if D_i^t and D_i^{t+1} are the data collected by sensor i at time t and $t+1$, then there will be many common parts between D_i^t and D_i^{t+1}. After a gateway node receiving the D_i^t and D_i^{t+1}, a good similar data detection algorithm can find more sharing fragments in D_i^{t+1} reference to D_i^t. This provides convenient conditions for compact data compression. And as a result, the data transmission rate from gateway node to task management center is improved.

Our purpose is to construct a good similar data detection algorithm so as to get a sharing fragment set which can well represent the redundancy of D_i^{t+1} with respect to D_i^t.

III. Similar Data Detection and Evaluation

A. Measurement of Redundancy

Before describing the SFS algorithm, we introduced some definitions in assist.

Definition 1. Given D_i^t and D_i^{t+1} are data collected at time t and $t+1$, and d_i^t and d_i^{t+1} are the fragments of D_i^t and D_i^{t+1}, respectively. If $d_i^t = d_i^{t+1}$ for each $(d_i^t, d_i^{t+1}) \in LS$, LS is a

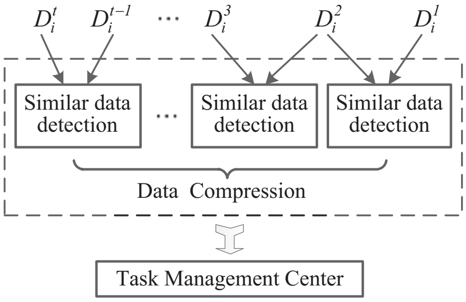


Fig. 2. Data preprocess at gateway node.

sharing fragment set while d_i^t and d_i^{t+1} are sharing fragment in D_i^t and D_i^{t+1}.

Definition 1 shows the definition of sharing fragment set and sharing fragment. Given D_i^t and D_i^{t+1}, there are many sharing fragment sets. Our purpose is to find a good sharing fragment set which can represent the redundancy between two data, which can be divided into two parts. One is how to measure the redundancy of D_i^{t+1} with reference to D_i^t. Another is how can we construct an algorithm to get the good sharing fragment set. In order to solve the former, we give Definition 2 as follows.

Definition 2. Given D_i^t is a data collected at time t, $d_1 = D_i^t[f_1, f_1']$ and $d_2 = D_i^t[f_2, f_2']$ are two fragments in D_i^t where f_1 and f_1' are the beginning and end location of d_1 in D_i^t while f_2 and f_2' are that of d_2, there is

1) If $f_1' < f_2$, d_1 and d_2 are separated. This case is denoted as $d_1 \asymp d_2$;
2) If $f_2 \le f_1' < f_2'$, d_1 and d_2 are overlapped. This case is denoted as $d_1 \uplus d_2$;
3) If $f_1 \le f_2 < f_2' \le f_1'$, d_1 contains d_2. This case is denoted as $d_1 \odot d_2$;

Definition 2 shows three types of relationship of two fragments of a data. From which, we can get Lemma 1 as follow.

Lemma 1. Given D_i^t is a data collected at time t, d_1 and d_2 are two fragments in D_i^t , one and only one of the following cases can hold:

1) $d_1 \asymp d_2$; 2) $d_1 \uplus d_2$; 3) $d_1 \odot d_2$.

Given two fragments, we determine their relationship by the beginning and end location. Since fragments in different position may have the same content, judging the relationship of two fragments by position is more reasonable than that by content. Then we introduce Definition 3 to measure the redundancy of D_i^{t+1} in reference to D_i^t.

Definition 3. Given two data D_i^t and D_i^{t+1}, and a sharing fragment set $LS = \{(d_1^t, d_1^{t+1}), \ldots, (d_k^t, d_k^{t+1})\}$. To measure the set LS, we define $Len(LS)$ as:

$$Len(LS) = \sum_{i=1}^{k} len(d_i^{t+1}) \tag{1}$$

Let

$$\begin{aligned}\Omega(D_i^t, D_i^{t+1}) = \{LS \mid \forall(d_i^t, d_i^{t+1})\ and\ (d_j^t, d_j^{t+1}) \in LS,\\ [(d_i^t \asymp d_j^t) \vee (d_i^t \uplus d_j^t) \vee ((d_i^t \odot d_j^t)] \wedge\\ (d_i^{t+1} \asymp d_j^{t+1}) = 1\end{aligned} \tag{2}$$

Meanwhile, we introduce

$$L_h(D_i^t, D_i^{t+1}) = max\{Len(LS) \mid LS \in \Omega(D_i^t, D_i^{t+1})\} \tag{3}$$

to measure the redundancy of D_i^{t+1} with respect to D_i^t.

Equation (1) indicates that the length of a sharing fragment set is the sum of the lengths of all the sharing fragments in D_i^t or D_i^{t+1}. Usually, there are more than one sharing fragment set when given D_i^t and D_i^{t+1}. Equation (3) expresses that the redundancy of D_i^{t+1} with respect to D_i^t is the maximal sum of the lengths of all the separated sharing fragments in D_i^{t+1}.

B. The SFS Algorithm

The SFS algorithm is presented to find a good sharing fragment set whose length can represent the redundancy of D_i^{t+1} with reference to D_i^t. Before describing the SFS algorithm, we introduce Definition 4 as follow.

Definition 4. Given D_i^t is a data collected at time t, $S =< d_1, \ldots, d_n >$ where d_i is a sharing fragments in D_i^t .

1) If $d_i \asymp d_{i+1}$ and $f_i < f_{i+1}$, S is a separated sequence. Let
$$L(S) = \sum_{i=1}^{n} len(d_i) \tag{4}$$
be the total length of S and
$$N(S) = n \tag{5}$$
be the total number;
2) If $d_i \uplus d_{i+1}$ and $f_i < f_{i+1}$, S is an overlapped sequence;
3) Assuming $d_1 = D_i^t[f_1, f_1']$ and $d_n = D_i^t[f_n, f_n']$ when S is an overlapped sequence, M is the merger of S if there is $M = D_i^t[f_1, f_n']$;
4) Let $R =< r_1, \ldots, r_m >$ be a separated sequence where r_i is a sharing fragment of D_i^t and S is an overlapped sequence. $\forall i \in \{1, \ldots, m\}$, if there exist $j \in \{1, \ldots, n\}$ meeting that $d_j \odot r_i$, R is a choice of S;
5) If G is a choice of S and $L(G) = max\{L(R) \mid R\ is\ a\ choice\ of\ S\}$, G is a good choice of S;
6) If B is a good choice of S and $N(B) = min\{N(G) \mid G\ is\ a\ good\ choice\ of\ S\}$, B is a best choice of S.

Definition 4 mainly introduces the merger, choice, good choice and best choice of an overlapped sequence. From which, we get Lemma 2 as follow.

Lemma 2. Given a data D_i^t, an overlapped sequence $S =< d_1, \ldots, d_n >$ and its best choice $B =< b_1, \ldots, b_k >$, we have:

1) Each d_j contains at most one b_i;
2) $k \le n$.

Usually, the best choice is not unique, but it always exists. Meanwhile, there is a special situation that an overlapped sequence contains only one sharing fragment. This situation is easy to be disposed. And the good choice is itself, as well as the best choice. For a separated sequence, the good choice and best choice is itself as well.

Based on the definitions above, we propose SFS algorithm which consists of eight steps as given below. The inputs of SFS algorithm are D_i^t and D_i^{t+1} and the output is C' which is a good sharing fragment set.

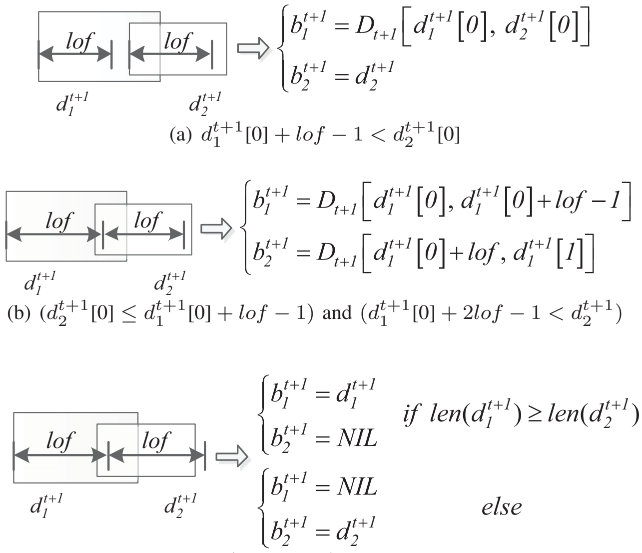
Algorithm 1 SFS algorithm Part 1

Require: D_i^t and D_i^{t+1}.
Ensure: C'.

1: Set $CC = \{\ \}$ and $C' = \{\ \}$. CC is a sharing fragment set whose d_i^{t+1} constructing an overlapped sequence while C' is a sharing fragment set whose d_i^{t+1} constructing a separated sequence.
2: A rolling hash function $H(x)$ is used to slide a window with width of lof along D_i^t to generating $\mid D_i^t \mid +1 - lof$ hash values. The $\mid D_i^t \mid$ is the size of D_i^t and lof is the minimal length of matching we need. The hash values are stored in a hash table HT with chain of linked list to resolve collisions. The nodes in the linked list of the hash table HT store the offset where hash values are computed.
3: Let $k = 0$ where k is the current offset to calculate the hash value in D_i^{t+1}, that is to start with the beginning of D_i^{t+1}.
4: If $k > \mid D_i^{t+1} \mid +1 - lof$ where $\mid D_i^{t+1} \mid$ is the size of D_i^{t+1}, go to step 6. Otherwise, generate the hash value $H(D_i^{t+1}[k, k + lof - 1])$ at offset k.
5: If there is not any value h in HT equal to $H(D_i^{t+1}[k, k + lof - 1])$, let $k = k + 1$, then return to step 4. Otherwise, there is a value h in HT matching $H(D_i^{t+1}[k, k + lof - 1])$. For each offset p corresponding to h in the linked list, we compare $D_i^t[p, p+lof-1]$ with $D_i^{t+1}[k, k+lof-1]$. If they are identical, extend sharing fragment forward in both D_i^t and D_i^{t+1} as far as possible. If the length of matching increases to L, there is $D_i^t[p, p + L - 1] = D_i^{t+1}[k, k + L - 1]$. We may find multiple nodes in the linked list, do this for each one and pick out the longest one of all, that is, $D_i^t[p, p + M - 1] = D_i^{t+1}[k, k + M - 1]$ where M is the maximum length of all. Then we extend the longest matching back as far as possible. If we extend back at length of N, there is $D_i^t[p - N, p + M - 1] = D_i^{t+1}[k - N, k + M - 1]$ as long as $p - N \neq 0$. Add $(D_i^t[p - N, p + M - 1], D_i^{t+1}[k - N, k + M - 1])$ to CC. Let $k = k + M - lof$, go to step 4.

Algorithm 1 describes our SFS algorithm in detail. It mainly refer to two functions. One is the hash function in step 2. Another is function $F = GetChoi(q)$ in step 7.

As for the former, we can use the Karp-Rabin hash function. Karp-Rabin hash algorithm is not the fastest, but it produced a very uniform distribution with fewer collisions. An alternative is the rolling version Rsync is also known as a tool of remote differential compression.



(a) $d_1^{t+1}[0] + lof - 1 < d_2^{t+1}[0]$

(b) $(d_2^{t+1}[0] \leq d_1^{t+1}[0] + lof - 1)$ and $(d_1^{t+1}[0] + 2lof - 1 < d_2^{t+1})$

(c) $d_2^{t+1}[1] \leq d_1^{t+1}[0] + 2lof - 1$

Fig. 3. A best choice when $m = 2$.

Algorithm 2 SFS algorithm Part 2

6: Let $CC = \{(d_1^t, d_1^{t+1}), \ldots, (d_m^t, d_m^{t+1})\}$. Merge all overlapped sharing fragment of $\{(d_1^{t+1}, d_m^{t+1})\}$ into one fragment. For example, if $< d_j^{t+1}, \ldots, d_k^{t+1} >$ is a overlapped sequence, we merge them into one fragment q and add q into C_T. C_T is a set consist of fragments merge from overlapped sequences. Then put the rest of CC into C_1.
7: For each $q \in C_T$, we assume q is merged from $< d_j^{t+1}, \ldots, d_k^{t+1} >$ of $CC = \{(d_1^t, d_1^{t+1}), \ldots, (d_m^t, d_m^{t+1})\}$. By using $F = GetChoi(< d_j^{t+1}, \ldots, d_k^{t+1} >)$, we find a best choice $< b_j^{t+1}, \ldots, b_k^{t+1} >$ of $< d_j^{t+1}, \ldots, d_k^{t+1} >$ where b_p^{t+1} may be NIL without loss of generality. For each $b_n^{t+1} \neq NIL$ as a fragment of d_n^{t+1}, we get its associated fragment y_n as a fragment of d_n^t where $(d_n^t, d_n^{t+1}) \in CC$. Now, for $n \in \{j, \ldots, k\}$, if $b_n^{t+1} \neq NIL$, we add (y_n^t, b_n^{t+1}) into C_2.
8: C_2 plus C_1 is C'.

The latter function $F = GetChoi(q)$ is used to pick out a best choice of $< d_i^{t+1}, \ldots, d_k^{t+1} >$. It is accomplished in two cases: $m = 2$ and $m = 3$. To save the run time, we select a best choice directly by analysis instead of exhaustive method. We use $d_i^{t+1}[0]$ and $d_i^{t+1}[1]$ to store the beginning and end of d_i^{t+1} in D_i^{t+1} when $1 \leq i \leq k$, and $< b_i^{t+1}, \ldots, b_k^{t+1} >$ to represent the best choice where b_i^{t+1} may be NIL without loss of generality. When $m = 2$, a best choice can be picked out easily since $d_i^{t+1}[1]$ and $d_i^{t+1}[2]$ must be overlapped. Fig. 3 illustrates the three cases when $m = 2$.

The case $m = 3$ is more complex than $m = 2$. When $m = 3$, we should consider two exclusive case as Fig. 4 shown. When d_1^{t+1} and d_3^{t+1} are overlapped, it has:

1) If $d_3^{t+1}[1] < d_0^{t+1}[1] + 2lof - 1$, we compare the length of

(a) d_1^{t+1} and d_3^{t+1} are separated (b) d_1^{t+1} and d_3^{t+1} are overlapped

Fig. 4. The case $m = 3$.

d_1^{t+1}, d_2^{t+1} and d_3^{t+1}. If d_1^{t+1} is the longest, b_1^{t+1} is equal to d_1^{t+1} while d_2^{t+1} and d_3^{t+1} are both NIL. Anyway the versa.

2) If $d_0^{t+1}[1]+2lof-1 \leq d_3^{t+1}[1]$, b_2^{t+1} is NIL, meanwhile d_1^{t+1} and d_3^{t+1} construct a overlapped sequence. In this case, we refer to the case $m = 2$.

When d_1^{t+1} and d_3^{t+1} are separated, the method to get a best choice is the same with the case $m = 2$, hence we do not give more detail here. If $m \geq 3$, it is complicated and may be worthy of another paper. In this paper, we assume that an algorithm exists at our disposal and the output from which is same as that from the simple greedy algorithm when $m \geq 3$.

IV. SFS Algorithm Analysis

Let us analyse the result of the SFS algorithm. Lemma 3 following assures that the proposed SFS algorithm can find a good sharing fragment set.

Lemma 3. GivenD_i^t and D_i^{t+1}, we can obtain $C' \neq \{\}$ from SFS algorithm. Based on which, we have

$$Len(C') = L_h(D_i^t, D_i^{t+1}) \tag{6}$$

Proof: By construction of the SFS algorithm, we have $C' \subseteq \Omega(D_i^t, D_i^{t+1})$. Hence $Len(C') \leq L_h(D_i^t, D_i^{t+1})$. $\forall W \in \Omega(D_i^t, D_i^{t+1})$, we need to prove $Len(W) \leq Len(C')$. Let $W' = \{(y_1^t, b_1^{t+1}), \dots, (y_k^t, b_k^{t+1})\}$ and $B' = \{b_1^{t+1}, \dots, b_k^{t+1}\}$, we need to prove $Len(W') \leq Len(C_2)$. From step 6, we have $C_T = \{g_1^{t+1}, \dots, g_m^{t+1}\}$ where each $g^{t+1} \in C_T$ is formed by merging an overlapped sequence S into one fragment of D_i^{t+1}. $\forall b_i^{t+1}$ and b_j^{t+1}, it exists $b_i^{t+1} \asymp b_j^{t+1}$ since $W' \in \Omega(D_i^t, D_i^{t+1})$. $\forall b_i^{t+1} \in B$, we have three cases which exclude each other:

1) $\forall g^{t+1} \in C_T$, we have $b_i^{t+1} \asymp g^{t+1}$;
2) $\exists g^{t+1} \in C_T$ such that $b_i^{t+1} \uplus g^{t+1}$;
3) $\exists g^{t+1} \in C_T$ such that $b_i^{t+1} \odot g^{t+1}$ or $g^{t+1} \odot b_i^{t+1}$.

The 1) can be excluded, otherwise the algorithm should include a $d^{t+1} \in C_T$ which contains b_i^{t+1}, it is a contradiction. With proof by construction, we can also exclude the 2) by the construction of C_T. Hence the 3) is true. If $b_i^{t+1} = g^{t+1}$, $g^{t+1} \in b_i^{t+1}$. When $b_i^{t+1} \neq g^{t+1}$, b_i^{t+1} can not contain g^{t+1}, by the construction of C_T again. Hence, $g^{t+1} \odot b_i^{t+1}$. Therefore, we can conclude that $\forall b_i^{t+1} \in B'$, there is $g^{t+1} \in C_T$ such that $g^{t+1} \odot b_i^{t+1}$. For each $g^{t+1} \in C_T$, let $Q =< b_i^{t+1}, \dots, b_l^{t+1} >$ be the whole subset of B' that g^{t+1} contains and $c =< d_j^{t+1}, \dots, d_k^{t+1} >$ be the overlapped sequence that merges into g^{t+1} at step 6. Then $< b_i^{t+1}, \dots, b_l^{t+1} >$ is a choice of c. If P is the best choice of c, we have $L(Q) \leq L(P)$. For each $g^{t+1} \in C_T$, we denote the best choice as P_i . Hence $Len(W') = len(b_1^{t+1}) + \cdots + len(b_k^{t+1}) \leq L(P_1) + \cdots + L(P_m) = Len(C_2)$. Because all the pairs of sharing fragments in C_1 are separated to each other and separated to the sharing fragments in C_2, $Len(C_1)$ has nothing to do with $Len(W')$. Hence, we have $Len(W) = Len(W') + Len(C_1) \leq Len(C_2) + Len(C_1) = Len(C')$. Now we can conclude $Len(C') = L_h(D_i^t, D_i^{t+1})$. The proof is completed. ■

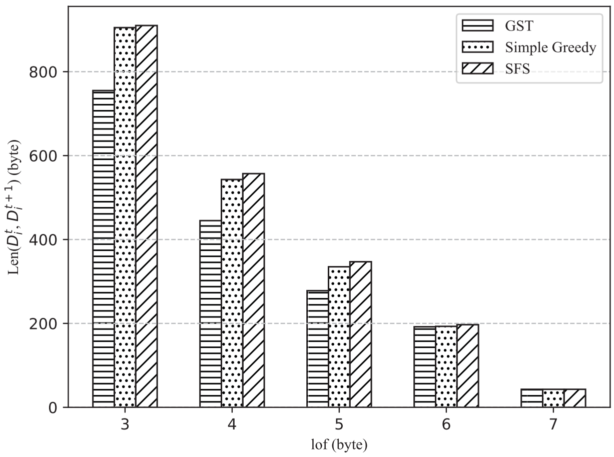


Fig. 5. $Len(D_i^t, D_i^{t+1})$ versus the correlation coefficient with $lof = 4$.

So far, it proves that the sharing fragment set produced by SFS algorithm gets the maximum length. Hence, the result from SFS algorithm can represent the redundancy of D_i^{t+1} in reference to D_i^t.

V. Experiment Results

In this section, we compare our SFS algorithm with GST and simple greedy algorithm in python 2.7.5. The experiment runs on a machine with 3.30GHz Intel Core i5-4590 CPU, 4GB main memory, and a Windows operating system.

We construct the experiment using the synthetic dataset. The alphabet consists of 10 Arabic numbers. The D_i^t is produced randomly while the D_i^{t+1} are produced by two parts where one is copied data fragments from D_i^t and another is added data fragments generated randomly. The D_i^t and D_i^{t+1} are both 1000 bytes in size. Firstly, we investigate the influence of the correlation coefficient on the SFS algorithm. Correlation coefficient is the ratio of the total size of the fragments copied from D_i^t to that of D_i^{t+1}, which means it can be evaluated between 0 to 1. Fig. 5 shows the result of the SFS algorithm when lof is 4 if the correlation coefficient goes from 0.1 to 0.8. The program is ran by 1000 times and the average value is computed. It can be seen that the proposed SFS algorithm performs better than the GST and simple greedy algorithm no matter what the correlation coefficient of two data is. On average, the total length of the sharing fragment set produced by the SFS algorithm is 23.3% larger than that produced by the GST algorithm and 2.4% larger than that by the simple greedy algorithm.

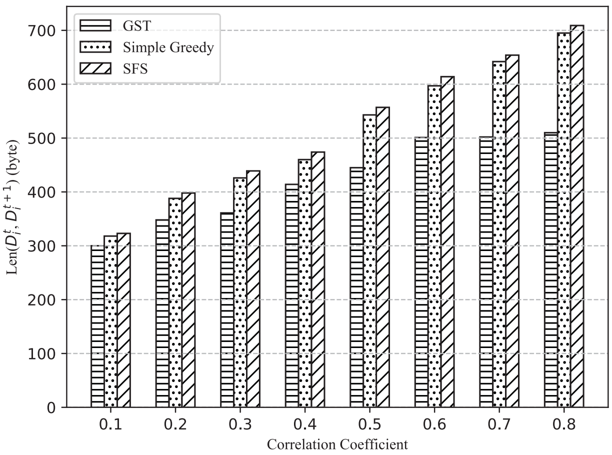


Fig. 6. $Len(D_i^t, D_i^{t+1})$ versus lof when the correlation coefficient is 0.5.

Next we evaluate the detection performance of the proposed SFS algorithm when the lof changes. Let correlation coefficient be equal to 0.5 and lof go from 3 to 7. The program is also ran by 1000 times and the average value is computed. Fig. 6 shows the detection results. Among the 1000 times, SFS algorithm performs well in the case that the lof is 4 and 5. When lof is chosen as 4, the length of the sharing fragment set generated by the proposed SFS algorithm is 25.2% larger than that of the GST and 2.6% larger than that of the simple greedy algorithm. When lof is 5, the length of the sharing fragment set generated by SFS algorithm is 24.8% larger than that of the GST and 2.8% larger than that of the simple greedy algorithm. Since a best choice in case of $m > 3$ is too complicated to get, the result from SFS algorithm is same as the simple greedy algorithm when lof is more than or equal to 7. Combined with the theoretical analysis, we can get that the proposed SFS algorithm can find a better sharing fragment set than the simple greedy algorithm as well as the GST.

VI. Conclusion

In this paper, we have studied the similar data detection between two files, to improve the data transmission efficiency for cooperative spectrum monitoring in space-ground integrated networks. We firstly define a sharing fragment set whose elements are pairs of common fragments of two data. Then a measurement is provided to evaluate whether a sharing fragment set is good enough. Based on which, we proposed the SFS algorithm to find a sharing fragment set which could indicate how much of one data is redundant for another one. Theoretical analysis has revealed that the length of the sharing fragment set produced by SFS algorithm can stand for the redundancy between two data. Furthermore, we have programmed to realize SFS algorithm on test data. Numerical results have been provided to validate the theoretical analysis and shown that the proposed SFS algorithm performs better in finding a good sharing fragment set than the GST algorithm as well as the simple greedy algorithm.

References

[1] J. Liu, Y. Shi, Z. M. Fadlullah and N. Kato, "Space-Air-Ground Integrated Network: A Survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2714-2741, 4th Quart. 2018.

[2] N. Zhang, S. Zhang, P. Yang, O. Alhussein, W. Zhuang and X. S. Shen, "Software Defined Space-Air-Ground Integrated Vehicular Networks: Challenges and Solutions," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 101-109, July 2017.

[3] S. Zhou, G. Wang, S. Zhang, Z. Niu and X. S. Shen, "Bidirectional Mission Offloading for Agile Space-Air-Ground Integrated Networks," *IEEE Wireless Commun.*, vol. 26, no. 2, pp. 38-45, Apr. 2019.

[4] S. Yin, D. Chen, Q. Zhang, M. Liu, and S. Li, "Mining spectrum usage data: a large-scale spectrum measurement study," *IEEE Trans. Mobile Computing*, vol. 11, no. 6, pp. 1033-1046, June. 2012.

[5] Y. Pei, Y. C. Liang, K. C. Teh, and K. H. Li, "How much time is needed for wideband spectrum sensing ?" *IEEE Trans. Wireless Commun.*, vol. 8, no. 11, pp. 5466-5471, Nov. 2009.

[6] M. Sun and K. C. Ho, "An asymptotically efficient estimator for TDOA and FDOA positioning of multiple disjoint sources in the presence of sensor location uncertainties," *IEEE Trans. Signal Process.*, vol. 59, no. 7, pp. 3434-3440, Jul. 2011.

[7] C. Zhang, C. Jiang, J. Jin, S. Wu, L. Kuang, and S. Guo, "Spectrum Sensing and Recognition in Satellite Systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2502-2516, March 2019.

[8] D. Wang, N. Zhang, Z. Li, F. Gao, and X. Shen, "Leveraging High Order Cumulants for Spectrum Sensing and Power Recognition in Cognitive Radio Networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 2, pp. 1298-1310, Feb. 2018.

[9] F. Gao, J. Li, T. Jiang, and W. Chen, "Sensing and recognition when primary user has multiple transmit power levels," *IEEE Trans. Signal Process.*, vol. 63, no. 10, pp. 2704C2717, May 2015.

[10] X. Benavent, A. Garcia-Serrano, R. Granados, J. Benavent, and E. de Ves, "Multimedia information retrieval based on late semantic fusion approaches: Experiments on a wikipedia image collection," *IEEE Trans. Multimedia*, vol. 15, no. 8, pp. 2009-2021, Dec. 2013.

[11] T. Li, and M. Ogihara, "Toward intelligent music information retrieval," *IEEE Trans. Multimedia*, vol. 8, no. 3, pp. 564-574, Sep. 2006.

[12] L. Prechelt, G. Malpohl, and M. Phlippsen, "JPlag: Finding plagiarisms among a set of programs," Technical Report 2000-1, March 28, 2000.

[13] N. Cheng, Z. Yu and K. Wang, "String similarity computing based on position and cosine," in Proc. of the 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC), pp. 256-261, 2017.

[14] M J. Wise, "Running Karp-Rabin Matching and Greedy String Tiling," Technical Report Number 463, March 1993.

[15] M.J. Wise, "String Similarity via Greedy String Tiling and Running Karp-Rabin Matching," Department of Computer Science Technical Report, Sydney University, Sydney. June 1993.

[16] H. Teraoka, F. Nakahara and K. Kurosawa, "Incremental update method for vehicle microcontrollers," in Proc. of the 6th Global Conference on Consumer Electronics (GCCE), pp. 1-2, 2017.

[17] M. Ajtai, R. Burns, R. Fagin, RD.D.E.Long, and L. Stockmeyer, "Compactly Encoding Unstructured Inpouts With Differential Compression," JACM, vol. 49, no. 3, pp. 318-367, 2002.

[18] Tichy and F. Walter "The string-to-string correction problem with block move," ACM Trans. Comput. Syst. vol. 2, no. 4, pp. 309-321, 1984.

[19] C. Reichenberger, "Delta Storage for Arbitrary Non-Text Files," in Proc. of the 3rd International Workshop on Software Configuration Management, pp. 144-152, June. 1991.