Interval-Orientation Patterns in Spatio-temporal Databases

Dhaval Patel

National University of Singapore
dhaval@comp.nus.edu.sg

Abstract. In this paper, we present a framework to discover a spatio-temporal relationship patterns. In contrast to previous work in this area, features are modeled as durative rather than instantaneous. Our method takes into account feature's duration to capture the temporal influence of a feature on other features in spatial neighborhood. We have developed an algorithm to discover a temporal-spatial feature interaction patterns, called the *Interval-Orientation Patterns*. Interval-Orientation pattern is a frequent sequence of features with annotation of temporal and directional relationships between every pairs of features. The proposed algorithm employs Hash-based joining technique to improve the efficiency. We also extend our approach to accommodate an incremental mining as updates in real world spatio-temporal databases are common. The incremental algorithm employs an optimization that is based on previously generated patterns to prune the non-promising candidates early. We evaluate our algorithms on synthetic dataset to demonstrate its efficiency and scalability. We also present the patterns identified from real world drought, vegetation and video action databases. We also show that the patterns discovered from video dataset can improve the classification accuracy of activity recognition.

1 Introduction

Pattern mining in spatio-temporal databases aims to discover useful spatio-temporal relationships that are hidden among features. Existing algorithms[14,16,17,18,6,10] have focused on discovering patterns from instantaneous features, that is, features with no duration. This assumption allows the discovered pattern to be simplified to a set of features such as {NormalTemp, HighPrecepitation, NoDrought} or an ordered sequence of features such as {NormalTemp $\rightarrow$ HighPrecepitation $\rightarrow$ NoDrought}. However, features in many real world spatio-temporal databases are durative. For example, spatio-temporal features extracted from video, climate or image dataset such as STIP descriptors[8], SaTScan descriptors[7,9], Cuboid descriptors[4] and etc. have location and duration information. Using location and duration information of features, we can mine useful spatio-temporal relationships.

For example, consider a spatio-temporal pattern, $<\{88 \rightarrow 304\}$,{Overlap}, {North}$>$, shown in Fig 1. This pattern, obtained from KTH video dataset[4], indicates that feature 88 overlaps feature 304 in time and feature 88 is in the nearby "North" region of feature 304. A detailed investigation revealed that this pattern occurs in 6 videos having walking activity, 1 video having running activity and 1 video having

P. García Bringas et al. (Eds.): DEXA 2010, Part I, LNCS 6261, pp. 416–431, 2010.

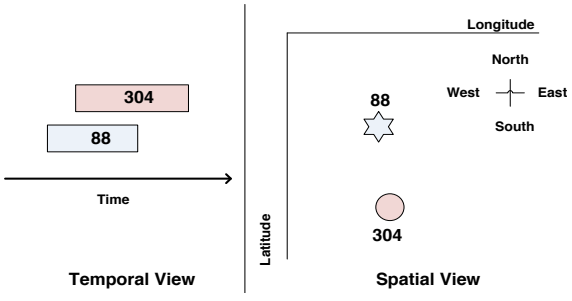


Fig. 1. Spatio-temporal relationship between STIP features in video dataset

jogging activity. In short, this pattern help to discriminate a walking activity with running or jogging activities. Moreover, our experiments reveal that such patterns leads to increase in the classification accuracy of activity recognition. Similarly, we observe that earthquake occurs "during" high atmospheric pressure occurring in the nearby "North" region [1]. This insight is useful in the development of effective earthquake prediction.

In this paper, we define a new class of pattern called Interval-Orientation Pattern, in short IO pattern, to capture the important spatio-temporal relationship among features. IO pattern is an ordering of features having annotation of temporal and directional relationships among all pairs of features in the sequence. As our dataset is non-transactional, we use prevalence index[13] to calculate the support of IO pattern. Moreover, updates in real world databases are common. For example, remote sensing images are captured daily in multitemporal satellite image dataset(http://www.sat.dundee.ac.uk), monthly mean temperature and precipitation are captured at each weather stations in National Climatic Data Center(www.ncdc.noaa.gov). Such update may introduce new patterns or invalid some existing patterns. Recomputing frequent IO patterns from scratch is time consuming process. Thus, we suggest an efficient incremental method to maintain the discovered patterns. The key contributions of this work are summarized as follow:

1. We define a new class of pattern to capture the invariant ordering among features with annotation of temporal and spatial relationship between each pair of features. An efficient algorithm called IOMiner is designed to mine complete set of frequent IO patterns. IOMiner is two-stage algorithm. The first stage adapts disjoint cubes based hashing[15] and efficiently discovers length 2 IO patterns. In the second stage, we use Hash-based join to find the longer length IO patterns.
2. Our incremental approach, called as IncIOMiner(**I**ncremental IOMiner), is constructed under the framework defined by IOMiner. IncIOMiner prunes the search space by estimating the upper bound of prevalence index using previous computations. A Pattern Tree is used to maintain all previous computations.
3. We evaluate our framework on synthetic and real-world dataset. Experiments on synthetic dataset show scalability and efficiency of the proposed approach. We

[1] `http://www.usatoday.com/weather/research/2009-11-12-hquakeweather12_ST_N.htm`

discover IO patterns from drought, vegetation and video dataset. Further, we show that the patterns discovered from video dataset can improve the classification accuracy of activity recognition.

2 Preliminaries

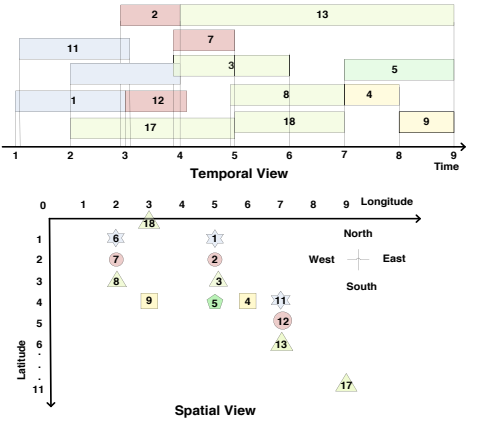
Let F is a set of k features; $F = \{f_1, f_2, \cdots, f_k\}$. Let D be a set of feature instances, where each feature instance is given by a tuple <identifier, feature, {latitude, longitude}, {start_time, end_time}>. We use identifier, d_i, to refer a feature instance and d_i.start_time to refer the start time of d_i. Figure 2(a) lists the features instances used as working example in this paper. We use $D_{0,t}$ to denote a set of feature instances having star_time ≥ 0 and end_time $<$ t. This dataset is updated with a set of feature instances having start_time $\geq$ t and end_time $< (t + \triangle t)$ denoted as $D_{t,t+\triangle t}$.

(a) Original Dataset $D_{0,t}$

<1,f_1,{1,5},{1,3}>, <6,f_1,{1,2},{2,4}> <11,f_1,{4,7},{1.1,3.1}>
<2,f_2,{2,5},{2.9,4}>, <7,f_2,{2,2},{3.9,5}> <12,f_2,{5,7},{3,4.1}>
<3,f_3,{3.1,5},{3.9,6}>, <8,f_3,{3.1,2},{4.9,7}> <13,f_3,{6.1,7},{4,9}>, <17,f_3,{11,9},{2,5}> <18,f_3,{0,3},{5,7}>
<4,f_4,{4.1,6},{7,8}>, <9,f_4,{4.1,3},{8,9}>
<5,f_5,{4.1,5},{7,9}>

(b) Incremental Dataset $D_{t,t+\triangle t}$

<14,f_4,{7.1,8},{10,11}>
<10,f_5,{5.2,2},{10,11}>, <15,f_5,{8.1,7},{12,13}>
<16,f_6,{9.2,6},{13,14}>



(a) Dataset (b) Temporal and Spatial views of $D_{0,t}$

Fig. 2. Working Dataset

Two feature instances d_i and d_j are **neighbor** in **time** domain if (min{d_i.end_time, d_j.end_time} - max{d_i.start_time, d_j.start_time} ≥ 0 or min{|d_i.end_time - d_j.start_time|, |d_i.start_time-d_j.end_time|} $\leq T_\delta$), where T_δ is a time threshold. For example, when T_δ = 1, instances 1 and 2 are neighbor in time but instances 1 and 18 are not.

Two feature instances d_i and d_j are **neighbor** in **space** domain if $Dist(d_i, d_j) \leq R_\delta$, where R_δ is a distance threshold. In this paper, $Dist(d_i, d_j)^2 = (d_i.latitude - d_j.latitude)^2 + (d_i.longitude - d_j.longitude)^2$. For example, when $R_\delta = 2$, instances 1 and 2 are **neighbor** in **space** but instances 1 and 13 are not.

The temporal relationship of feature instance d_i with respect to feature instance d_j, denoted as $TR(d_i, d_j)$, is one of the eight values : **E**qual, **M**eet, **O**verlap, **C**ontain, **B**efore, **S**tart, **S**tart_**B**y, **F**inish_**B**y. For example, $TR(1,2) = O$ and $TR(1,13) = B$.

The directional relationship of feature instance d_i with respect to feature instance d_j, denoted as $DR(d_i, d_j)$, is one of the nine values : **N**orth, **S**outh, **E**ast, **W**est,

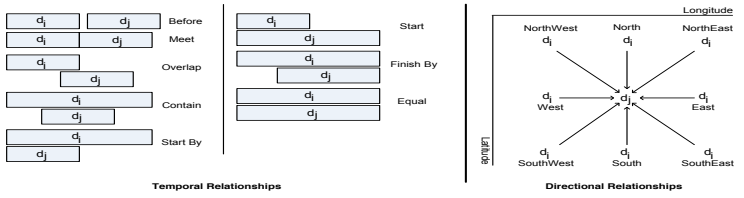


Fig. 3. Temporal and Directional Relationships of instance d_i w.r.t. instance d_j

North**E**ast, **N**orth**W**est, **S**outh**E**ast, **S**outh**W**est and **ND**. **ND** represent a situation when both instances are from same location. For example, $DR(1,2) = N$ and $DR(1,18) = SE$. Fig. 3 shows the temporal and directional information of d_i with respect to d_j.

The temporal relationship between two features f_i and f_j(denoted as $TR(f_i, f_j)$) also takes one of the eight values as described earlier. However, $TR(f_i, f_j)$ is just an abstract specification. Similarly, $DR(f_i, f_j)$ is an abstract directional relationship specification between features f_i and f_j and also takes one of the nine directional values.

We define an **interval-orientation** pattern as a sequence of features with abstract temporal and directional relationship specifications between every pairs of features. It is given as $P = <\{f_1 \rightarrow f_2 \rightarrow ... \rightarrow f_n\}$, $\{TR(f_1,f_2)$, ..., $TR(f_1,f_n)$,$TR(f_2,f_3)$, ..., $TR(f_2,f_n)$,..., $TR(f_{n-1},f_n)$ $\}$, $\{DR(f_1,f_2)$, ..., $DR(f_1,f_n)$,$DR(f_2,f_3)$, ..., $DR(f_2,f_n)$,..., $DR(f_{n-1},f_n)$ $\}>$. The length of IO pattern P, in short $|P|$, is the number of features in the sequence. For example, $P = <\{f_1 \rightarrow f_2\}, \{O\}, \{N\} >$ is a length 2 pattern.

A **pattern instance** of length n IO pattern P is a sequence of n feature instances from D, denoted as $I = \{i_1 \rightarrow i_2 \rightarrow i_3 \rightarrow ... \rightarrow i_n\}$, such that I satisfies following conditions

- feature of j^{th} instance in I is equal to j^{th} feature in P
- start time of j^{th} instance in $I \geq$ start time of all i^{th} instances in I, where i < j
- adjacent instances in I must be neighbor in **time** and **space**
- the temporal and directional relationship between every pairs of instances in I conform to those specified in P
- instances in I are unique

For example, $I = \{1 \rightarrow 2\}$ is a pattern instance of $P = <\{f_1 \rightarrow f_2\}, \{O\}, \{N\} >$ as feature of instance 1(instance 2) is $f_1(f_2)$, the start time of instance 1 is earlier than start time of instance 2, instance 1 and 2 are neighbor in time and space, $TR(1,2) = O$ and $DR(1,2) = N$. Similarly, $\{6 \rightarrow 7\}$ and $\{11 \rightarrow 12\}$ are two other pattern instances of P.

Given an IO pattern P, let $ISet$ is a set of all **pattern instances** of P. We use $ISet_i$ to refer i^{th} pattern instance in $ISet$ and $ISet_{i,j}$ to refer j^{th} feature instance from pattern instance $ISet_i$. Also, $ISet_{*,j}$ is a set of j^{th} feature instances from all pattern instances in $ISet$. The **participation ratio** of j^{th} feature f_j in IO pattern P, denoted as $p\_ratio(f_j,P)$, is defined as:

$$p\_ratio(f_j,P) = \frac{\text{\# unique instances of } f_j \text{ in } ISet_{*,j}}{\text{\# instances of } f_j \text{ in } D} \quad (1)$$

The **prevalence index** of P, denoted as $pi(P)$, is the minimum participation ratio of the features present in pattern P. Formally, $pi(P)$ = minimum$\{p_ratio(f_i,P)$ where $f_i \in P\}$. A pattern P is **frequent** if pi$(P) \geq min_pi$.

For example, consider a pattern $P = <\{f_1 \rightarrow f_2\}, \{O\}, \{N\} >$. $ISet = \{\{1 \rightarrow 2\}, \{6 \rightarrow 7\}, \{11 \rightarrow 12\}\}$ is a set of all pattern instances of P. $ISet_1$ refers to $\{1 \rightarrow 2\}$. $ISet_{1,2} = 2$, $ISet_{*,1} = \{1,6,11\}$ and $ISet_{*,2} = \{2,7,12\}$. $p\_ratio(f_1,P) = \frac{3}{3}$ and $p\_ratio(f_2,P) = \frac{3}{3}$. Hence, $pi(P) = \text{minimum}\{1, 1\} = 1$.

Problem Statement: Given a set of spatio-temporal features $D_{0,t}$, a distance threshold R_δ, a time threshold T_δ and minimum prevalence index threshold *min_pi*, we aim to find all frequent IO patterns efficiently. Further, given an incremental database $D_{t,t+\triangle t}$ to the existing database $D_{0,t}$, find all frequent IO patterns in $D_{0,t+\triangle t} = \{D_{0,t} \cup D_{t,t+\triangle t}\}$, with minimum possible recomputations.

3 Algorithm IOMiner

Algorithm IOMiner is two-stage algorithm; the first stage discovers length 2 IO patterns and second stage uses length 2 IO patterns to extend the length $l(\geq 2)$ pattern into length $l+1$ pattern. Algorithm 1 outline IOMiner. In Line 1, we generate all length 2 IO patterns from given database D. The generated frequent patterns are stored in fre_2_Set and exp_PatSet. Next, each frequent pattern from ext_PatSet is extended into longer length patterns(Lines 3-8). Now, we explain how length 2 patterns are discovered and extended further. The input parameter is $D = D_{0,t}$, $T_\delta = 1$, $R_\delta = 2$ and $min\_pi = 0.50$.

Algorithm 1. IOMiner(D, min_pi, T_δ, R_δ)

Output: $patSet$ = frequent IO patterns
1 fre_2_Set = {**Discover** length 2 IO patterns from D}
2 $patSet = fre\_2\_Set$, $ext\_PatSet = fre\_2\_Set$
3 **while** $ext\_PatSet \neq \phi$ **do**
4 Select pattern P from ext_PatSet and remove P from ext_PatSet
5 ext_Set = {**Extend** P}
6 $ext\_PatSet = ext\_Set \cup ext\_PatSet$
7 Add P to $patSet$
8 **end**
9 return $patSet$

3.1 Discover Length 2 Patterns

The naive method to generate length 2 patterns is candidate set generation and test approach. However, this approach generates total $N^2 * 8 * 8$ length 2 candidate patterns when number of features are N. Clearly, this is not a feasible solution.

Our method first hashes all feature instances from D into disjoint cubes. We use latitude, longitude and start time information of instance for hashing. In particular, space-time dimensions(i.e., {latitude $\times$ longitude $\times$ start time}) are divided into set of disjoint cubes $\{<x_1,y_1,t_1>, <x_2,y_2,t_2>, \ldots, <x_p,y_p,t_p>, \ldots, <x_q,y_q,t_q>\}$. A disjoint window of width $\frac{R_\delta}{2}$ is used to divide each space dimension and disjoint window of width T_δ is used to divide the time dimension. Figure 4(a) represents hashing of instances given in Fig. 2(a)(a).

Instances	Cube
$<1,f_1,\{1,5\},\{1,3\}>$	$<1,5,1>$
$<6,f_1,\{1,2\},\{2,4\}>$	$<1,2,2>$
$<11,f_1,\{4,7\},\{1.1,3.1\}>$	$<4,7,1>$
$<2,f_2,\{2,5\},\{2.9,4\}>$	$<2,5,2>$
$<7,f_2,\{2,2\},\{3.9,5\}>$	$<2,2,3>$
$<12,f_2,\{5,7\},\{3,4.1\}>$	$<5,7,3>$
$<3,f_3,\{3.1,5\},\{3.9,6\}>$	$<3,5,3>$
$<8,f_3,\{3.1,2\},\{4.9,7\}>$	$<3,2,4>$
$<13,f_3,\{6.1,7\},\{4,9\}>$	$<6,7,4>$
$<17,f_3,\{11,9\},\{2,5\}>$	$<11,9,2>$
$<18,f_3,\{0,3\},\{5,7\}>$	$<0,3,5>$
$<4,f_4,\{4.1,6\},\{7,8\}>$	$<4,6,7>$
$<9,f_4,\{4.1,3\},\{8,9\}>$	$<4,3,8>$
$<5,f_5,\{4.1,5\},\{7,9\}>$	$<4,5,7>$

(a) Hashing instances into cube

(a) Processing f_1

$<\{f_1 \rightarrow f_2\},\{O\},\{N\}>$
$\{1 \rightarrow 2\}$
$\{6 \rightarrow 7\}$
$\{11 \rightarrow 12\}$

(b) Processing f_2

$<\{f_2 \rightarrow f_3\},\{O\},\{N\}>$
$\{2 \rightarrow 3\}$
$\{7 \rightarrow 8\}$
$\{12 \rightarrow 13\}$

(c) Processing f_3

$<\{f_3 \rightarrow f_4\},\{B\},\{NW\}>$
$\{3 \rightarrow 4\}$
$\{8 \rightarrow 9\}$

$<\{f_3 \rightarrow f_5\},\{B\},\{N\}>$
$\{3 \rightarrow 5\}$

(d) fre_2_Set : Subset of frequent length 2 patterns with their pattern instances

$<\{f_1 \rightarrow f_2\},\{O\},\{N\}>$
$\{1 \rightarrow 2\}$
$\{6 \rightarrow 7\}$
$\{11 \rightarrow 12\}$

$<\{f_2 \rightarrow f_3\},\{O\},\{N\}>$
$\{2 \rightarrow 3\}$
$\{7 \rightarrow 8\}$
$\{12 \rightarrow 13\}$

(b) Processing features and generated frequent patterns

Fig. 4. Discovering length 2 IO patterns

Given a feature instance $d_i \in D$, we use hashing to discover all other instances d_j from D such that d_i and d_j are neighbor in time and space as well as d_i.start_time $\leq$ d_j.start_time. Let d_i is in cube $< x_m,y_m,t_n >$. All instances satisfying above conditions with d_i must be in one of the cubes $< x_i,y_i,t_s >$ in the set $N_{<x_m,y_m,t_n>} = \{< x_i,y_i,t_s > |(|x_i - x_m| \leq 2) \wedge (|y_i - y_m| \leq 2) \wedge (t_n \leq t_s) \wedge (d_i.end\_time + T_\delta \geq (t_s * T_\delta))\}$. We refer $N_{<x_m,y_m,t_n>}$ as neighbor-set of the cube $< x_m,y_m,t_n >$. For example, consider an feature instance 4. It belongs to cube $< 4,6,7 >$. Hence, other feature satisfying above conditions must be in cubes $< 4,6,7 >$ and $< 4,5,7 >$ w.r.t. our working example. Note that, cube $< 4,7,1 >$ is not selected as start time of instances in $< 4,7,1 >$ are earlier than the start times of instances in cube $< 4,6,7 >$.

Now, we discuss the basic steps to find length 2 IO patterns. We process each feature separately. In our example, we have total five features $F = \{f_1, f_2, f_3, f_4, f_5\}$. Consider a feature $f_1 \in F$. It's instances are 1, 6 and 11 in D. We process each of these instances one by one.

- While processing instance 1, we obtain that it belongs to cube $< 1,5,1 >$ and it's $N_{<1,5,1>} = \{< 1,5,1 >, < 2,5,2 >, < 3,5,3 >\}$. We form pattern instances between instance 1 and instances from cube in $N_{<1,5,1>}$. Here, cubes $< 1,5,1 >$, $< 2,5,2 >$ and $< 3,5,3 >$ contain instances 1, 2 and 3 respectively. Thus, the resultant pattern instances are $\{1 \rightarrow 1\}$, $\{1 \rightarrow 2\}$ and $\{1 \rightarrow 3\}$. we use these pattern instances to generate candidate length 2 patterns. Using $\{1 \rightarrow 2\}$, we generate $P = <\{f_1 \rightarrow f_2\},\{O\},\{N\}>$. We also maintain $\{1 \rightarrow 2\}$ as a pattern instance of P. Next, $\{1 \rightarrow 3\}$ is not valid pattern instance since $Dist(1,3) > R_\delta$. Also, $\{1 \rightarrow 1\}$ is not valid pattern instance as it contains instance 1 more than one time. As a result, we do not generate any candidate pattern for $\{1 \rightarrow 1\}$ and $\{1 \rightarrow 3\}$.
- Next, we process instance 6. Instance 6 belongs to cube $< 1,2,2 >$ and it's $N_{<1,2,2>} = \{< 1,2,2 >, < 2,2,3 >, < 3,2,4 >, < 0,3,5 >\}$. This cubes contain feature instances 6, 7, 8 and 18. Hence, we generate candidate patterns using pattern instances $\{6 \rightarrow 6\}$, $\{6 \rightarrow 7\}$, $\{6 \rightarrow 8\}$ and $\{6 \rightarrow 18\}$. Pattern instance $\{6 \rightarrow 7\}$ generates $<\{f_1 \rightarrow f_2\},\{O\},\{N\}>$. As, this pattern is already generated, we append $\{6 \rightarrow 7\}$ as

it's pattern instance. We do not process $\{6 \rightarrow 6\}$ as explained previously. Remaining pattern instances are not neighbor in space.
- Instance 11 belongs to cube $< 4,7,1 >$ and it's $N_{<4,7,1>} = \{< 4,7,1 >, < 5,7,3 >, < 6,7,4 >, < 2,5,2 >\}$. We follow similar process as explained previously.

Once all instances of feature f_1 are processed, we calculate *pi* of each generated patterns. Those patterns satisfying *min_pi* threshold are stored in fre_2_Set with their pattern instances. In similar manner, we process other features from F. Figure 4(b) lists generated candidate patterns using features f_1, f_2 and f_3.

3.2 Extend IO Pattern

We extend IO pattern $P(\in ext\_PatSet)$ by length 2 pattern $Q(\in fre\_2\_Set)$ if last feature of P is same as the first feature of Q. The extension process generates candidate patterns and it's pattern instances. This method goes as follow: a pattern instance p_i of P is joined with pattern instance q_i of Q, if last feature instance in p_i matches with the first feature instance in q_i. Joining p_i with q_i generates one candidate pattern and it's pattern instance.

For example, consider extension of IO pattern $P = <\{f_1 \rightarrow \underline{f_2}\}\{\text{O}\},\{N\} >$(See Figure 4(b)(d)). This pattern will be extended by $Q = <\{\underline{f_2} \rightarrow f_3\},\{\text{O}\},\{\text{N}\}>$. During extension, a pattern instance $\{1 \rightarrow \underline{2}\}$ from P is joined with pattern instance $\{\underline{2} \rightarrow 3\}$ from Q and generate pattern instance $\{1 \rightarrow 2 \rightarrow 3\}$. Using $\{1 \rightarrow 2 \rightarrow 3\}$, we generate a pattern $<\{f_1 \rightarrow f_2 \rightarrow f_3\},\{\text{O,B,O}\},\{\text{N,N,N}\}>$ with pattern instance $\{1 \rightarrow 2 \rightarrow 3\}$. Similarly, other possible pattern instances are $\{6 \rightarrow 7 \rightarrow 8\}$ and $\{11 \rightarrow 12 \rightarrow 13\}$. Both pattern instances suggest to generate candidate pattern $<\{f_1 \rightarrow f_2 \rightarrow f_3\},\{\text{O,B,O}\},\{\text{N,N,N}\}>$. Once all possible joining are done between pattern instances of P and Q, *pi* is calculated for each generated patterns. For the current case, only one pattern in generated and it is also frequent pattern. We follow similar process if P can be extended by any other pattern from fre_2_Set. All generated frequent patterns are stored in ext_PatSet for further extension.

Note that, when we extend a pattern P by length 2 pattern Q, we join pattern instances of P with pattern instances of Q. In this paper, we use Hash-based joining method to improve the efficiency. In particular, we build a hash table for each pattern Q, $Q \in fre\_2\_Set$ after finishing the first stage. Hash table of pattern Q hashes the pattern instance of Q using it's first feature instance as a key.

4 Algorithm IncIOMiner

We have applied IOMiner to database $D_{0,t}$. Now, $D_{0,t}$ is updated by $D_{t,t+\triangle t}$. The main motivation of IncIOMiner is to improve the time complexity by avoiding unnecessary computations. To achieve this goal, all generated patterns from $D_{0,t}$ are maintained in pattern tree PT(See Figure 5). The pattern with solid rectangle are frequent patterns. The negative border pattern, represented as dotted rectangle, is infrequent pattern but its prefix is frequent. Observe that, each pattern P in PT stores information derived from it's pattern instances. For example, consider a pattern $P = <\{f_1 \rightarrow f_2\},\{\text{O}\},\{\text{N}\}>$

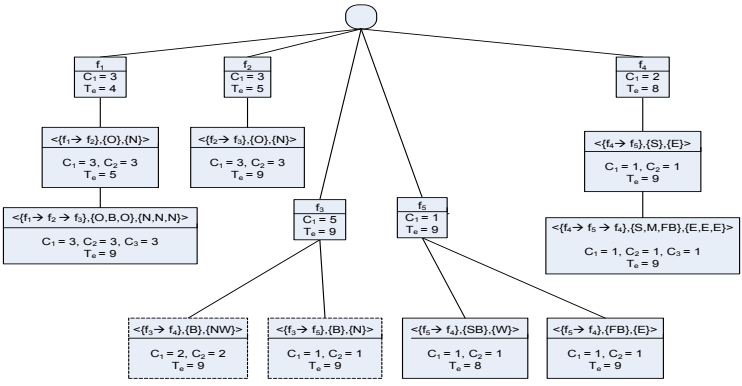


Fig. 5. PT: Pattern Tree

in PT and it's set of pattern instances $ISet$ in Figure 4(b). C_i denotes the number of unique feature instances in $ISet_{*,j}$. Hence, $C_1 = 3$ and $C_2 = 3$. T_e is the end time of feature instance $d_i \in ISet_{*,|P|}$ such that d_i.end_time is greater than end time of any other instances from $ISet_{*,|P|}$. Recall, $|P|$ is the length of P.

Algorithm 2. IncIOMiner($D_{0,t}$, $D_{t,t+\triangle t}$, min_pi, T_δ, R_δ, PT)

Output: $patSet$ = frequent temporal patterns
1 $[fre\_2\_Set, ext\_Set]$ = **GetExtendExtensionPattern**($D_{0,t}$, $D_{t,t+\triangle t}$, PT, min_pi, T_δ, R_δ)
2 $patSet = fre\_2\_Set$
3 **while** $ext\_Set \neq$ **do**
4 Select pattern P from ext_Set and remove P from ext_Set
5 $new\_Set = \{$**Extend** $P\}$
6 $ext\_Set = ext\_Set \cup new\_Set$
7 Add P to $patSet$
8 **end**
9 $canSet = \{$patterns $P \mid P \in PT \wedge$ no new pattern instance of P is generated$\}$
10 $freSet = \{$calculate pi of each pattern in $canSet$ and find frequent patterns$\}$
11 return $\{patSet \cup freSet\}$

Algorithm 2 outline the details. In Line 1, we obtain fre_2_Set and ext_Set. Here, fre_2_Set is set of those frequent length 2 patterns for which new pattern instance is generated after the database update. The pattern from fre_2_Set is used for **extension**. Also, ext_Set is a set of those frequent patterns for which new instance are created or there is a chance that new instance will be created on extension. A pattern P from ext_Set will be extended using patterns from fre_2_Set(Similar to Section 3.2). Also generated patterns are stored in ext_Set for further extension(Lines 3-7). Finally, Lines 9-10 process those patterns from PT for which no new pattern instance is generated after the update. For such pattern, PT contains all information to calculate it's pi value.

4.1 Obtain *fre_2_Set*

Proposition 41 explains how frequent patterns used for extension are obtained.

Proposition 41. *Let D' represents a set of all feature instances from $D_{0,t}$ having end_time $\geq$ t - T_δ. Let E' and E denote the set of all features having instances in D' and $D_{t,t+\triangle t}$ respectively. Let $P = <\{f_1 \rightarrow f_2\}, \{TR(f_1,f_2)\}, \{DR(f_1,f_2)\}>$ is length 2 pattern. When database $D_{0,t}$ is updated to $D_{0,t+\triangle t}$, new pattern instance of P may be generated if one of the following condition is satisfied:*

(i) both features f_1 and f_2 are present in E.
(ii) feature f_1 is present in E' and feature f_2 is present in E.

Proof. Any pair of feature instances from incremental data $D_{t,t+\triangle t}$ generates new pattern instance. As E is a set of features that has instance in $D_{t,t+\triangle t}$, a new pattern instance of $P = <\{f_1 \rightarrow f_2\},\{TR(f_1,f_2)\},\{DR(f_1,f_2)\}>$ may be generated if E contains both f_1 and f_2. Next, a pair of instances where first instance is from $D_{0,t}$ and second instance if from $D_{t,t+\triangle t}$ might generate new pattern instance for P. As instances in $D_{t,t+\triangle t}$ have start time $\geq t$, only those instances from $D_{0,t}$ having end time $\geq$ (t - T_δ) are close in time with any instance from $D_{t,t+\triangle t}$. In other word, a pattern P having first feature from E' and second feature from E might generate new pattern instance. □

Following proposition 41, we use feature instances from $\{D' \cup D_{t,t+\triangle t}\}$ and obtain a set of those length 2 patterns for which new pattern instance is generated after the database update. In our case, $D' = \{4,5,9,13\}$. The generated patterns are stored in *can_2_Set*(See second column in Table 1). Our next step is to obtain frequent patterns from *can_2_Set*. In short, we have to obtain pattern instances of each pattern in *can_2_Set* using complete data $D_{0,t+\triangle t}$. As we have already generated pattern instances using $\{D' \cup D_{t,t+\triangle t}\}$, we generate pattern instances of each pattern in *can_2_Set* using $D_{0,t}$.(i.e., original dataset). However, not all patterns in *can_2_Set* require to regenerate it's pattern instances from $D_{0,t}$. Now, we present a technique for estimating upper bound of $pi(P)$, $P \in$ *can_2_Set*.

Table 1. Analysis of length 2 patterns obtained using proposition 41

Sr. No.	P $\in$ *can_2_Set*	Instances using proposition 41	Info. of P in PT $\{C_1,C_2\}$	estimated p_ratio(f_i,P)	$UB\_pi(P)$
1	$<\{f_4 \rightarrow f_5\}$,{B},{NE}>	{9 →10}, {14 → 15}	-	$\{\frac{2}{3},\frac{2}{3}\}$	0.50
2	$<\{f_5 \rightarrow f_6\}$,{M},{NE}>	{15 → 16}	-	$\{\frac{1}{3},\frac{1}{1}\}$	0.33
3	$<\{f_3 \rightarrow f_4\}$,{B},{NW}>	{13 → 14}	{2,2}	$\{\frac{3}{5},\frac{3}{3}\}$	0.60
4	$<\{f_3 \rightarrow f_5\}$,{B},{N} >	{13 → 15}	{1,1}	$\{\frac{2}{5},\frac{2}{3}\}$	0.40

Given a length 2 pattern P from *can_2_Set* and it's pattern instances set I generated using $\{D' \cup D_{t,t+\triangle t}\}$, one of the following two cases is possible:

Case : $P \notin PT$**.** There is no pattern instance of P in $D_{0,t}$. In short, we already have true *p_ratio* of each feature in P. For this case, we put P in *fre_2_Set* if $pi(P) \geq min\_pi$. We also record information about P in PT and remove it from *can_2_Set*.

Case : $P \in PT$**.** We estimate upper bound of *p_ratio* for each feature in P(See Equation 2) and then finally obtain $pi(P)$(See Equation 3). If $UB\_pi(P) < min\_pi$, we update C_i and T_e of P in PT and remove P from *can_2_Set*. However, if $UB\_pi(P) \geq min\_pi$, we have to obtain it's pattern instances from $D_{0,t}$.

$$p\_ratio(f_i, P) = \frac{C_i + \{\text{\# of unique instance in } I_{*,i}\}}{\text{\# of instance of } e_i \text{ in } D_{0,t+\triangle t}} \tag{2}$$

$$UB\_pi(P) = minimum\{p\_ratio(f_i, P)\} \tag{3}$$

Table 1 present an analysis of proposed technique on our working example. Note that, first two patterns satisfy first case (they are not generated in previous round) and last two patterns satisfy second case. Clearly, pattern 1 qualify *min_pi* threshold and is kept in fre_2_Set. pattern 2 and 4 do not qualify *min_pi* threshold and thus removed from *new_2_Set*. We generate valid instances for the remaining patterns in *new_2_Set* by scanning database $D_{0,t}$ and those patterns satisfy *min_pi* threshold requirement are later transferred to fre_2_Set.

4.2 Obtain *ext_Set*

Proposition 42 explain how we obtain IO patterns which will be **extended** further.

Proposition 42. *Let frequent pattern R is generated by extending pattern P with length 2 pattern Q. A new pattern instance of pattern R is generated after the database update if at least one of the following condition is satisfied:*

(i) The value of $P.T_e$ *in PT* $\geq$ *(t-*T_δ*)*
(ii) at least one pattern instance is generated for pattern P after the database update

Proof. Let appending $D_{t,t+\triangle t}$ to $D_{0,t}$ generate at least one pattern instance v for pattern R. Careful observation suggest that, at least one feature instance in v must be from incremental part $D_{t,t+\triangle t}$. Otherwise, v must be generated in previous iteration.

If only last feature instance in v is from $D_{t,t+\triangle t}$, then immediate prefix of R (i.e., P) must be from previous database $D_{0,t}$. However, The value of $P.T_e$ must be $\geq$ (*t-*T_δ) to qualify T_δ. If more than one feature instances of v are from updated dataset $D_{t,t+\triangle t}$ then immediate prefix of R(i.e., P) has at least one pattern instance generated after the database update. □

To obtain the patterns satisfying condition 1, we traverse PT and obtain those patterns having $(t - T_e) \leq t_\delta$ and put them in *ext_Set*. Further, fre_2_Set contains prefix of those patterns which satisfy the second condition. Finally, *ext_Set* = *ext_Set* $\cup$ fre_2_Set. Table 2 presents fre_2_Set and *ext_Set* obtained for our working example.

Algorithm 3 summarize the details. We generate length 2 patterns using Proposition 41(Line 2). Next, we refine *can_2_Set* to remove non frequent candidate without generating their pattern instances. In Line 4, we traverse PT and obtain a set of patterns satisfying condition 1 discussed in Proposition 42. Note that, we only consider those patterns satisfying *min_pi* threshold while traversal. In Line 5, we generate pattern instances of pattern present in *can_2_Set* and *ext_Set*. To derive pattern instances of given

Table 2. fre_2_Set and ext_Set

fre_2_Set	ext_Set
$<\{f_4 \rightarrow f_5\},\{B\},\{NE\}>$	$<\{f_4 \rightarrow f_5\},\{B\},\{NE\}>$
$<\{f_3 \rightarrow f_4\},\{B\},\{NW\}>$	$<\{f_3 \rightarrow f_4\},\{B\},\{NW\}>$
	$<\{f_1 \rightarrow f_2 \rightarrow f_3\},\{O,B,O\},\{N,N,N\}>$
	$<\{f_2 \rightarrow f_3\},\{O\},\{N\}>$

pattern P, use similar process explained in section 3. For example, consider a pattern P = $<\{f_1 \rightarrow f_2 \rightarrow f_3\},\{O,B,O\},\{N,N,N\}>$. We first derive pattern instances of $<\{f_1 \rightarrow \underline{f_2}\}\{O\},\{N\}>$ and $<\{\underline{f_2} \rightarrow f_3\},\{O\},\{N\}>$ and then join them to derive it's pattern instances. Finally, fre_2_Set and ext_Set is returned. Note that, ext_Set does not contain $P = <\{f_4 \rightarrow f_5\},\{S\},\{E\}>$ and it's extended patterns as P is not frequent. We omit proof of completeness of IncIOMiner for brevity.

Algorithm 3. GetExtendExtensionSet($D_{0,t}$, $D_{t,t+\triangle t}$, PT, min_pi, T_δ, R_δ)

Output: fre_2_Set, ext_Set
1 Obtain D' from $D_{0,t}$
2 can_2_Set = {length 2 patterns using $\{D' \cup D_{t,t+\triangle t}\}$}
3 can_2_Set = Refine can_2_Set using upperbound estimation (Eq. 2 and 3)
4 ext_Set = $\{P \mid (P \in PT) \wedge (|P| > 1) \wedge (P.T_e \geq (\text{t-T}_\delta)) \wedge (\text{UB\_pi}(P) \geq min\_pi)\}$
5 Generate pattern instances of patterns from (ext_Set and can_2_Set) using $D_{0,t+\triangle t}$
6 $fre\_2\_Set = fre\_2\_Set \cup$ {frequent patterns from can_2_Set}
7 $fre\_ext\_Set = fre\_2\_Set \cup$ {frequent patterns from ext_Set}
8 return $\{fre\_2\_Set, fre\_ext\_Set\}$

5 Experimental Results

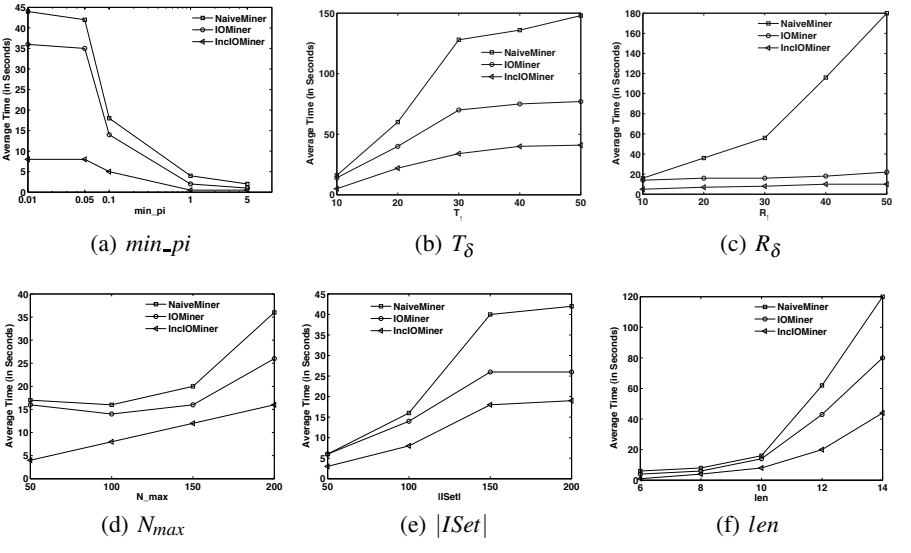
We conduct detailed performance study. All algorithms are implemented in C and compiled with -O2 option. We use a Pentium 4 machine with 3GB RAM running windows. No other user processes were running at that time. We also implement an algorithm called *NaiveMiner* that use only cube based hashing to obtain the longer length pattern. Actually, NaiveMiner is motivated from [19,12].

5.1 Efficiency Experiments

We first implement a synthetic data generator to obtain spatio-temporal features. Table 3 summarizes the experimental parameters used to generate the dataset. Essentially, we generate features so that the generated data has total N_{max} maximal patterns of average length len and each pattern has $|ISet|$ number of average pattern instances. While generating features, we also use parameters R_δ, T_δ and min_pi. With using generator and user defined input parameters, we create an initial dataset D_0 and it's four increments. The size of increment (i.e., inc_size) is 10% of original data size.

Table 3. Experimental Parameter

Parameter	Definition	Default value
N_{max}	Number of maximal IO patterns	100
len	Average length of maximal IO patterns	10
$\|ISet\|$	Average number of pattern instances for each IO patterns	100
dur	Average duration of features	10
N	Number of features type	100
R_δ	Distance threshold for spatial proximity	10
T_δ	Distance threshold for temporal proximity	10
min_pi	Prevalence index threshold	0.1
inc_size	Size of incremental database	10%
map	Latitude-Longitude extent of the map	1000 x 1000



(a) *min_pi* (b) T_δ (c) R_δ

(d) N_{max} (e) $|ISet|$ (f) *len*

Fig. 6. Effects of varying *min_pi*, T_δ and R_δ, *N_Max*, $|ISet|$ and *len*

In our first set of experiments, we use default value of parameters given in Table 3 to generate spatio-temporal features and vary user defined parameters especially, *min_pi*, R_δ or T_δ. Note that, the data generated using default parameters has around 70K number of instances in the original data and subsequent increment has around 5K, 7K, 8K and 7K instances. In our experiment, we capture runtime of NaiveMiner, IOMiner and IncIOMiner for mining each datasets(Original dataset and it's 4 increment) and report the average running time[11]. We first vary *min_pi* from 0.01% to 5%. Figure 6(a) reports the obtained result. As we reduce *min_pi*, running time increased. However, IncIOMiner runs very fast as *min_pi* reduce. Similarly, we vary T_δ from 10 to 50(See Figure 6(b)) and R_δ from 10 to 50 (See Figure 6(c)). For both cases, run time of NaiveMiner and IOMiner increase rapidly. The reason is, increasing R_δ or T_δ generates many patterns.

In next set of experiments, we vary N_{max}, len or $|ISet|$. Note that, increasing any of these parameters indirectly increases the size of database and also required to increase the number of features(i.e., N) while generating dataset. More specifically, these set of

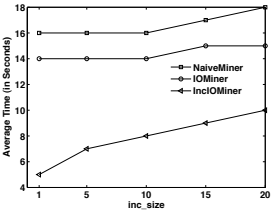


Fig. 7. Effects of varying size of increment

experiments also capture effects of increasing size of database and number of features on average running time. We vary N_{max} from 10 to 200 (size of original database linearly increases from 7K to 142K). Figure 6(d) present the results. Clearly, as size of data increase, the performance gap between IOMiner and IncIOMiner also increase. Similar apply when we vary $|ISet|$ from 10 to 200(See Figure 6(e)). Similarly, we vary average length of maximal patterns from 6 to 14 and observed that algorithm IncIOMiner is efficient(See Figure 6(f)). Note that, we use default value of increment size(10%) for these set of experiments.

We vary *inc_size* from 1% to 20% (See Figure 7). Note that, as we increase the size of increment, total database size also increase. It is obvious that keeping *min_pi* constant and increasing total database size will generate less number of patters. We can observe that when size of increments is less, IncIOMiner is very efficient algorithm. Also, NaiveMiner is outperformed by IOMiner in all the experiments.

5.2 Effectiveness Experiments

We conduct qualitative experimental study on three real-world public datasets named drought, vegetation and human activity videos.

Drought Dataset. This dataset, obtained from National Climatic Data Center, includes monthly average of three variables named temperature(AvgTemp), precipitation (AvgPcp) and palmer drought severity index(PDSI). Climate divisions at various spatial locations record value of above variables. Current dataset has total 334 climatic divisions and we use 10 year data captured from 1999 to 2008. In short, we have real valued time series of each variable at each climate division. We use discretization process[2] to extract spatio-temporal features such as episode of high or low temperature(precipitation), episode of cold or wet drought at particular location. Finally, our working dataset has around 85K features. From such dataset, we discover key spatio-temporal relationships to link the drought condition to high/low events. We run IOMiner with $R_\delta = 30$, $T_\delta = 5$ and $min\_pi = 5\%$. It has been noted that high temperature and low precipitation creates many problems and one of the problem is period of dry spell(i.e., Low PDSI)(http://www.sciencedaily.com/releases/2007/08/070820144517.htm)(See Pattern P3 in Table 4).

Vegetation Dataset. This dataset, obtained from ISLSCP II Data Archive, includes seven variables named precipitation(Pcp), temperature(Temp), cloud cover(CloudCover), diurnal temperature range(DiurnalTemp), wet dry frequency(WDFre), vapor pressure

(VPressure) and normalized difference vegetation index(NDVI). This dataset has global mean monthly value for each variable from 1991-1995. We use discretization process[2] to extract spatio-temporal features where a feature represent an episode of high or low value for each attribute at certain location. Finally, our working dataset has around 45K features. From this dataset, we discover key spatio-temporal relationships that links high/low vegetation condition to high/low episode of any variables. We run IOMiner with $R_\delta = 20$, $T_\delta = 3$ months and $min\_pi = 5\%$.

Human Activity Video Dataset. We use two different datasets named KTH dataset (http://www.nada.kth.se/cvap/actions/) and ICPR 2010 Challenge Dataset. KTH dataset has 25 persons engaged in the following activities: running, walking, jogging, boxing, clapping and waving. We use subset of dataset that includes 10 persons where each person repeats each activity 4 times for about 4 seconds each, wearing different clothing. Thus we have total 240 video clips. First, we extract spatio temporal word, a kind of STIP descriptor, described in [4] from each video clip. Finally, our working dataset has around 60K features. From this dataset, we discover key spatio-temporal relationships that help to discriminate human activities. Patterns P8-P10 in Table 4 are the patterns discovered from KTH dataset.

To further validate the usefulness of IO patterns, we extend a bag of spatio-temporal words classification technique[4] for human action recognition. Our approach extends the bag of words dataset with discovered IO patterns as suggested in [1]. Next, we perform 10 fold cross validation using SVM classifier on initial bag of words dataset, denoted as Bag_of_Words, and extended bag of word dataset, denoted as Bag_of_Word+IO. Instead of IO patterns, we also consider other patterns such as temporal pattern(TP) and interval Pattern(IP). Temporal pattern does not consider the space dimension[12] and interval pattern is an IO pattern without the direction relationship. We observe that when we use IO pattern with bag of words representation, the classification accuracy is improved by **2.5%**. We also use ICPR 2010 challenge dataset. This dataset contains videos

Table 4. Subset of IO patterns obtained from real world dataset

Drought Dataset
P1. <{High_PDSI → High_AvgPcp},{C}{NO}>
P2. <{(Normal_AvgTemp → High_AvgPcp → Normal_PDSI},{M,B,B},{NW,NW,NW}>
P3. <{Low_PDSI → High_AvgTemp → Low_AvgPcp}{C,C,B},{N,N,N}>
Vegetation Dataset
P4. <{Low_CloudCover → Low_NDVI},{M},{NO}>
P5. <{High_Temp → Low_NDVI},{S},{N}>
P6. <{High_Pcp → High_CloudCover → Low_DiurnalTemp},{S,S,E},{N,N,N}
P7. <{High_Pcp → High_VPressure → High_NDVI},{S,S,E}{N,N,NO}
Human Activity Video Dataset
P8. <{88 → 304},{O},{N}> [6:Walking, 1:Running and 1:Jogging]
P9. <{109 → 259},{C},{N}> [1: Walking, 3:Running]
P10. <{302 → 322},{C},{S}> [1:Running, 3:Jogging]

Table 5. Comparing classification results of human activity detection

Datasets	Bag_of_Words	Bag_of_Words+TP	Bag_of_Words+SP	Bag_of_Words+IO
KHT Dataset	79.43	76.45	79.85	**81.93**
ICPR 2010 Challenge Dataset	39.00	40.00	46.53	**55.00**

of executions of 6 classes of human-human interactions: shake-hands, point, hug, push, kick and punch. We use video from set 1(total 60 videos) and obtain spatio temporal words for each execution(i.e., video). We repeat 10 fold cross validation using SVM classifier on initial bag of words dataset and extended dataset. Table 5 lists the results.

6 Related Work

Existing studies have proposed various spatio-temporal patterns from instantaneous features. A co-location pattern is a set of features whose instances are frequently located together in space[13]. A topological pattern [15] is extension of co-location pattern with additional consideration of temporal constraints. Table 6 summarize various ordered patterns proposed for spatio-temporal databases. Each pattern in Table 6 is a sequence of features with constraints in space and time dimensions. All these proposals consider non-transactional database.

Table 6. Spatio-Temporal Patterns

<table>
<tr><td rowspan="2">Patterns</td><td colspan="2">Space Dimension</td><td colspan="2">Time Dimension</td></tr>
<tr><td>Directional Relation</td><td>Spatial Proximity</td><td>Temporal Distance</td><td>Allen's Temporal Relationship</td></tr>
<tr><td>Generalized flow pattern[16,10]</td><td>√</td><td rowspan="4">All features in the sequence are neighbor of each other</td><td rowspan="2">All features in the sequence are neighbor of each other</td><td rowspan="2">Before</td></tr>
<tr><td>Flow pattern[17]</td><td>√</td></tr>
<tr><td>Sequential pattern[6]</td><td>×</td><td rowspan="3">Adjacent features in the sequence are neighbor of each other</td><td rowspan="3">All</td></tr>
<tr><td>Interval-Orientation pattern</td><td>√</td></tr>
<tr><td>Interval patterns</td><td>×</td></tr>
</table>

Dealing with non-transactional database is an issue. [16,17] discretize the space and obtain the transactions. Later, they extend the prefix based pattern growth algorithm for mining desired patterns. Adapting transactional approach to discover IO patterns requires every feature instance must be present in each transaction. Majority of papers that does not discretize the space uses the prevalence index [13,15,5,10] as an interesting measure. The work in [6] use non-antimonote density ratio as an interesting measure for sequential pattern. Further, the proposed algorithm is candidate set generation-and-test approach(i.e., NaiveMiner). Our work use prevalence index as an interesting measure and hash-based joining [11] to generate the IO patterns. The works in [14] deals with moving object databases.

Recently, many algorithms consider incremental mining of traditional transactional database, such as [11,3]. However, these algorithms are not applicable directly for incremental mining. Reason is our underlying database is non-transactional and we use prevalence index measure as an interesting measure for IO patterns. To the best of our knowledge, there is no existing method which can be adapted to discover IO patterns efficiently and incrementally from spatio-temporal databases.

7 Conclusion

In this paper, we discuss an algorithm for mining IO patterns from spatio-temporal databases. We extended the proposed algorithm to work in incremental fashion. The

incremental algorithm reuse already known knowledge to reduce the search space. Experiments on synthetic data indicates efficiency and scalability of the proposed algorithm. Experiments on real world data suggest that we mine useful knowledge. In future, we like to extend our framework such that it allow an interactive analysis.

References

1. Cheng, H., Yan, X., Han, J., Yu, P.S.: Direct discriminative pattern mining for effective classification. In: ICDE, pp. 169–178 (2008)
2. Ding, W., Tomasz, S., Josue, S.: Discovery of geospatial discriminating patterns from remote sensing dataset. In: SDM (2008)
3. Ding, Y., Lee, K., Cheng, H., Krishna, G., Li, Z., Ma, X., Zhou, Y., Han, J.: Cispan: Comprehensive incremental mining algorithms of closed sequential patterns for multi-versional software mining. In: SDM, pp. 84–95. SIAM, Philadelphia (2008)
4. Dollar, P., Rabaud, V., Cottrell, G., Belongie, S.: Behavious recognization via sparse spatio-temporal features. In: VS-PETS, pp. 65–72 (2005)
5. Huang, Y., Zhang, L., Zhang, P.: Can we apply projection based frequent pattern mining paradigm to spatial co-location mining. In: AKDDM. LNCS, pp. 433–448 (2005)
6. Huang, Y., Zhang, L., Zhang, P.: A framework for mining sequential patterns from spatio-temporal event data set. In: TKDE, pp. 433–448 (2008)
7. Kulldorff, M., Athas, W., Feuer, E., Miller, B., Key, C.: Evaluating cluster alarms: A space-time scan statistic and brain cancer in los alamos. In: AJPH, pp. 1377–1380 (1998)
8. Laptev, I.: On space-time interest points. In: IJCV, pp. 107–123 (2005)
9. Mohammadi, S., Janeja, V., Gangopadhyay, A.: Discretized spatio-temporal scan window. In: SIAM (2009)
10. Mohan, P., Shekhar, S., Shine, J., Rogers, J.: Cascading spatio-temporal pattern discovery: A summary of results. In: SDM (2010)
11. Parthasarathy, S., Zaki, M.J., Ogihara, M., Dwarkadas, S.: Incremental and interactive sequence mining. In: KDD (1999)
12. Patel, D., Hsu, W., Lee, M.L.: Mining relationships among interval-based events for classification. In: SIGMOD (2008)
13. Shekhar, S., Huang, Y.: Discovering spatial co-location patterns: A summary of results. In: Jensen, C.S., Schneider, M., Seeger, B., Tsotras, V.J. (eds.) SSTD 2001. LNCS, vol. 2121, pp. 236–256. Springer, Heidelberg (2001)
14. Verhein, F.: Mining complex spatio-temporal seqeunce patterns. In: SDM (2009)
15. Wang, J., Hsu, W., Lee, M.L.: A framework for mining topological patterns in spatio-temporal databases. In: CIKM, pp. 429–436. ACM, New York (2005)
16. Wang, J., Hsu, W., Lee, M.L.: Mining generalized spatio-temporal patterns. In: Zhou, L.-z., Ooi, B.-C., Meng, X. (eds.) DASFAA 2005. LNCS, vol. 3453, pp. 649–661. Springer, Heidelberg (2005)
17. Wang, J., Hsu, W., Lee, M.L., Wang, J.: Flowminer: Finding flow patterns in spatio-temporal databases. In: ICTAI, pp. 14–21. IEEE, Los Alamitos (2004)
18. Wie, L., Shan, M.: Mining temporal co-orientation pattern from spatio-temporal databases. In: Zhou, Z.-H., Li, H., Yang, Q. (eds.) PAKDD 2007. LNCS (LNAI), vol. 4426, pp. 895–903. Springer, Heidelberg (2007)
19. Yang, H., Parthasarathy, S., Mehta, S.: A generalized framework for mining spatio-temporal patterns in scientific data. In: KDD, pp. 716–721. ACM, New York (2005)