Computers and Geosciences 124 (2019) 128–139

Contents lists available at ScienceDirect

Computers and Geosciences

journal homepage: www.elsevier.com/locate/cageo

Research paper

An efficient pixel clustering-based method for mining spatial sequential patterns from serial remote sensing images

Xiaozhu Wu[a,b,*], Ximei Zhang[a]

[a] Key Labortary of Spatial Data Mining & Information Sharing, Ministry of Education, Fuzhou University, Fuzhou 350012, China
[b] College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350012, China

ARTICLE INFO

Keywords:
Spatial sequential pattern
Sequence mining
Serial remote sensing images
Pixels cluster

ABSTRACT

The accumulation of serial remote sensing images provides plentiful data for discovering sequential spatial patterns in various fields such as agricultural monitoring, urban development, and vegetation cover. Otherwise, traditional sequential pattern-mining algorithms cannot be directly or efficiently applied to remote sensing images. In this study, we propose a pixel clustering-based method to improve the efficiency of mining spatial sequential patterns from raster serial remote sensing images (SRSI). Firstly, the images are compressed by using the Run-Length coding schema. Then, pixels with identical sequences are clustered by means of the Run-length code-based spatial overlay operation. Finally, a pruning strategy is proposed, to extend the prefixSpan algorithm to skip unnecessary database scanning when mining from pixel groups. The experimental results indicate that the method presented in this paper could extract spatial sequential patterns from SRSI efficiently. Although accurate support rates for the patterns may not be obtained, our method could ensure that all patterns are extracted with a lower time cost.

1. Introduction

The progress of earth observation technology boasts a continuous accumulation of spatial data. Among such data, owing to their spatial and temporal features, serial remote sensing images (SRSI) provide the potential to keep track of environmental change, urban expansion, and agricultural development, etc. Much research has been carried out, with a focus on mining knowledge from SRSI (Zhang et al., 2016) (Molijn et al., 2016). Spatial sequential patterns are one of the most meaningful types of knowledge that are hidden in SRSI, and they attract great interest among researchers (Helmi and Banaei-Kashani, 2016) (Obulesu and Rama Mohan Reddy, 2016).

Sequential pattern mining (SPM), which is the technical base of spatial sequential pattern mining (SSPM), mainly aims to extract frequent patterns from transaction databases. This is different from its predecessor, SSPM, which tries to discover spatio-temporally frequent sequences from geospatial databases. The complexity of structures, the huge volumes, and the specific features of the geospatial data, such as spatial autocorrelation and spatial relationships among objects, prevent the algorithms of SPM from being effectively applied for mining spatial sequential patterns (Shekhar et al., 2011).

In this paper, we present a pixel-clustering-based method for extracting spatial sequential patterns from SRSI. SRSI can be defined as a set of remote sensing images within different periods aiming for the same area. Some events that are reflected by SRSI occur frequently and periodically, and they are potential sequential patterns to be found. For example, deciduous forests are luxuriant in summer, and will defoliate in winter every year. This phenomenon can be easily captured from SRSI. However, according to the definition of a sequential pattern, only those events with sufficient numbers of occurrences that are higher than a user-defined threshold can be chosen as frequent sequences. In SRSI, these events are related to pixels, and they are indicated by their colors or other attributes. Each pixel represents a certain acreage of land. Therefore, in order to find the spatial sequential pattern, it is necessary to scan all of the pixels in SRSI several times, by using the current sequential pattern mining methods. Otherwise, a typical SRSI may contain billions of pixels, especially high-resolution ones, which will lead to low performance. According to Tobler's First Law of Geography, we can observe that neighbor pixels of SRSI always have similarly changing patterns, as depicted in Fig. 1. An SRSI with three images is shown in Fig. 1, and the colors in the images denote the different crops that are planted. It can be seen that the color of most of the pixels in the area labeled A shift from yellow (corn) in 2010, to green (soy bean) in 2011, and then to yellow in 2012. The other areas have similar phenomena. Intuitively, if we want to find the planting pattern hidden in Fig. 1, we can compute this by using area level

* Corresponding author. Key Labortary of Spatial Data Mining & Information Sharing, Ministry of Education, Fuzhou University, Fuzhou 350012, China.
E-mail addresses: wxz@fzu.edu.cn (X. Wu), n165520013@fzu.edu.cn (X. Zhang).

https://doi.org/10.1016/j.cageo.2019.01.005
Received 22 January 2018; Received in revised form 6 January 2019; Accepted 8 January 2019
Available online 09 January 2019
0098-3004/

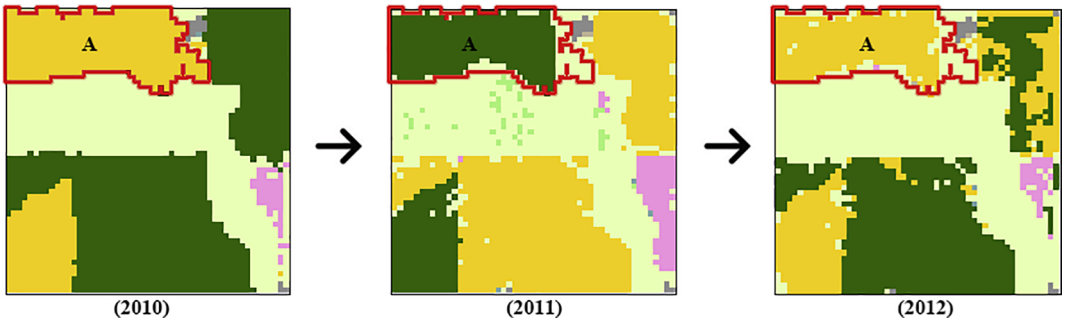

Fig. 1. The pixels in a certain neighborhood tend to share similarly changing patterns.

instead of pixel level. Hence, it is possible to reduce the computation overhead by clustering pixels that share the same changing patterns, and then to perform SSPM on the basis of the clusters.

Based on the idea presented above, in this paper, we extend the current SSPM methods by two aspects:

Firstly, we propose a pixel cluster algorithm to compress the raw dataset of SRSI. For the sake of efficiency, the pixels of SRSI are encoded by run-length coding (RLC). On the basis of RLC representation, to cluster pixels that share the same changing pattern, images of SRSI are intersected with each other at the line level. Afterwards, the SRSI will be combined into one file, which only contains information on the clusters. Finally, the same clusters can be merged, to compress the data volume further.

Secondly, we extend the PrefixSpan algorithm to deal with mining sequential patterns from clusters. The number of pixels in a cluster not only denotes the size of the cluster, but it also represents the number of sequences. We alternate the method of how to calculate the support of a sequence that is defined in PrefixSpan (Han et al., 2001), by considering the size of the cluster. A pruning strategy is also proposed.

The rest of the paper is organized as follows: Section 2 discusses the recent progress of SPM and SSPM. The related concepts and the problem statement are presented in Section 3. Details of the cluster-based mining method and its extension to PrefixSpan are presented in Section 4. Experiments and results are discussed in Section 5. Section 6 draws the conclusion.

2. Related work

The aim of SSPM is to find frequent event sequences from a spatial database that has specific spatial and temporal features. SSPM derives from SPM and extends it from the transaction space to the spatio-temporal space. Research on SPM has produced many useful methods and algorithms since Agrawal and Srikant first introduced the problem of sequential pattern mining in 1995 (Agrawal and Srikant, 1995). Therefore, related SSPM research can be simply classified into two families: spatial and non-spatial (SPM). In actual fact, most problems of SSPM can be converted to SPM by dealing with the spatial properties properly.

As SPM attempts to determine frequent sequential patterns from transaction databases such as shopping databases, similar to the extraction of frequent patterns as in association rule mining, a priori-like methods were firstly proposed. Popular algorithms such as GSP (Srikant and Agrawal, 1996) and SPADE (Zaki, 2001) are examples of these. However, these methods encounter performance bottlenecks when they run through large databases, due to the fact that they need to scan the database many times. Correspondingly, project-based methods presented in (Han et al., 2001) (Ayres et al., 2002) improve the efficiency by shrinking the database on each iteration by projection, which will reduce the number of records that need to be scanned. To meet the needs of actual applications, Pei et al. explored constraint-based SPM, and proposed a set of monotonic and anti-monotonic constraints (Pei et al., 2007) (Pei et al., 2002). These constraints could be used for pruning the searching space, and for greatly improving the efficiency of SPM. Based on these approaches, many studies have been carried out to resolve problems in specialized fields (Wright et al., 2015; Xue et al., 2016; Fan et al., 2016; Cao et al., 2016; Desai and Ganatra, 2015; Hassani et al., 2015; Kemmar et al., 2015). However, the SPM focuses on transactional or text-based data, and ignores the location and inner spatial relationships of events, which are dealt with by SSPM instead.

As for SSPM, there are two typical applications: one is trajectory mining, and the other one is spatial event mining. The SSPM for trajectory mining tries to dig out movement patterns from massive constant trajectory data. For example, most tourists' order of sightseeing is "view spot A → view spot B → view spot C", and then such routes can be extracted by applying SSPM (Tsai and Lai, 2015). More complicated applications of trajectory mining include human mobility in a metropolis, the migration of wild animals, and natural disasters, etc. (Campisano et al., 2016; Geetha and Ramaraj, 2016; Cao et al., 2005; Shaw and Gopalan, 2014). Trajectory data are usually collected from GPS or IOT sensors at any time and frequency, which makes it difficult to precisely relate the trajectory data to a specific location. In such circumstances, the models and algorithms of trajectory mining are quite different from those used in spatial event mining, where the dataset usually contains discrete spatiotemporal events. Tsoukatos et al. presented a Depth-First-Search-like approach to discover long sequential patterns rapidly under different spatial granularities (Tsoukatos and Gunopulos, 2001). Huang et al. proposed a novel sequence index to measure the density of event co-occurrences in a spatiotemporal context, and two corresponding algorithms, named STS-Miner and Slicing-STS-Miner (Huang et al., 2006) (Huang et al., 2008). Julea et al. presented a concept named "group-frequent sequential pattern (GFS-Pattern)" to extract sets of connected pixels sharing a similar temporal evolution for agricultural monitoring from satellite images (Julea et al., 2011) (Julea et al., 2012). Although the idea of GFS-Pattern could be useful for the pruning searching space, it is still too time-consuming, due to it requiring calculation support for sequential patterns on eight-neighbors of pixels.

3. Problem definition

In this section, we formally define the problem of mining spatial sequential patterns from serial remote sensing images. Firstly, the related concepts in the problem context are explained, which include SRSI, image, pixel, spatial sequence, and spatial sequential pattern. Then, the problem definition is given formally.

3.1. Related concepts

Definition 1. SRSI. SRSI refers to a set of remote sensing images that cover the same area at different timestamps, and with uniform resolutions and dimensions. Let SRSI = $\{image_1, image_2...image_k\}$, where $image_i$ is an image at $time_i$. In general, the time gaps between the images are equidistant. According to the application, the measurement units of the time gaps may be the day, month, quarter, or year.

Definition 2. Image. An image is a set of pixels that reflects on the status of the land about a certain theme. The structure and information determine how and what can be mined from the images. Without a loss of generality, the image contains raster data, and then we can let image = $\{line_1, line_2...line_m\}$, where $line_i = \{pixel_{i1}, pixel_{i2}...pixel_{in}\}$. So, each image of SRSI has m rows, n columns, and $m \times n$ pixels. Usually, the image needs a preprocessing procedure, such as image classification, before it can be fed into the SSPM algorithm. In this paper, we assume that the image for SSPM is the product of the raw remote sensing images, rather than a raw remote sensing image itself.

Definition 3. Pixel. The pixel is the elementary unit of the image, and it represents a certain acreage of area, according to the resolution of the image. After remote sensing image preprocessing, such as image classification, each pixel shall contain a value to indicate the state of its corresponding land. The value usually will be represented by the color of the pixel after interpretation. A function event = value(color) can map the color of the pixel to a certain value, which can denote an event that happened to the pixel at $time_i$.

According to the definition presented above, the images in Fig. 1 can be illustrated as follows:

From Fig. 2, we can see that the same pixels in different images may hold different values on different timestamps. This means the event happening on an area of specific land can change overtime. For example, we suppose that the color *yellow* of $pixel_{21}$ denotes corn that was planted in t_1(2010) and t_3(2012), and that the *green* color denotes soy bean that was planted in t_2(2011). These events relating to the same pixel form a spatial sequence.

Definition 4. Spatial Sequence. The spatial sequence is a list of events that happened at different times, and that relate to identical land (pixel). Let $S(pixel_{ij}) = \{(event_1, t_1), (event_2, t_2), ...(event_k, t_k)\}$ denote a spatial sequence of $pixel_{ij}$, where t_i is the timestamp, and $t_1 < t_2... < t_k$. In general, the length of the sequence is less than or equal to the number of images in SRSI. According to Fig. 2, the spatial sequences of $pixel_{21}$ are: {(value(*yellow*), 2010)}, {(value(green), 2011)}, {(value(yellow), 2012)}, {(value(*yellow*), 2010), (value(green), 2011)}, {(value(*yellow*), 2010), (value(yellow), 2012)}, {(value(green), 2011), (value(yellow), 2012)} and {(value(*yellow*), 2010), (value(green), 2011), (value(yellow), 2012)}.

Table 1
Example of table-style serial remote sensing images (SRSI).

Object	Timestamp	Events
$Pixel_{11}$	2010	value(yellow)
$Pixel_{11}$	2011	value(white)
$Pixel_{11}$	2012	value(white)
$Pixel_{12}$	2010	value(yellow)
...	...	...

The number of sequences related to a pixel is $\sum_{i=0}^{k} C_k^i$, where k is the number of images in SRSI (also the longest length of sequences), then the entire quantity of sequences in SRSI is $m \times n \times \sum_{i=0}^{k} C_k^i$, where m and n are the heights and widths of the image, respectively.

Definition 5. Spatial Sequence Pattern. The spatial sequence pattern refers to a spatial sequence that occurs in SRSI frequently enough, which will usually be larger than a user-defined threshold H. The threshold means the minimum occurrence of the sequence, so that the value of H shall be $0 < H \leq (m \times n)$. Let |S| be the number of sequences S in SRSI, and then S is a spatial sequence pattern, if and only if $|S| \geq H$.

3.2. Problem statement

Base on the above definition, the problem of spatial sequential pattern mining from serial remote sensing images can be defined as follows: given a SRSI = $\{image_1, image_2...image_k\}$ and the threshold H, how to find all frequent sequences that $|S| \geq H$.

Obviously, it is easy to convert this problem into transaction-based sequential pattern mining, as we treat pixels as customers (or objects) and the value of the pixels as transaction events. Following this idea, the SRSI can be described as the following example table (see Table 1):

This table is a vertical-style transaction dataset, and it can be simply mined by SPM algorithms such as SPADE and prefixSpan, etc. However, considering the big data volume of SRSI, it would occupy too much time in converting the images to a vertical data table, and executing SPM algorithms on it directly.

4. Methods

With the aim of tackling the problem presented in Section 3, in this section, we propose a pixels-clustering-based sequential pattern mining method. This method attempts to take advantage of the property that adjacent pixels always share the same sequential patterns. This means that it is unnecessary to compute every pixel, as long as we can group pixels to clusters that have common patterns. Following this idea,

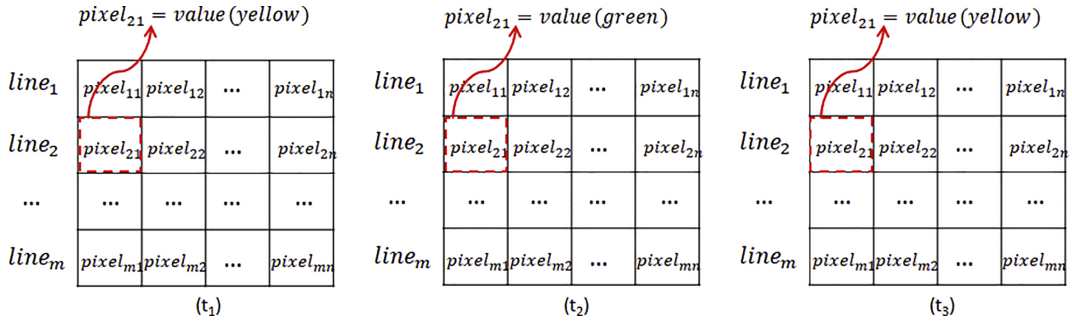

Fig. 2. The pixel view of serial remote sensing images.

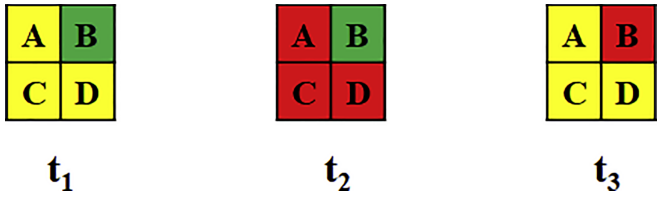

Fig. 3. An example of a *Pixels Group*.

firstly, we introduce the concept of the *Pixels Group*. Secondly, we present a method for clustering pixels on the basis of Run-Length Coding. Finally, the extended prefixSpan algorithm for finding a sequential pattern is described.

4.1. Pixels Group

The *Pixels Group* is a group of neighboring pixels that have identical spatial sequences, and it can also be said that spatial sequences cover this group of pixels. The neighborhood can be defined as the 4-neighbors of pixels. If the neighbor pixels have the same values at each timestamp (or in each image), then they can be grouped into the same *Pixels Group*. For example, considering the four pixels (A, B, C, and D) in Fig. 3, the neighbor pixels A, C, and D have the same changing sequences from (yellow, t_1) → (red, t_2) → (yellow, t_3), so that pixels A, C, and D can be arranged in the same *Pixels Group*. Only pixel B has a distinct sequence that makes it unable to belong to that group.

To find the *Pixels Group*, a simple approach is to test each pixel's neighbors, to find whether they have the same spatial sequences, and then to expand from the neighbor pixels in the *Pixels Group* iteratively until there are no new neighbor pixels to be found. This approach is simple but exhaustive. Here, we present a Run-Length Coding-based method to efficiently find the *Pixels Group*.

4.2. Run-length coding-based pixels clustering method

Run-length coding is a widely-used coding technology in telecommunication and image compression, since it was first introduced in 1967 (Robinson and Cherry, 1967). RLC is a line-based encoding schema that is suitable for compressing SRSI since the *Pixels Group* exists. There are many pixels in a line of SRSI that have same values, as we can see in Fig. 1. After encoding by RLC, each image of SRSI will be converted to a text dataset in the following example schema (see Table 2).

As depicted in Table 2, the image is encoded by line, and each line consists a series of items (sequence–number pairs). The item (value (color), number) depicts that number of successive pixels that have the same value (sequence). This also can be viewed as the *Pixels Group* with 1-length sequence in line. In order to find a *Pixels Group* with k-length, we need to combine all of the RLC data of the images in SRSI. The process of the combination is similar to the spatial overlay analysis, except that the combination is working on a line level. There are two steps in the combination: 1) to intersect the images one-by-one on the line level; 2) to merge the same sequence–number pair items in the RLC image.

Table 2
Example of run-length coding (RLC) data of an image.

Line number	Data of the line
1	(value(yellow),15),(value(white),6) … (value(green),20)
2	(value(yellow),26),(value(green),3) … (value(yellow),8)
…	
m	(value(green),20),(value(white),13) … (value(yellow),12)

To describe the process of intersection between images on the line level, considering the example shown in Fig. 4:

Supposing that the jth line of $image_{i-1}$ and $image_i$ is to be intersected. As depicted in the above figure, $image_{i-1}$ and $image_i$ are 1-sequence images. Let the operator ∧ be the intersection operation. When item intersects with item', where item belongs to $image_{i-1}$ and item' belong to $image_i$, if item.number is equal to item'.number, then they will directly produce a new item, as item∧item′ = (item. value, item′. value, item′. number) ; if item.number is larger than item'.number, then item ∧ item′ = (item. value, item′. value, item′. number), and the rest of item = (item. value, item. number − item′. number) will intersect with next item of $image_i$, and vice versa. For example, $(v_1, 4) \wedge (v_2, 2) = (v_1, v_2, 2)$, the rest of the former item $(v_1, 2)$ needs to intersect with the other items. When the intersection of the two images is finished, a new RLC data containing a longer sequence item will be produced. Then, the new RLC data will intersect with next image of SRSI until all of the images have been combined.

After step 1, the *k* images of SRSI would be combined into one file that contains k-sequence items. However, up until now, we have only clustered the neighbor pixels in the same line into a *Pixel Group*. In step 2, we will merge items with the same sequence in adjacent lines, to make sure that the neighbor pixels in different lines can be clustered into the same *Pixel Group*. However, it can be observed that the pixels that are not in the neighborhood also can have same sequence, even they are far away. So, we can merge these pixels into the same Pixel Group, to further compress the data volume. To fulfill this task, we create a < key, value > structure, namely Map, where the sequence will be the key, and the number will be the value. Then, we can simply scan the output of step 1 only once, from the first item to the last item. Each item will be decomposed into two parts; one is the sequence and the other is the number. The number of the same sequences will be accumulated to the same < key, value > pair.

After step 2, all of the pixels of *k* images would be clustered into the *Pixels Group*. In actual fact, the *Pixels Group* only preserves information about what the sequence is, and how many times it occurs in the SRSI. The accurate position of the pixel, and the pixel itself are longer required. Thus, the data volume of the SRSI is compressed.

Here, we present the algorithm for clustering the pixels into a *Pixels Group* (see Tables 3–6).

4.3. Extracting a spatial sequence base with an extension to prefixSpan

As mentioned in Sections 4.1 and 4.2, pixels and their related events in varying timestamps will be clustered into multiple *Pixels Groups*. The set of *Pixels Groups* can be easily stored in a vertical-style table, which is the most popular data format used in SPM. Table 7 demonstrates an

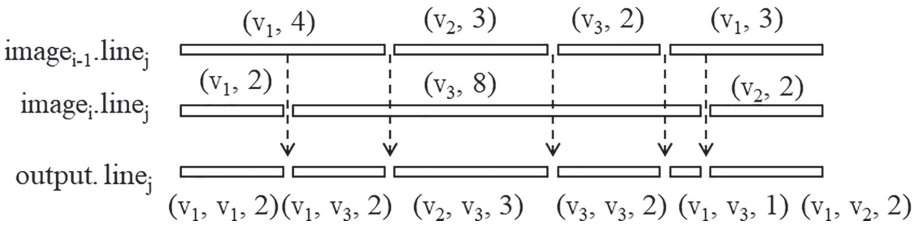

Fig. 4. An example of line intersection.

Table 3
Algorithm for clustering the pixels into a *Pixels Group*

input: SRSI (a set of images)
output: a text file containing *Pixels Group*

```
output=∅ ;
k=|SRSI|;
new RLCVector[k];
for each image_i in SRSI
   RLCVector [i]=convert2RLC(image_i);   //   (1) encoding image to Run-Length code
end for
output=intersectImags(RLCVector);     //   (2) overlay images on line level
mergeItems(&output);   //   (3) merge items with same sequence
return output;
```

The pseudo-codes of functions (1) and (2) are presented in following tables.

Table 4
Pseudo code of function *convert2RLC()*.

input: image
output: RLCVector of image

```
width=getImageProperty(image,"width");   //retreive width of image
height=getImageProperty(image,"height");   //retreive height of image
output=∅ ; // a vector to store RLC code of image
line=∅ ;   // a vector to store RLC code of each line
for i=0 to height-1
   item=∅ ; // to store RLC code of adjacent pixels with same value in a line
   currentValue=getPixelValue(image,i,0);   //reteive value of pixel of image in (x,y)
   item.value=currentValue;
   item.number=0;
   for j=0 to width-1
      if(currentValue = = getPixelValue(image,i,j) )
         item.number++;
         continue;
      else
         line.add(item);
         currentValue=getPixelValue(image,i,j);
         item.value=currentValue;
         item.number=1;
      end if
   end for
   output.add(line);
   line=∅ ;
end for
return output;
```

example.

The table contains *Pixels Groups*. The table rows store items (event, timestamp) of the *Pixels Group*. The items belonging to same *Pixels Group* would be arranged in adjacent rows, and sorted by ascending timestamps. The rows have the same value of the GroupID column, indicating that they are items that belong to the same *Pixels Group*. Compared to Table 1, Table 7 has an additional column, *Size*, which determines how many pixels exit in a *Pixels Group*. Upon the new column, we can make extensions to prefixSpan to accelerate the speed of finding sequence patterns, and we call the extended prefixSpan *group-prefixSpan*.

The Apriori-like algorithms will produce explosive candidate sequences, and scan the database many times, which will lead to low performance. To avoid drawbacks, prefixSpan will not generate a candidate sequence. Instead, it uses projection technology to shrink the database, and to confine the search and the growth of subsequent fragments. Here, we present the related definitions, and briefly describe the procedure of prefixSpan.

Definition 6. Subsequence. Given two sequences: $\alpha = \{e_1, e_2 \ldots e_n\}$ and $\beta = \{e'_1, e'_2 \ldots e'_m\}$ $(n \leq m)$, if there were integers $1 \leq j_1 \leq j_2 \ldots \leq j_n \leq m$ such that $e_1 \subseteq e'_{j_1}$, $e_2 \subseteq e'_{j_2} \ldots e_n \subseteq e'_{j_n}$, then α is a

Table 5
Pseudo-code of function *intersectImags()*.

```
input: RLCVector (a set of RLC code vectors of SRSI)
output: a text file containing Pixels Group on the line level
output=∅ ;
k=|RLCVector|;
currentImage= RLCVector[0];
height=|currentImage.lines|;
for i=1 to k-1
   for j=0 to height-1
      newLine=intersectLines(currentImage.lines[j], RLCVector[i].lines[j]); //intersect two lines
      currentImage.lines[j]=newLine;
   end for
end for
Output=currentImage;
return output;
```

subsequence of β.

Definition 7. Prefix. A sequence $\beta=\{e'_1, e'_2...e'_m\}$ is a prefix of another sequence $\alpha=\{e_1, e_2...e_n\}$ (m≤ n), if and only if (1) $e'_i = e_i$ for (i≤ m− 1), (2) $e'_m \subseteq e_m$, (3) all of the items in $(e_m - e'_m)$ are after those in e'_m. In our case, the condition shall be $e_i = e'_i$ for (i≤ m), because there is only one item in e_i and e'_i.

Definition 8. Projection. Given two sequences: α and β, β is a subsequence of α. A subsequence a' of α is a projection of α w.r.t. β, if and only if (1) β is the prefix of a', (2) a' is the longest subsequence of α that meets the above requirements.

Definition 9. Postfix. Given that sequence $a' = \{e_1, e_2...e_n\}$ is the projection of α w.r.t. prefix $\beta=\{e_1, e_2...e_{m-1}, e'_m\}$ $(m \le n)$, a sequence $\gamma=\{e''_m, e_{m+1}...e_n\}$ is called the postfix of α w.r.t. prefix β, where $e''_m = (e_m - e'_m)$.

The main steps of prefixSpan are as follows:

(1) Scan the sequence database once to find all length-1 sequential patterns.
(2) For each length-1 sequential pattern s_i, use it as a prefix, and construct its projection database by finding projections of sequences w.r.t. s_i.

Table 6
Pseudo-code of function *intersectLines()*.

```
input: line1, line2 (two lines containing items from different images)
output: a new line of items
output=∅ ;
length1=|line1.items|; // retrieve the number of line1's items
length2=|line2.items|;
j=0;
for i=0 to length1-1
   item1=line1.items[i];
   while(j<length2)
      item2=line2.items[j];
      if(item1.number=item2.number)
         output.add(makeNewItem(item1.sequence+item2.sequence,item1.number));
         j++;
         break;
      else if(item1.number<item2.number)
        output.add(makeNewItem(item1.sequence+item2.sequence,item1.number));
        item2.number= item2.number-item1.number;
        break;
      else if(item1.number>item2.number)
        output.add(makeNewItem(item1.sequence+item2.sequence,item2.number));
        item1.number= item1.number-item2.number;
        j++;
      end if
   end while
end for
return output;
```

Table 7
Example of the vertical table of the *Pixels Groups*.

GroupID	Timestamp	Events	Size
1	2010	value(yellow)	940
1	2011	value(white)	940
1	2012	value(white)	940
2	2010	value(yellow)	25
2	2011	value(yellow)	25
…	…	…	

Table 8
The details of the first experimental datasets.

Dataset ID	Region	Number of Pixels	Time Range	Data Volume
D1	A small region of Butler County	2756	2010–2016	0.21 MB
D2	Butler County	1,783,404	2010–2016	11.9 MB
D3	ASD_1910	21,101,514	2010–2016	141 MB
D4	IOWA State	161,936,666	2010–2016	1090.7 MB

(3) For each projection database, repeat steps (1) and (2) recursively until no length-1 sequential patterns can be found. The length-1 sequential patterns that are produced from the projection database would be appended to the corresponding prefix to grow sequential patterns.

We found that the prefixSpan algorithm spends most of its time in scanning and constructing a projection database. It would encounter a performance problem if prefixSpan runs on a large database. Hence, if we can reduce the volume of the database and the times of scanning, the performance can be improved. Here we present two extensions to prefixSpan.

Firstly, we simply revise the counting method of the sequences' occurrence. As we can see from Table 7, the volume of the database has been already compressed, due to pixels that were clustered into *Pixels clusters*. When searching the length-1 sequential patterns, we can use the size of the *Pixels Groups* as an occurrence number to save the searching cost. In other words, if it is without the *Pixels Group*, prefixSpan shall scan at least n rows in order to find a length-1 sequential pattern if it occurs n times in a database. On the contrary, only one row of scanning is needed with the *Pixels Group*. So, compared to running the algorithm on the original database, our method can speed up the procedure of searching for patterns.

Secondly, we use the size of the *Pixels Group* to avoid some unnecessary row scanning. In the original prefixSpan algorithm, to find a length-1 spatial sequence, at least n rows need to be scanned when the threshold $H = n$, and in the worst case, all rows need to be scanned. This is a time-consuming process. However, it can be observed that most of the sequential patterns only exist in some big *Pixels Groups*. If we can find the sequential patterns from a few top big *Pixels Groups*, then those small ones can be ignored, which are a large majority in the database. Following the idea, firstly, we sorted the database in descending order according to size of the *Pixels Group*, and then a pruning strategy was applied:

Let m be the total number of sequences in a database; $n[i]$ is the occurrence number of the ith length-1 sequence up to the present, k is the number of sequences that have been scanned.

During the search for length-1 sequential patterns, we started from the biggest *Pixels Group*, and the search was terminated when there was no length-1 sequence that had been found that could meet one of the termination criteria:

$$n[i] < H \;\&\&\; (n[i] + (m - k)) > H \tag{1}$$

$$n[i] \geq H \;\&\&\; (m - k) > H \tag{2}$$

If there is no length-1 sequence that satisfies the criteria (1), this means that those non-frequent sequences have no opportunity to be frequent, because their occurrence number $n[i]$, plus the rest of the number of sequences $(m - k)$ is less than the threshold; (2) all sequences that have been found are frequent ones, and it is unnecessary to scan the rest of the sequences, due to the number of un-scanned sequences $(m - k)$ being less than the threshold, that is, no new frequent sequences can be found. So, if there are no sequences that satisfy the two conditions, there would be $(m - k)$ sequences that can be skipped safely.

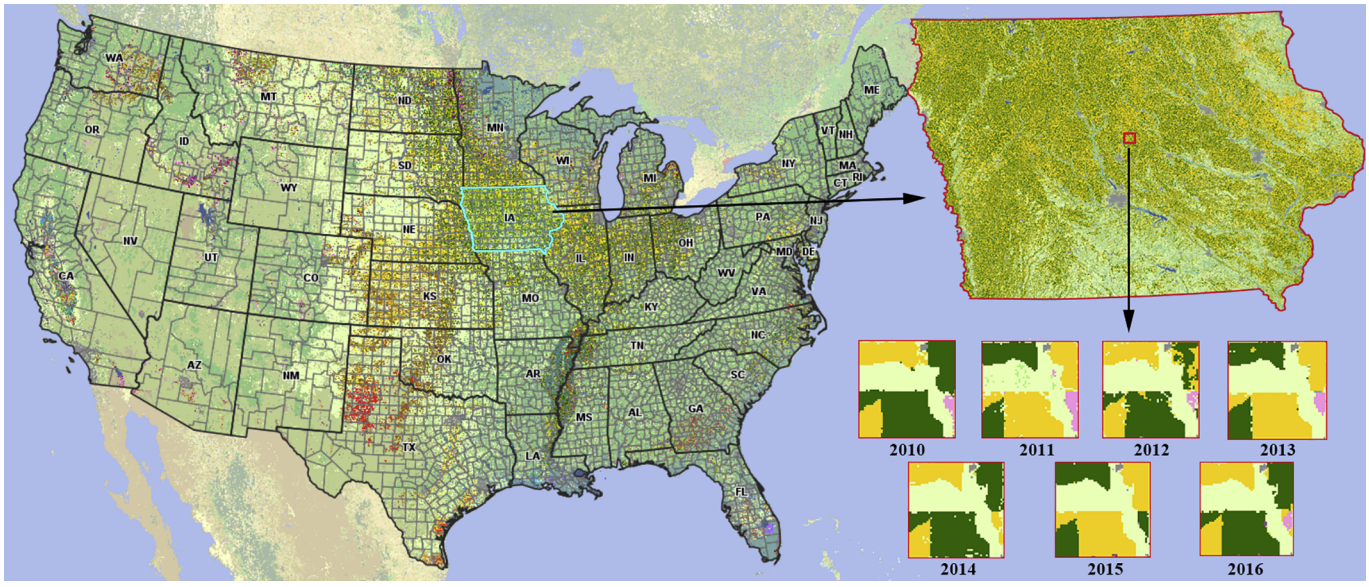

Fig. 5. The first data source for the experiment: Cropland Data Layer (CDL).

Table 9
The details of the second experimental dataset.

Dataset ID	Region	Number of Pixels	Time Range	Data Volume
D5	MCD12Q1.h26v05	5,760,000	2001–2013	214 MB

5. Experiments

To verify the efficiency and effectiveness of our method, in this section, we will report the experiments and the results analysis. Firstly, the information of the dataset for the experiments is described. After that, experiments are carried out, and the results are discussed. All of the experiments are performed on a laptop with an Intel I7-7500 CPU and 8 GB RAM.

5.1. Dataset description

5.1.1. Dataset1: Cropland Data Layer

The dataset used in the experiments was downloaded from the Cropland Data Layer (CDL), which is hosted on CropScape (https://nassgeodata.gmu.edu/CropScape/). The CDL is a raster, geo-referenced, crop-specific land cover data layer that is created annually for the continental United States, using moderate resolution (30 m) satellite imagery and extensive agricultural ground truth (Boryan et al., 2011). Fig. 5 demonstrates a sample of CDL. The color of the map indicates the type of crops that are planted in the particular year.

CropScape allows users to download the CDL dataset at various levels from counties, ASD (Agricultural Statistic District), to states. We retrieved four datasets from CropScape from 2010 to 2016. Each dataset is a typical SRSI that contains seven TIFF images. Table 8 shows the details of these datasets:

The data volume of the four datasets ranges from small, medium, to large, which can be used to test the stability of the method presented in this paper. These datasets are all related to Iowa State, which is one of the largest agricultural states of USA. The acreage of Iowa is about 145,743 km^2, and one pixel of the images represents about 900 m^2. By applying the SSPM algorithm onto these datasets, we expect to find the typical crop planting patterns.

5.1.2. Dataset2: MCD12Q1.051

The second dataset used in the experiments was a subset of the MODIS Land Cover Type product (Short Name: MCD12Q1.051) which provides global land cover types with temporal coverage from 2001 to 2013 (NASA LP DAAC, 2014). MCD12Q1.051 contains five classification schemes, which describe land cover properties derived from observations spanning a year's input of Terra- and Aqua-MODIS data (Friedl et al., 2010). We used the primary land cover scheme, which identifies 17 land cover classes as defined by the International Geosphere Biosphere Program (IGBP).

The subset of MCD12Q1.051 used in this paper was scene number h26v05, which covers several provinces of China, such as Shanxi, Inner Mongolia, and Beijing, etc. The spatial resolution is 500 m, and the image dimensions are 2400 × 2400 pixels. Table 9 shows the details of the dataset:

The time range of dataset D5 is from 2001 to 2013, which means that D5 contains 13 images, and the longest sequence length is 13.

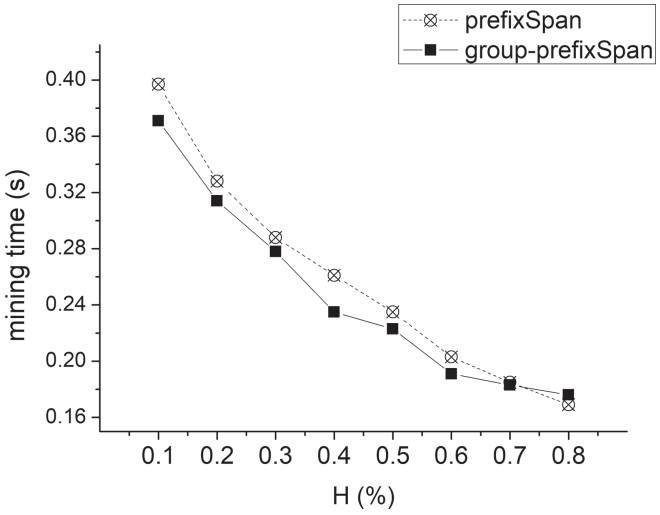

Fig. 6. Evaluation of time consumption on the test datasets: a) D1, b) D2, c) D3, d) D4, e) D5, f) acceleration ratio.

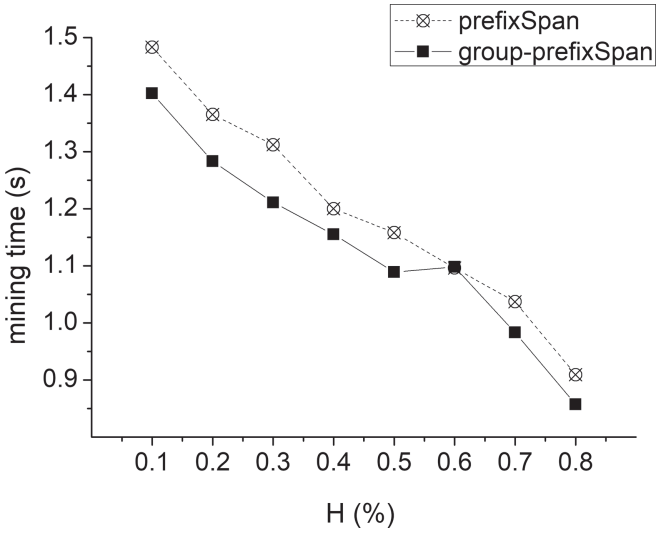

Fig. 6. (*continued*)

Table 10
A comparison of the sequence databases produced by RLC-based pixel clustering and the direct method.

Dataset ID	Produce Directly		RLC-based Pixel Clustering		Compression Ratio
	DV	RowNum	DV	RowNum	
D1	0.14 MB	11,959	0.009 MB	667	94.42%
D2	141 MB	9,645,660	0.97 MB	64,263	99.33%
D3	1.86 GB	121,672,826	4.46 MB	284,121	99.76%
D4	6.91 GB	900,367,862	120 MB	1,307,880	99.85%
D5	1.17 GB	74,880,000	77.3 MB	4,684,810	93.74%

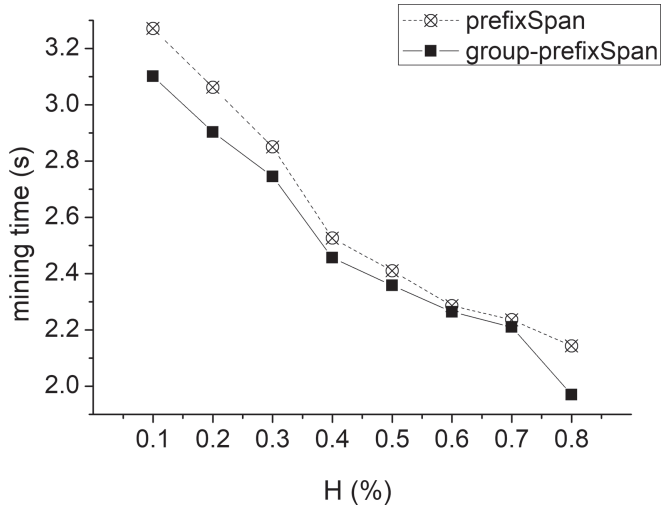

Fig. 6. (*continued*)

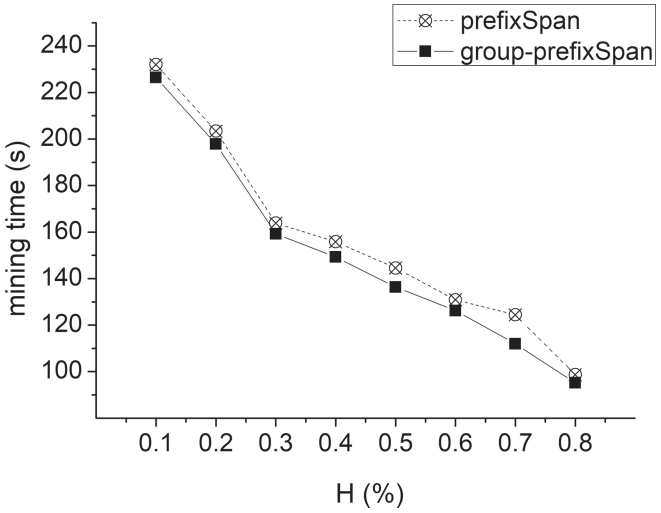

Fig. 6. (*continued*)

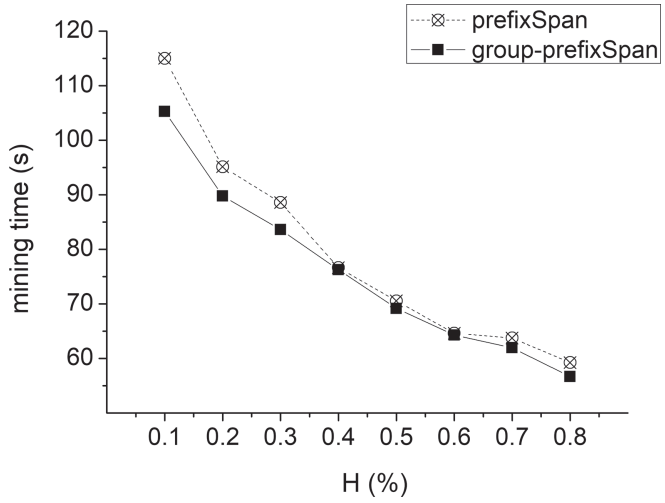

Fig. 6. (*continued*)

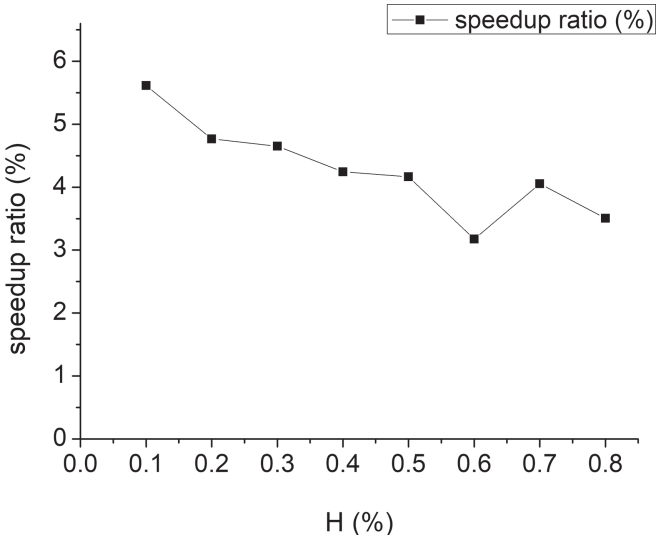

Fig. 6. (*continued*)

5.2. Experiments and results analysis

To demonstrate the effectiveness of the RLC-based pixels clustering method, we produced a sequence database directly from the experimental datasets, and then the volume of the output sequence database was compared. To produce a sequence database directly from the experimental datasets, we read every pixel's value from each image of the dataset, and wrote it to a row of the output sequence database, as in the format of Table 1. The total sequence number of the sequence database equaled to the number of pixels in one image, and the overall number of rows of the sequence database was (*number of pixels*) × (*number of images*). The sequence database produced by the RLC-based pixels clustering method had the same format as Table 7. The comparison is shown in Table 10.

The DV column and the RowNum column, respectively, represent the data volume and the number of rows in the sequence database. Due to each sequence item being stored in one row, the data volume will increase rapidly when the sequence database contains a large number of sequences, such as D3 and D4. As a sharp contrast, the RLC-based pixels clustering method could substantially lower the row number and the data volume. The compression ratio indicates how many rows are merged, by clustering pixels to the *Pixels Group*. Obviously, the shrinking database would greatly reduce the cost of mining sequential patterns from SRSI.

To evaluate the efficiency of group-prefixSpan, we ran the prefixSpan and group-prefixSpan algorithms, respectively, on the five datasets produced by RLC-based pixels clustering. The original prefixSpan algorithm was also extended to handle the *Pixels Group*. In this experiment, the value of H was set in a range of [0.1%, 0.8%], and the increment was 0.1%.

As Fig. 6 (a)-(e) illustrates, the execution time of both of the two algorithms on the five datasets decrease as the grown of threshold H, but the group-prefixSpan algorithm outperformed the prefixSpan algorithm in most situations, although there are some exceptions, in that group-prefixSpan would spend more time in mining patterns, such as D1 (H = 0.8%) and D4 (H = 0.6%). The reason is that the time for examining the termination criteria (1) and (2) was longer than the time saved by skipping the row-checking. For example, when running group-prefixSpan on D4 at H = 0.6%, group-prefixSpan would test the

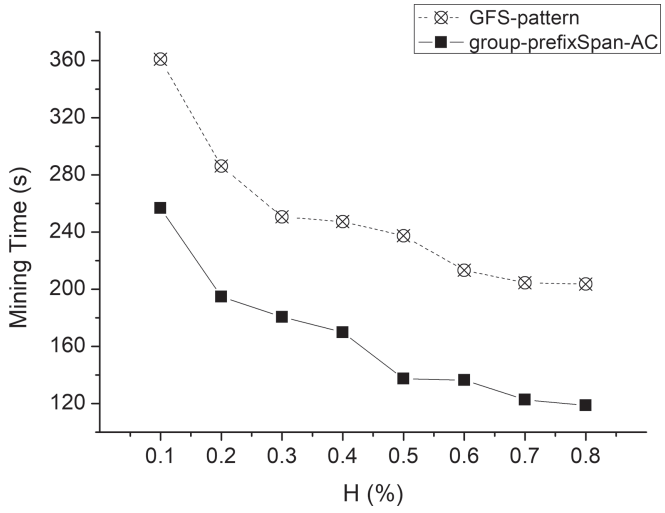

Fig. 7. Comparison of group-prefixSpan with the GFS-pattern algorithm on mining time.

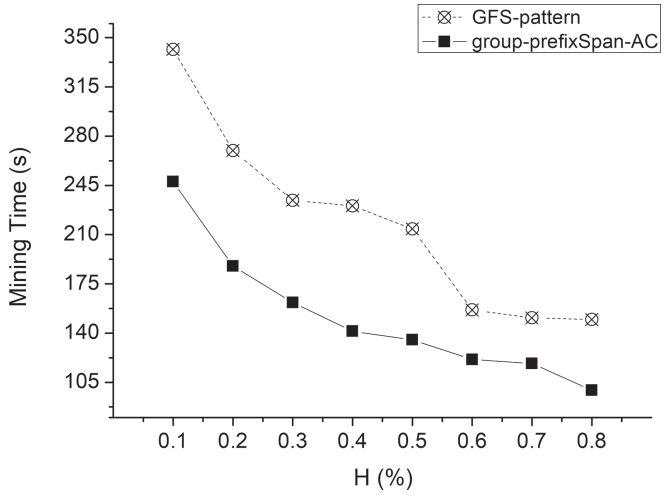

Fig. 7. (*continued*)

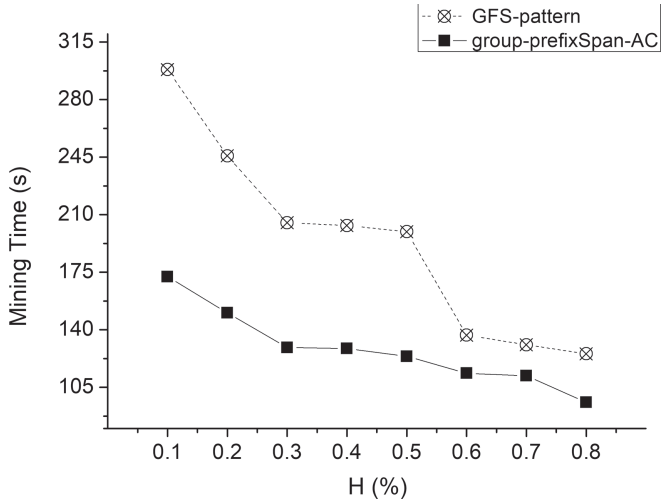

Fig. 7. (*continued*)

terminating criteria 14318 times, while only 12309 rows could be skipped. To analyze the impact of parameter H on the performance of group-prefixSpan, we used the formula $speedup_{ratio} = (time_{prefixSpan} - time_{group-prefixSpan})/time_{prefixSpan}$ to calculate the speedup ratio of group-prefixSpan to prefixSpan. After testing on the five datasets, the average speedup ratio according to threshold H from 0.1% to 0.8% is exhibited in Fig. 6 (f). It was observed that the average speedup ratio slowly decreased with the increase of threshold H. The major reason lies in that a lower H value made it more possible for the group-prefixSpan to check fewer rows to find sequential patterns. However, it can be found that the average speedup ratio declined suddenly at $H = 0.6\%$. Obviously, this was due to the poor performance of group-prefixSpan on D2 at $H = 0.6\%$.

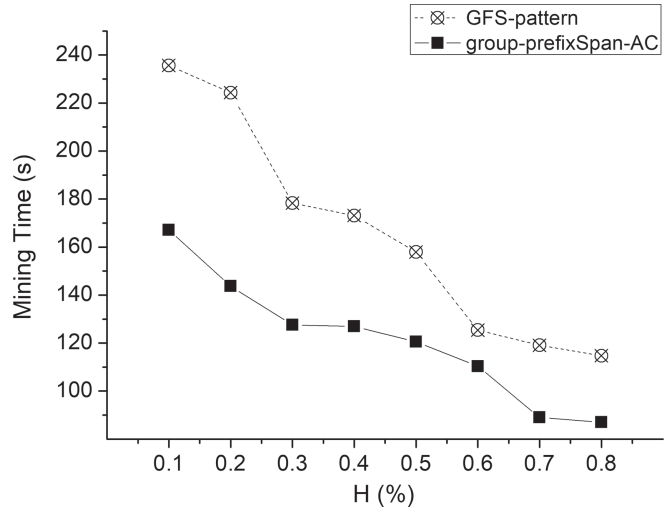

Fig. 7. (*continued*)

Table 11
Sequential patterns extracted from D2 (1 denotes corn and 5 denotes soy bean).

P1: 1 (support = 84.2%)	P9: 1 → 5 → 1 → 1 (support = 61.1%)
P2: 1 → 1 (support = 80.7%)	P10: 5 (support = 77.1%)
P3: 1 → 1 → 1 (support = 78.1%)	P11: 5 → 1 (support = 73.4%)
P4: 1 → 1 → 1 → 5 (support = 60.3%)	P12: 5 → 1 → 1 (support = 67.9%)
P5: 1 → 1 → 5 (support = 69.5%)	P13: 5 → 1 → 5 (support = 63.7%)
P6: 1 → 1 → 5 → 1 (support = 64.9%)	P14: 5 → 5 (support = 65.2%)
P7: 1 → 5 (support = 73.4%)	P15: 5 → 5 → 1 (support = 60.9%)
P8: 1 → 5 → 1 (support = 70.5%)	

Furthermore, we compared group-prefixSpan with the GFS-pattern algorithm that was proposed in (Julea et al., 2011) (Julea et al., 2012). The main idea of the GFS-pattern was to partially push the pixel connectivity constraint into a sequence pattern-extracting process, to prune the searching space. Otherwise, it is a time-consuming operation to check the pattern's average connectivity (Julea et al., 2011). For the sake of fairness, we also integrated the group-prefixSpan algorithm with the pixel connectivity constraint, and named this new algorithm as group-prefixSpan-AC. To perform the comparison, a sub-dataset with 1000 × 1000 pixels was randomly extracted from dataset1. As in (Julea et al., 2012), the average connectivity threshold k was set to be 4, 5, 6, and 7. The comparison result is illustrated in Fig. 7. The experiment shows that group-prefixSpan-AC could reduce the execution time significantly from 12% ($H = 0.6\%$, $k = 7$) to 42% ($H = 0.6\%$, $k = 4$).

What needs to be pointed out is that the group-prefixSpan algorithm may not obtain the accurate support of the sequential patterns, due to it skipping some unnecessary row scanning. For example, there are 15 patterns that would be extracted from D2 at $H = 60\%$ (the details of the patterns are listed in Table 11, and explained in Section 5.3). Fig. 8 shows that the group-prefixSpan algorithm and the prefixSpan algorithm can produce all of these patterns. Otherwise, the support of some

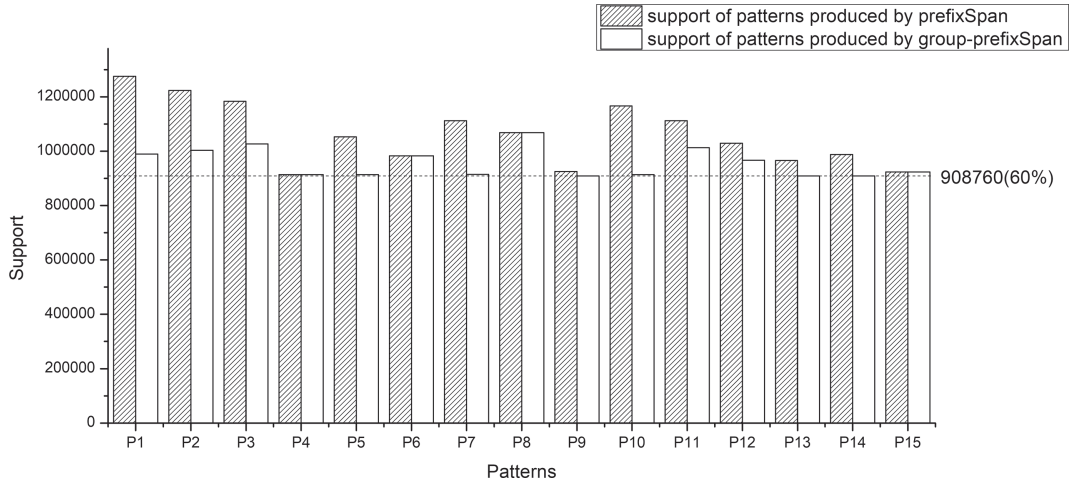

Fig. 8. Comparison of the support of sequential patterns that are produced by prefixSpan with group-prefixSpan.

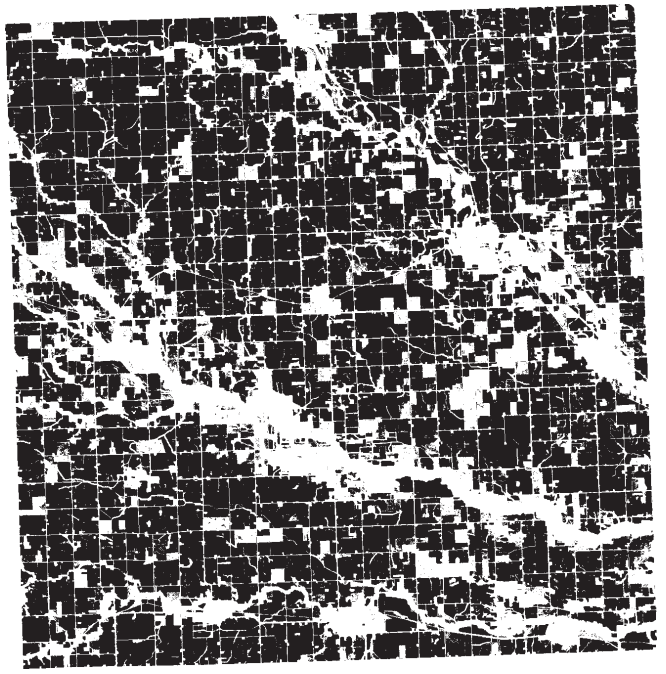

Fig. 9. (a) Map covered by pattern "1" (b) Map covered by pattern "1 → 5 → 1 → 5 → 1 → 5 → 1".

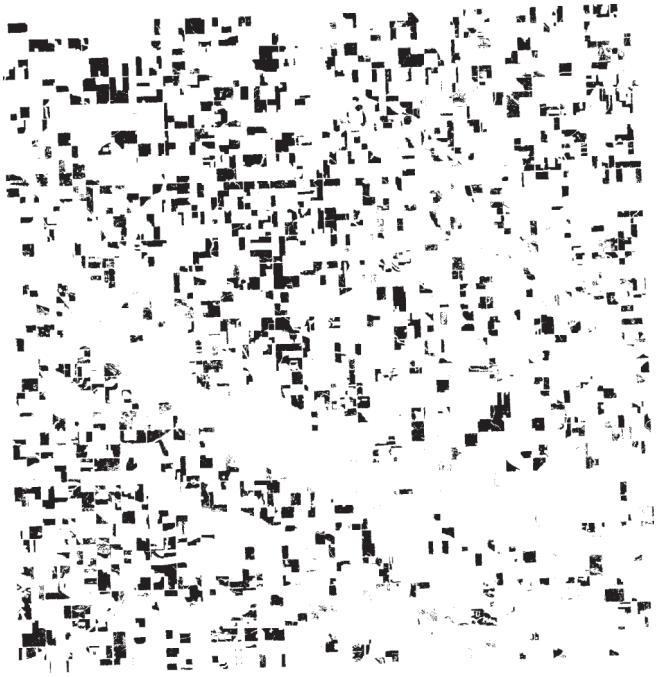

Fig. 9. (*continued*)

patterns produced by group-prefixSpan would be less than that which was produced by prefixSpan. The difference was the price of skipping scanning to accelerate the pattern mining process. Despite that, Fig. 8 also indicates that group-prefixSpan could produce all of the sequential patterns above the threshold, as in prefixSpan.

5.3. Explanation of spatial sequential patterns mining from SRSI

Spatial sequential patterns mining from SRSI aims to reveal the law of development in certain areas. The experiments conducted in this paper for mining on CDL could determine crop planting patterns. For instance, considering the following patterns extracted from D2 (H = 60%):

The pattern P1 showed that from 2010 to 2016, 84.2% (1147.91 km^2) of the cultivated area of Butler County, Iowa State, had corn crops. Pattern P7 shows that 73.4% (1001.05 km^2) of the cultivated area had planted corn after planting soy bean. If we lower the threshold H to 10%, more interesting patterns could be found, such as "1 → 5 → 1 → 5 → 1 → 5 → 1", which reveals that some areas plant corn and soy bean alternately, and that this is useful for crop yield prediction. Obviously, the longer patterns provided more information. Fig. 9 (a) and (b) presents a map of the areas covered by pattern "1" and "1 → 5 → 1 → 5 → 1 → 5 → 1".

6. Conclusions

Mining spatial sequential patterns from big-volume and high-resolution remote sensing image sets is a big challenge. This paper proposed a pixel clustering-based method for spatial sequential pattern mining. The proposed method is highly efficient, as it compresses remote sensing images set by the concept of the *Pixels Group*. To cluster pixels into a *Pixels Group* rapidly, images are converted to the Run-

length coding schema, from which images can be overlaid with each other efficiently, to produce pixels with a sequence list. By sorting a sequence database according to the size of the *Pixels Group*, we show that a portion of records could be ignored while scanning the database. Following this idea, the group-prefixSpan algorithm was proposed, which extends from the prefixSpan algorithm. Experiments showed that even with big data sets, the method can generate all sequential patterns in reasonable times.

Future research will focus on mining local sequential patterns with various local thresholds. Current studies only use a global support threshold, so that only global-frequent sequences can be found. Otherwise, there are some sequential patterns that are frequent enough at local areas, although they may not be frequent globally. These patterns are meaningful to the local area, but they will be ignored, due to only a global support threshold being used. Therefore, it is interesting to conduct future research on mining local sequential patterns.

Conflicts of interest

The authors declare no conflict of interest.

Authorship statement

Xiaozhu Wu: conceived and designed the experiments; performed the experiments; analyzed the data; wrote the paper; revised the paper.

Ximei Zhang: performed the experiments; analyzed the data; revised the paper.

Acknowledgments

The research was supported by the Key Science and Technology Plan Projects of Fujian Province (2015H0015), Education and Technology Plan Projects of Fujian Province (JAT160088), and Foundation of China Scholarship Council (201706655035).

References

Agrawal, R., Srikant, R., 1995. Mining sequential patterns. In: Data Engineering, 1995. Proceedings of the Eleventh International Conference on. IEEE, pp. 3–14.

Ayres, J., Flannick, J., Gehrke, J., et al., 2002. Sequential pattern mining using a bitmap representation. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 429–435.

Boryan, C., Yang, Z., Mueller, R., et al., 2011. Monitoring US agriculture: the US department of agriculture, national agricultural statistics service, cropland data layer program. Geocarto Int. 26 (5), 341–358.

Campisano, R., Porto, F., Pacitti, E., et al., 2016. Spatial Sequential Pattern Mining for Seismic Data. SBBD, pp. 241–246.

Cao, H., Mamoulis, N., Cheung, D.W., 2005. Mining frequent spatio-temporal sequential patterns. In: IEEE International Conference on Data Mining. IEEE Computer Society, pp. 82–89.

Cao, L., Dong, X., Zheng, Z., e-NSP, 2016. Efficient negative sequential pattern mining. Artif. Intell. 235, 156–182.

Desai, N.A.K., Ganatra, A., 2015. Efficient constraint-based Sequential Pattern Mining (SPM) algorithm to understand customers' buying behaviour from time stamp-based sequence dataset. Cogent Eng. 2 (1), 1072292.

Fan, Y., Ye, Y., Chen, L., 2016. Malicious sequential pattern mining for automatic malware detection. Expert Syst. Appl. 52, 16–25.

Friedl, M.A., Sulla-Menashe, D., Tan, B., Schneider, A., Ramankutty, N., Sibley, A., Huang, X., 2010. MODIS Collection 5 global land cover: algorithm refinements and characterization of new datasets. Rem. Sens. Environ. 114, 168–182.

Geetha, P., Ramaraj, E., 2016. An efficient algorithm for frequent trajectory itemset. In: International Conference on Emerging Research in Computing, Information, Communication and Applications. Springer, Singapore, pp. 613–628.

Han, J., Pei, J., Mortazavi-Asl, B., et al., 2001. Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth. In: Proceedings of the 17th International Conference on Data Engineering, pp. 215–224.

Hassani, M., Beecks, C., Töws, D., et al., 2015. Sequential Pattern Mining of Multimodal Streams in the Humanities. BTW, pp. 683–686.

Helmi, Shahab, Banaei-Kashani, Farnoush, 2016. Mining frequent episodes from multivariate spatiotemporal event sequences. In: Proceedings of the 7th ACM SIGSPATIAL International Workshop on GeoStreaming. ACM, pp. 9.

Huang, Y., Zhang, L., Zhang, P., 2006. Finding sequential patterns from a massive number of spatio-temporal events. In: Siam International Conference on Data Mining, April 20-22, 2006. DBLP, Bethesda, Md, Usa.

Huang, Y., Zhang, L., Zhang, P., 2008. A framework for mining sequential patterns from spatio-temporal event data sets. IEEE Trans. Knowl. Data Eng. 20 (4), 433–448.

Julea, A., Meger, N., Bolon, P., et al., 2011. Unsupervised spatiotemporal mining of satellite image time series using grouped frequent sequential patterns. IEEE Trans. Geosci. Rem. Sens. 49 (4), 1417–1430.

Julea, A., Méger, N., Rigotti, C., et al., 2012. Efficient spatiotemporal mining of satellite image time series for agricultural monitoring. Trans. Mach. Learn. Data Min. 5 (1), 23–44.

Kemmar, A., Loudni, S., Lebbah, Y., et al., 2015. PREFIX-PROJECTION global constraint for sequential pattern mining. In: International Conference on Principles and Practice of Constraint Programming. Springer, Cham, pp. 226–243.

Molijn, R.A., Iannini, L., Hanssen, R.F., et al., 2016. Sugarcane growth monitoring through spatial cluster and temporal trend analysis of radar and optical remote sensing images. In: Geoscience and Remote Sensing Symposium. IEEE, pp. 7141–7144.

NASA LP DAAC, 2014. Land cover type yearly L3 global 500 m SIN grid. Version 5.1. NASA EOSDIS land processes DAAC, USGS earth resources observation and science (EROS) center, sioux falls, south Dakota. https://lpdaac.usgs.gov, Accessed date: 1 August 2018.

Obulesu, O., Rama Mohan Reddy, A., 2016. Fast and efficient mining of frequent and maximal periodic patterns in spatiotemporal databases for shifted instances. In: Advanced Computing (IACC), 2016 IEEE 6th International Conference on. IEEE, pp. 35–40.

Pei, J., Han, J., Wang, W., 2002. Constraint-based sequential pattern mining in large databases. In: Proc. 2002 Int. Conf. On Information and Knowledge Management (CIKM'02). ACM, New York, pp. 18–25.

Pei, J., Han, J., Wang, W., 2007. Constraint-based sequential pattern mining: the pattern-growth methods. J. Intell. Inf. Syst. 28 (2), 133–160.

Robinson, A.H., Cherry, C., 1967. Results of a prototype television bandwidth compression scheme. Proc. IEEE. IEEE. 55 (3), 356–364.

Shaw, A.A., Gopalan, N.P., 2014. Finding frequent trajectories by clustering and sequential pattern mining. J. Traffic Transport. Eng. 1 (6), 393–403.

Shekhar, S., Evans, M.R., Kang, J.M., et al., 2011. Identifying patterns in spatial information: a survey of methods. Wiley Interdiscipl. Rev. Data Min. Knowledge Discovery 1 (3), 193–214.

Srikant, R., Agrawal, R., 1996. Mining sequential patterns: generalizations and performance improvements. Adv. Database Technol.—EDBT 96, 1–17.

Tsai, C.Y., Lai, B.H., 2015. A location-item-time sequential pattern mining algorithm for route recommendation. Knowl. Base Syst. 73, 97–110.

Tsoukatos, I., Gunopulos, D., 2001. Efficient mining of spatiotemporal patterns. In: International Symposium on Advances in Spatial and Temporal Databases. Springer-Verlag, pp. 425–442.

Wright, A.P., Wright, A.T., McCoy, A.B., et al., 2015. The use of sequential pattern mining to predict next prescribed medications. J. Biomed. Inf. 53, 73–80.

Xue, F., Shan, Z., Yan, L., et al., 2016. A improved sequential pattern mining algorithm based on PrefixSpan. In: World Automation Congress (WAC), 2016. IEEE, pp. 1–4.

Zaki, M.J., 2001. SPADE: an efficient algorithm for mining frequent sequences. Mach. Learn. 42 (1), 31–60.

Zhang, P., Gong, M., Su, L., et al., 2016. Change detection based on deep feature representation and mapping transformation for multi-spatial-resolution remote sensing images. ISPRS J. Photogrammetry Remote Sens. 116, 24–41.