IEICE TRANS. INF. & SYST., VOL.E100–D, NO.4 APRIL 2017

PAPER Special Section on Data Engineering and Information Management

Grouping Methods for Pattern Matching over Probabilistic Data Streams

Kento SUGIURA[†a)], ***Nonmember***, **Yoshiharu ISHIKAWA**[†b)], ***Senior Member***, ***and*** **Yuya SASAKI**[††c)], ***Nonmember***

SUMMARY As the development of sensor and machine learning technologies has progressed, it has become increasingly important to detect patterns from *probabilistic data streams*. In this paper, we focus on *complex event processing* based on *pattern matching*. When we apply pattern matching to probabilistic data streams, numerous *matches* may be detected at the same time interval because of the uncertainty of data. Although existing methods distinguish between such matches, they may derive inappropriate results when some of the matches correspond to the real-world event that has occurred during the time interval. Thus, we propose two *grouping* methods for matches. Our methods output groups that indicate the occurrence of complex events during the given time intervals. In this paper, first we describe the definition of groups based on temporal overlap, and propose two grouping algorithms, introducing the notions of *complete overlap* and *single overlap*. Then, we propose an efficient approach for calculating the occurrence probabilities of groups by using deterministic finite automata that are generated from the query patterns. Finally, we empirically evaluate the effectiveness of our methods by applying them to real and synthetic datasets.

key words: *probabilistic data streams, complex event processing, pattern matching, grouping*

1. Introduction

It has become increasingly important to detect patterns from uncertain data, owing to the development of the sensor and machine learning technologies. As a lot of sensing data are obtained from smart phones and wearable devices with various sensors, such as GPS sensors, many researchers are studying their effective utilization. Machine learning techniques are of particular interest in recent years, and they are applied to many fields, such as human activity recognition [1], [2]. When we deal with such analysis results from sensing data, we should consider uncertainty. In machine learning, it is common to select the class with the highest confidence score as the result of classification. However, if the boundary of classes is ambiguous, it may not be appropriate to simply select one classification result. By contrast, we can keep all candidates and their confidence scores as probabilities: they show the analysis results without any loss of information. As an example, consider the *probabilistic data stream* shown in Fig. 1. We assume that the location of a person is monitored continually, using sensing devices. Each value indicates the probability that the person is in the rooms or hall at the time. In this paper, we consider *complex event processing* over such a probabilistic data stream.

We focus on complex event processing based on *pattern matching* over probabilistic data streams. For example, if we want to detect a movement from "`RoomA`" to "`RoomB`" through "`Hall`" in Fig. 1, the corresponding pattern is $p = \langle \texttt{RoomA}^+ \ \texttt{Hall}^+ \ \texttt{RoomB}^+ \rangle$. Note that we use the Kleene plus because we do not know how long the person stays at each location. Let us represent open and closed time intervals as $(t_s : t_e)$ and $[t_s : t_e]$, respectively. Given pattern p, we can detect 35 matches from Fig. 1 in the time interval $[1 : 6]$. Figure 2 shows some of them with the corresponding probabilities, where we represent "`RoomA`", "`Hall`", and "`RoomB`" as "`a`", "`h`", and "`b`", respectively. Each match indicates a movement path. For example, m_2 indicates that the person was in "`RoomA`" at time 1. Then the person passed through "`Hall`" between time 2 and 5, and finally reached "`RoomB`" at time 6. The occurrence probability $P(m_2)$ indicates that the probability of the movement is about 0.31.

Numerous matches may cause a problem when we apply pattern matching to probabilistic data streams. A match is uniquely determined when we apply pattern matching to a non-probabilistic data stream. In the case of probabilistic data streams, however, multiple matches are detected in the same time interval owing to the uncertainty of data. Because it is costly to detect all matches, existing methods se-

event symbol	time step						
	1	2	3	4	5	6	7
RoomA (a)	1.0	0.3	0.1	0.2	0	0	0.3
Hall (h)	0	0.7	0.7	0.7	0.9	0	0.7
RoomB (b)	0	0	0.2	0.1	0.1	1.0	0

Fig. 1 A probabilistic data stream.

match	time step						probability
	1	2	3	4	5	6	
m_1	a	h	b				0.14
m_2	a	h	h	h	h	b	0.3087
m_3	a	a	h	h	h	b	0.1323
m_4		a	h	h	h	b	0.1323
m_5				a	h	b	0.18

Fig. 2 Matches of $\langle \texttt{a}^+ \ \texttt{h}^+ \ \texttt{b}^+ \rangle$.

Manuscript received June 22, 2016.
Manuscript revised November 7, 2016.
Manuscript publicized January 17, 2017.

[†]The authors are with Graduate School of Information Science, Nagoya University, Nagoya-shi, 464–8601 Japan.
[††]The author is with Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565–0871 Japan.
a) E-mail: sugiura@db.ss.is.nagoya-u.ac.jp
b) E-mail: ishikawa@is.nagoya-u.ac.jp
c) E-mail: sasaki@ist.osaka-u.ac.jp

DOI: 10.1587/transinf.2016DAP0014

lect matches with high probabilities [3] or sum the probabilities of matches detected at the same time [3], [4]. However, it may not be appropriate to distinguish among matches because they may correspond to the same real-world event in a given time interval. Let us continue the above example. Using our grouping method described later, we obtain about 0.92 as the probability that a complex event of pattern p has occurred in the time interval [1 : 6]. By contrast, the maximum probability of the matches is at most $P(m_2) \simeq 0.31$. We can get a greater value by summing the probabilities of the matches, but it does not provide the exact occurrence probability of the real-world event. For instance, consider $\{m_2, m_3, m_4, m_5\}$, each of which is concurrently detected at time 6. The sum of the probabilities is about 0.75, which is still less than 0.92.

In this paper, we propose *grouping* methods to summarize numerous matches into groups that indicate the occurrence of complex events in given time intervals. There are some differences between matches detected in the same time interval, but they essentially correspond to the same real-world event. Thus, we aggregate the matches that occur in overlapping time periods. An obtained group is denoted by $g_{[t_s:t_e]}$, where t_s and t_e are start and end time, respectively. The probability $P(g_{[t_s:t_e]})$ denotes the occurrence probability of the group in the time interval $[t_s : t_e]$. We also propose an efficient calculation method for $P(g_{[t_s:t_e]})$, using a deterministic finite automaton (DFA) that is obtained by converting the specified pattern. In the example above, the proposed method can generate a summary output such as $P(g_{[1:6]}) \simeq 0.92$ instead of all 35 matches.

The remainder of the paper is organized as follows. Section 2 introduces related work, and Sect. 3 defines probability spaces, query patterns, and groups as preliminaries. We explain grouping policies in Sect. 4, algorithms for generating groups in Sect. 5, and calculation of the probabilities of groups in Sect. 6. Section 7 evaluates the proposed methods by experiment, and Sect. 8 concludes the paper.

2. Related Work

2.1 Probabilistic Data Streams

Many existing research efforts related to probabilistic data streams focus on processing queries such as selection, projection, aggregation, and top-k queries [5]–[7]. Additionally, some researchers study frequent item mining [8] and clustering [9] for probabilistic data streams.

There are several studies that deal with pattern matching over probabilistic data streams [3], [4], [10]. Reference [10] considers temporal uncertainty of event occurrence. That approach, however, differs from our probabilistic data streams because it does not consider the uncertainty of event occurrence. Reference [4] deals with probabilistic data streams that have the Markov property. In that, probabilities of detected matches are computed at every time step. In contrast to our work, it does not consider when each match is started. Thus, we cannot use those methods for queries where the time intervals of matches are of interest. Reference [3] detects matches efficiently with top-k occurrence probabilities. However, the occurrence of events specified in a query are assumed to be intermittent. In other words, they allow a user to skip any events until the specified events occur. For example, suppose that $p = \langle \mathtt{RoomA}^+ \ \mathtt{Hall}^+ \ \mathtt{RoomB}^+ \rangle$, the stream in Fig. 1, and $k = 1$ are given. They detect $\langle \mathtt{RoomA}_1, \mathtt{Hall}_5, \mathtt{RoomB}_6 \rangle$ with the top-1 probability 0.9. That is, the events in the time interval [2 : 4] are skipped. Note that they ignore Kleene closures (* and $^+$) in their pattern matching because multiple events decrease probabilities, such as $P(\langle \mathtt{Hall}_4, \mathtt{Hall}_5 \rangle) < P(\langle \mathtt{Hall}_5 \rangle)$. Thus, m_2, m_3, and m_4 in Fig. 2 are never detected in their method. This approach may not be appropriate when the underlying real-world events are continuous, as found in human activity recognition. By contrast, our methods can deal with such continuous events because we perform pattern matching without skips.

In our previous work [11], we proposed the basic idea of the grouping methods. This paper has four extensions from the previous work as follows:

1. As the previous work cannot control the granularity of groups, it may output unexpected large groups. In this paper, we control the granularity of groups by introducing *overlap ratios* in Sect. 4.
2. We propose efficient grouping algorithms in Sect. 5. Although the previous paper proposed grouping algorithms that merge all candidates of matches into groups, they were inefficient because some candidates are not detected and removed from groups. Thus, we merge only detected matches in the improved algorithms.
3. The previous work dealt with limited regular expressions because we could not calculate the probabilities of groups with general regular expressions. As we propose the improved calculation method in Sect. 6, we can use general regular expressions in this paper.
4. We evaluate the grouping methods in more detail by using real and synthetic datasets. This paper provides an overall evaluation, such as effectiveness of our methods and effects of various parameters, in Sect. 7.

2.2 Non-Probabilistic Data Streams

In the literature of non-probabilistic data streams, many methods for pattern matching are proposed. The SASE project lets a user specify detailed conditions for pattern matching [12], [13]. In [14]–[16], data streams with special features, such as being disordered or distributed, are treated. Reference [17] considers occurrence frequencies of events and [18] uses field-programmable gate arrays to increase throughput. Reference [19] considers the decidability of pattern matching when ambiguous patterns and infinite data streams are given.

Pattern matching on strings has been well studied. The Thompson construction method [20] is particularly use-

ful. It generates a non-deterministic finite automaton (NFA) from a regular expression pattern. We can easily construct an NFA that implements the Knuth–Morris–Pratt algorithm [21] or the Aho–Corasick algorithm [22] by using the Thompson construction method [23]. In the paper, we use the Thompson construction method to detect matches and calculate the occurrence probabilities of groups.

3. Preliminaries

In this section, we define probability spaces, query patterns, matches, and groups. Table 1 shows the list of symbols used in the paper.

3.1 Probability Space

We define a probability space for pattern matching over a probabilistic data stream. Definition 1 defines a *probabilistic event* as an entry of a probabilistic data stream.

Definition 1: Let a symbol α be a type of an event e, with Σ the universal set of event symbols. A *probabilistic event* e_t is an event with its occurrence probability $P(e_t = \alpha)$ for each $\alpha \in \Sigma$ at time t. $P(e_t = \alpha)$ satisfies the following properties:

$$\forall \alpha \in \Sigma,\ 0 \leq P(e_t = \alpha) \leq 1 \tag{1}$$

$$\sum_{\alpha \in \Sigma} P(e_t = \alpha) = 1 \tag{2}$$

□

Our approach can be extended for probabilistic events with the Markov property as in [4]; we omit the details for the sake of brevity. Definition 2 defines a *probabilistic data stream* in terms of probabilistic events.

Definition 2: A *probabilistic data stream* $PDS = \langle e_i, e_{i+1}, \ldots, e_j, \ldots \rangle$ is a sequence of probabilistic events. □

For example, Fig. 1 shows a probabilistic data stream $PDS = \langle e_1, e_2, e_3, e_4, e_5, e_6, e_7 \rangle$ with the symbols $\Sigma = \{\mathtt{a}, \mathtt{h}, \mathtt{b}\}$. In the following, we represent α_t as $e_t = \alpha$.

Definition 3 introduces the notion of the set of *possible worlds* for a probabilistic data stream. Let $E_t \subseteq \Sigma$ be the set of symbols with non-zero probabilities at time t, and let "$\times$" be a product of symbols to generate event sequences.

Table 1 Notation used in the paper.

Symbol	Description
e_t	a probabilistic event e occurred at time t
Σ	finite event symbols
α_t	$e_t = \alpha$ ($\alpha \in \Sigma$)
PDS	the probabilistic data stream
$(x : y), [x : y]$	open/closed intervals
w	a possible world for PDS
$W_{[i:j]}$	all possible worlds in $[i : j]$
p	the query pattern
m	a match of p
r	a candidate of m
$g_{[i:j]}$	the group of matches occurred in $[i : j]$
θ	the threshold of occurrence probabilities
τ	the threshold of overlap ratios

Definition 3: Given a finite probabilistic data stream $PDS = \langle e_i, e_{i+1}, \ldots, e_j \rangle$, the universal set of possible worlds is $W_{[i:j]} = E_i \times E_{i+1} \times \ldots \times E_j$. The probability of a possible world $w = \langle \alpha_i, \alpha_{i+1}, \ldots, \alpha_j \rangle \in W_{[i:j]}$ is $P(w) = \prod_{\alpha_t \in w} P(\alpha_t)$. □

For example, the set of possible worlds $W_{[2:3]}$ in Fig. 1 is as follows:

$$\begin{aligned} W_{[2:3]} &= \{\mathtt{a}_2, \mathtt{h}_2\} \times \{\mathtt{a}_3, \mathtt{h}_3, \mathtt{b}_3\} \\ &= \{\langle \mathtt{a}_2,\mathtt{a}_3 \rangle, \langle \mathtt{a}_2,\mathtt{h}_3 \rangle, \langle \mathtt{a}_2,\mathtt{b}_3 \rangle, \langle \mathtt{h}_2,\mathtt{a}_3 \rangle, \langle \mathtt{h}_2,\mathtt{h}_3 \rangle, \langle \mathtt{h}_2,\mathtt{b}_3 \rangle\} \end{aligned}$$

Definition 4 defines the *probability space* for a probabilistic data stream using possible worlds.

Definition 4: Let $2^{W_{[i:j]}}$ be the power set of $W_{[i:j]}$. Given a finite probabilistic data stream $PDS = \langle e_i, e_{i+1}, \ldots, e_j \rangle$, the *probability space* is defined as $(2^{W_{[i:j]}}, P)$. Note that P gives each $x \in 2^{W_{[i:j]}}$ the probability $P(x) = \sum_{w \in x} P(w)$. □

3.2 Query Patterns and Matches

Definition 5 defines the grammar for query patterns.

Definition 5: Let α be an event symbol in Σ, and let ϵ be the empty symbol. An input pattern is generated by the following grammar:

$$p ::= \alpha \mid \epsilon \mid p\ p \mid p \vee p \mid p^* \mid p^+ \mid (p) \tag{3}$$

□

In other words, we assume that patterns are specified as regular expressions.

We introduce a *match* and its occurrence probability.

Definition 6: A *match* m is a sequence of symbols that fits a specified pattern p. Given a probability space $(2^{W_{[i:j]}}, P)$, let $W_m \subseteq W_{[i:j]}$ be a set of possible worlds that include m as a subsequence. An occurrence probability of m is $P(m) = \sum_{w \in W_m} P(w)$. □

For example, we consider the case of $W_{[1:4]}$ in Fig. 1. The probability of $m_1 = \langle \mathtt{a}_1, \mathtt{h}_2, \mathtt{b}_3 \rangle$ is calculated by the following possible worlds:

$$\begin{aligned} w_1 &= \langle \mathtt{a}_1, \mathtt{h}_2, \mathtt{b}_3, \mathtt{a}_4 \rangle,\ P(w_1) = 0.028 \\ w_2 &= \langle \mathtt{a}_1, \mathtt{h}_2, \mathtt{b}_3, \mathtt{h}_4 \rangle,\ P(w_2) = 0.098 \\ w_3 &= \langle \mathtt{a}_1, \mathtt{h}_2, \mathtt{b}_3, \mathtt{b}_4 \rangle,\ P(w_3) = 0.014 \end{aligned}$$

Thus, $P(m_1)$ is $P(w_1) + P(w_2) + P(w_3) = 0.14$.

In the paper, matches are detected by the DFA-based approach, as in traditional string matching. Note that we have to detect all possible matches because inputs to the DFA are probabilistic events. First, we generate an NFA by applying the Thompson construction method [20] to the

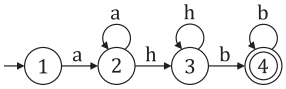


Fig. 3 DFA for pattern $p = \langle \mathtt{a}^+ \mathtt{h}^+ \mathtt{b}^+ \rangle$.

specified pattern. This NFA is converted to an equivalent and minimum DFA using the Hopcroft minimization method [24]. We generate candidate matches for the probabilistic events using the DFA. For example, suppose that the pattern $p = \langle \mathtt{a}^+ \mathtt{h}^+ \mathtt{b}^+ \rangle$ and the stream in Fig. 1 are given. Figure 3 shows the DFA for pattern p. At time 1, we generate a candidate match $r_1 = \langle \mathtt{a}_1 \rangle : P(r_1) = 1.0$, which corresponds to the transition from state 1 to state 2. At time 2, r_1 is extended to $r_{11} = \langle \mathtt{a}_1, \mathtt{a}_2 \rangle : P(r_{11}) = 0.3$ and $r_{12} = \langle \mathtt{a}_1, \mathtt{h}_2 \rangle : P(r_{12}) = 0.7$ because there are two transitions from state 2. We also generate a new candidate $r_2 = \langle \mathtt{a}_2 \rangle : P(r_2) = 0.3$ from the initial state. This process continues until the stream terminates, and we detect matches reaching the final state 4. Note that candidates are rejected only when they cannot transition to any states. In other words, we do not reject matches so long as they can be extended to other matches. In the above example, r_{12} becomes a match $m_1 = \langle \mathtt{a}_1, \mathtt{h}_2, \mathtt{b}_3 \rangle$ at time 3, and we retain m_1 because it can be extended to $m_2 = \langle \mathtt{a}_1, \mathtt{h}_2, \mathtt{b}_3, \mathtt{b}_4 \rangle$ at time 4.

We describe pruning of matches based on their occurrence probabilities. The number of matches increases rapidly when disjunctions ($\vee$) and Kleene closures (* and $^+$) are used. Because such numerous matches decrease the throughput, we allow the user to specify the threshold of occurrence probabilities θ for pruning matches with low probabilities. In the above example, r_{11} and r_2 are pruned at time 2 with the threshold $\theta = 0.5$.

3.3 Groups

Definition 7 defines a *group* as a set of matches.

Definition 7: A *group* g is a set of matches, and its occurrence probability is calculated as follows:

$$P(g) = \sum_{w \in \bigcup_{m \in g} W_m} P(w) \tag{4}$$

We take $g_{[t_s:t_e]}$ as the set of all the matches that have occurred in the time interval $[t_s : t_e]$. □

The occurrence probability of a group g is the probability that any match $m \in g$ has occurred. The probability of $g_{[t_s:t_e]}$ therefore indicates the occurrence probability of the grouped event in $[t_s : t_e]$. We describe the calculation method of the probability for a group in Sect. 6.

In the following sections, we describe the generation method of a group $g_{[t_s:t_e]}$. First, we put matches together into a group g, and then summarize g into $g_{[t_s:t_e]}$ using the start and end time of the matches in g. In other words, the minimum start time of the matches becomes t_s and the maximum end time becomes t_e. For example, $g_{[1:6]}$ is derived from the group of the matches in Fig. 2.

4. Grouping Based on Overlap

This section defines the grouping methods in terms of temporal overlaps of matches. First, we define an *overlap ratio* as a grouping criterion, and then propose *complete overlap* and *single overlap* as grouping policies.

4.1 A Grouping Criterion

We use a temporal *overlap ratio* of matches to decide whether we should merge them as a group. It is important for grouping whether matches overlap with each other. For example, it may not be appropriate to group m_1 and m_5 in Fig. 2 because they do not overlap at all. In contrast, it is probably appropriate to group m_2 and m_3 because they completely overlap. We therefore define the temporal overlap ratio based on the Jaccard similarity ($\mathrm{JS}(A, B) = |A \cap B|/|A \cup B|$). Note that we consider the overlap ratio only when matches overlap. Let $m.t_s$ and $m.t_e$ be the start and end time of a match m, respectively. The overlap ratio between m_i and m_j is as follows:

$$\mathrm{overlap}(m_i, m_j) = \frac{\min(m_i.t_e, m_j.t_e) - \max(m_i.t_s, m_j.t_s) + 1}{\max(m_i.t_e, m_j.t_e) - \min(m_i.t_s, m_j.t_s) + 1} \tag{5}$$

Equation (5) indicates how much m_i and m_j overlap. Note that we add 1 to both the denominator and numerator because each probabilistic event has the time interval. That is, we count the number of time steps. For example, consider m_1 and m_2 in Fig. 2. Their overlap ratio is calculated as follows:

$$\mathrm{overlap}(m_1, m_2) = \frac{3 - 1 + 1}{6 - 1 + 1} = 0.5$$

The overlap score takes the maximal value when the start and end times of m_i are equal to those of m_j, such as m_2 and m_3 in Fig. 2.

Matches m_i and m_j are grouped when their overlap ratio $\mathrm{overlap}(m_i, m_j)$ is greater than the threshold τ given by the user. If we use the overlap ratios for the grouping of matches, we can regard a grouping as a clustering of line segments in one-dimensional space. Thus, we use the ideas of the complete-link and single-link clustering methods [25] to define our grouping policies.

4.2 Complete Overlap

We define the *complete overlap* policy, inspired by the complete-link method [25]. The complete-link method generates clusters such that all items in a cluster are similar to each other. Similarly, the complete overlap policy requires that every match overlaps with all other matches in the same group. Definition 8 is the definition of the complete overlap property.

Definition 8: A group g has the property of *complete overlap* when g satisfies the following condition:

$$\forall m_i, m_j \in g, \text{overlap}(m_i, m_j) > \tau \tag{6}$$

□

For example, the group $g = \{m_2, m_3, m_4, m_5\}$ in Fig. 2 has the property of complete overlap with the threshold $\tau = 0.4$.

We can modify the granularity of groups by adjusting the threshold τ. Let us continue the above example with $g = \{m_2, m_3, m_4, m_5\}$. If we want to cut m_5 from g because its overlap ratio is smaller than that of other matches, we can do it by increasing τ. In this case, by specifying $\tau > 0.5$, we can divide g into $g_1 = \{m_2, m_3, m_4\}$ and $g_2 = \{m_5\}$.

4.3 Single Overlap

We define the *single overlap* policy, inspired by the single-link method [25]. The single-link method generates clusters such that every item is similar to at least one other item in the same cluster. Similarly, the single overlap policy requires that every match overlaps with at least one other match in the same group. Definition 9 is the definition of the single overlap property.

Definition 9: A group g has the property of *single overlap* when g satisfies the following condition:

$$\forall m_i, \exists m_j \in g, m_i \neq m_j \wedge \text{overlap}(m_i, m_j) > \tau \tag{7}$$

□

Single overlap can group matches like a chain. For instance, consider a grouping of the matches in Fig. 2 with $\tau = 0.4$. Complete overlap cannot merge m_1 and m_5 into a group because they do not overlap. On the other hand, single overlap can group them because both overlap(m_1, m_2) and overlap(m_2, m_5) are greater than τ. Thus, all the matches are put together into a group $g = \{m_1, m_2, m_3, m_4, m_5\}$.

Single overlap summarizes the results of pattern matching more than complete overlap does. We may be able to generate a group with a long chain. In addition, all generated groups do not overlap with each other when we generate maximal groups. Thus, we can easily check when group events have occurred.

5. Grouping Algorithms

This section describes the two grouping algorithms. We first explain the case of complete overlap, and then describes that of single overlap.

5.1 Algorithm for Complete Overlap

Complete overlap requires that every match overlaps with all other matches in the same group. The property of complete overlap is preserved by adding a match that overlaps with all the matches in an existing group. Figure 4 shows the

```
   Input: p, PDS, θ, τ
1  G ← ∅                    // candidates of groups
2  R ← ∅                    // candidates of mathces
3  M_t ← ∅                  // matches detected at t
4  foreach e_t ∈ PDS do
5      update R and M_t using p, e_t, and θ
6      if M_t ≠ ∅ then                          // grouping
7          sort M_t in ascending order of their start time
8          updateCompleteOverlapGroups(G, M_t)
9      foreach g ∈ G do                         // output groups
10         if ∀m ∈ g, ∄r ∈ R, overlap(m, r) > τ then
11             t_s ← min_{m∈g}(m.t_s)
12             t_e ← max_{m∈g}(m.t_e)
13             calculate P(g_[t_s:t_e]) and output g_[t_s:t_e]
14             G ← G \ {g}
```

Fig. 4 Algorithm for group generation.

```
1  updateCompleteOverlapGroups(G, M_t)
2  foreach m ∈ M_t do
3      foreach g ∈ G do      // add m to existing groups
4          if ∀m' ∈ g, overlap(m, m') > τ then
5              g ← g ∪ {m}
6      if ∄g ∈ G, m ∈ g then G ← G ∪ {m}
```

Fig. 5 Grouping based on complete overlap.

grouping algorithm. At line 5, we generate matches M_t and their candidates R using the method described in Sect. 3.2. At line 8, we call the procedure in Fig. 5 to generate groups based on complete overlap. At lines 11–14, we output a group $g_{[t_s:t_e]}$ that is derived from g as described in Sect. 3.3.

For example, suppose that the pattern $p = \langle \mathtt{a}^+\ \mathtt{h}^+\ \mathtt{b}^+ \rangle$, the probabilistic data stream PDS in Fig. 1, the threshold of occurrence probabilities $\theta = 0.1$, and the threshold of overlap ratios $\tau = 0.5$ are given. Note that the threshold $\theta = 0.1$ results in the five matches in Fig. 2. First, we update R only until time 3 because further matches are not detected (Fig. 4: 5). At time 3, $M_3 = \{m_1\}$ is detected (Fig. 4: 5). We generate a new group $g_1 = \{m_1\}$ and add it to G because G is still an empty set (Fig. 5: 6). g_1 is not output at time 3 because there are candidates that overlap with $m_1 \in g_1$, such as $\langle \mathtt{a}_1, \mathtt{h}_2, \mathtt{h}_3 \rangle \in R$ (Fig. 4: 9–14). At time 4 and 5, we only update R (Fig. 4: 5). At time 6, $M_6 = \{m_2, m_3, m_4, m_5\}$ is detected (Fig. 4: 5). Note that matches are processed in ascending order of their start time (Fig. 4: 7). m_2 is not added to g_1 because its overlap ratio with $m_1 \in g_1$ is not greater than τ (Fig. 5: 4). We therefore generate $g_2 = \{m_2\}$ and add it to G (Fig. 5: 6). We omit the description of g_1 in the following because $m_1 \in g_1$ does not overlap with all other matches in M_6. m_3 is added to g_2 because its overlap ratio with $m_2 \in g_2$ is greater than τ (Fig. 5: 4–5). Then, g_2 becomes $\{m_2, m_3, m_4\}$ because m_4 overlaps with both m_2 and m_3. Besides, we generate a new group $g_3 = \{m_5\}$ because the overlap ratio between m_5 and $m_2 \in g_2$ is not greater than τ (Fig. 5: 6). R becomes $\{\langle \mathtt{a}_7 \rangle\}$ at time 7. In other words, no candidate overlaps with all matches in any of the groups (Fig. 4: 10). Thus, we output $g_{[1:3]}$, $g_{[1:6]}$, and $g_{[4:6]}$, derived from g_1, g_2, and g_3, respectively (Fig. 4: 11–14).

```
1  updateSingleOverlapGroups(G, M_t)
2  foreach m ∈ M_t do
3      foreach g ∈ G do        // add m to existing groups
4          if ∃m' ∈ g, overlap(m, m') > τ then
5              g ← g ∪ {m}
6      foreach g_i, g_j ∈ G (i ≠ j) do        // grouping via m
7          if m ∈ g_i ∧ m ∈ g_j then
8              g_i ← g_i ∪ g_j
9              G ← G \ {g_j}
10     if ∄g ∈ G, m ∈ g then G ← G ∪ {m}
```

Fig. 6 Grouping based on single overlap.

```
Input: g_[t_s:t_e], Σ
Output: P(g_[t_s:t_e])
1  P(g_[t_s:t_e]) ← 0
2  w ← ⟨⟩        // initialize the possible world
3  enumeratePossibleWorlds(w, g_[t_s:t_e], Σ)

4  enumeratePossibleWorlds(w, g_[t_s:t_e], Σ)
5  foreach α ∈ Σ do
6      add α to the last of w
7      if w includes a match then
8          P(g_[t_s:t_e]) ← P(g_[t_s:t_e]) + P(w)
9      else if |w| < |[t_s : t_e]| then
10         enumeratePossibleWorlds(w, g_[t_s:t_e], Σ)
11     remove the last event from w        // backtrack
```

Fig. 7 The naïve method to calculate the probability of $g_{[t_s:t_e]}$.

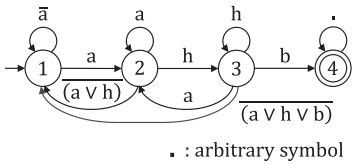


Fig. 8 DFA for $\langle .^* \; \mathtt{a}^+ \; \mathtt{h}^+ \; \mathtt{b}^+ \; .^* \rangle$.

5.2 Algorithm for Single Overlap

Single overlap requires that every match overlaps with at least one other match in the same group. The property of single overlap is ensured by adding a match that overlaps with any match in an existing group. In this, we can group matches that do not overlap with each other by using other overlapping matches. In other words, we can merge groups G into one group via a match m when every group $g \in G$ contains m. For example, suppose that there are $g_1 = \{m_1\}$ and $g_2 = \{m_5\}$ in Fig. 2, where the threshold τ is 0.4. We can add m_2 to both g_1 and g_2 because m_1 and m_5 overlap with m_2. Thus, the property of single overlap is kept even if we merge $g_1 = \{m_1, m_2\}$ and $g_2 = \{m_2, m_5\}$ into one group $g = \{m_1, m_2, m_5\}$ because of m_2.

Figure 6 shows the grouping procedure based on single overlap. We call this procedure at line 8 in Fig. 4. For example, suppose that $p = \langle \mathtt{a}^+ \; \mathtt{h}^+ \; \mathtt{b}^+ \rangle$, PDS in Fig. 1, $\theta = 0.1$, and $\tau = 0.4$ are given. Until time 6, grouping is the same in the case of complete overlap. At time 6, G is $\{g_1 = \{m_1\}\}$ and M_6 is $\{m_2, m_3, m_4, m_5\}$. m_2 and m_3 are added to g_1 because their overlap ratios with $m_1 \in g_1$ are greater than τ (Fig. 6: 4–5). The overlap ratio between m_4 and m_1 is not greater than τ, but we can add m_4 to g_1 because m_4 overlaps with $m_2 \in g_1$. m_5 is also added to g_1, and g_1 becomes $\{m_1, m_2, m_3, m_4, m_5\}$. At time 7, no candidate $r \in R$ overlaps with all matches in g_1 (Fig. 4: 10). Thus, we output $g_{[1:6]}$, derived from g_1 (Fig. 4: 11–14).

6. Efficient Calculation of Group Occurrence Probability

This section explains how to calculate the occurrence probability of a group. First, we describe the generation of a DFA, which plays an important role for our approach, and then explain the calculation method in detail.

6.1 Generation of DFA

The occurrence probability of a group $g_{[t_s:t_e]}$ is the sum of the probabilities of possible worlds that include a match as a subsequence, as shown in Eq. (4). The naïve approach enumerates all possible worlds $W_{[t_s:t_e]}$, and sums the probabilities of the possible worlds that include a match. Figure 7 shows the naïve method using the backtrack algorithm. We represent $|w|$ and $|[t_s : t_e]|$ as the lengths of a possible world w and a time interval $[t_s : t_e]$, respectively. That is, $|w|$ is the number of events in the sequence w, and $|[t_s : t_e]|$ is the number of time steps $(t_e - t_s + 1)$. The naïve method uses the recursive function to enumerate all possible worlds (Fig. 7: 4–11). If we can add events to w, we call the function recursively (Fig. 7: 9, 10) and add events (Fig. 7: 5, 6). We then add the probability of w to $P(g_{[t_s:t_e]})$ when w includes a match (Fig. 7: 7, 8). When we have checked whether w includes a match, we backtrack by removing the last event of w (Fig. 7: 11). The naïve approach, however, is not efficient because the number of possible worlds exponentially increases with the length of the time interval.

To calculate the probability of a group efficiently, we use a DFA that accepts all the possible worlds including a match. It is important to note that we can represent the set of possible worlds for pattern p as $\langle .^* \; p \; .^* \rangle$, where "." is an arbitrary event symbol in Σ. In other words, the possible worlds are sequences that contain arbitrary symbols in any number of times before and after the matches. Thus, we can generate the corresponding DFA by applying the Thompson construction method [20] and the Hopcroft minimization method [24] to $\langle .^* \; p \; .^* \rangle$. For example, Fig. 8 shows the DFA to accept possible worlds that include matches of the pattern $p = \langle \mathtt{a}^+ \; \mathtt{h}^+ \; \mathtt{b}^+ \rangle$. We represent $\overline{\alpha}$ as the negation of an symbol $\alpha \in \Sigma$.

6.2 DFA-Based Calculation

We explain how to calculate the probability of a group using a DFA $(Q, \Sigma, \delta, q_0, F)$. Let Q be the states of the DFA, $\delta : Q \times \Sigma \to Q$ be the transition function, $q_0 \in Q$ be the initial state, and $F \subseteq Q$ be the final states.

q	t_e						
	0	1	2	3	4	5	6
1	1.0	0	0	0.06	0.06	0.08	0.08
2	0	1.0	0.3	0.10	0.17	0	0
3	0	0	0.7	0.70	0.56	0.65	0
4	0	0	0	0.14	0.21	0.27	0.92

Fig. 9 Updating $V^q_{[1:t_e]}$ (rounded to three decimal places).

First of all, we consider possible worlds that arrive at a state in the DFA. We represent $W^q_{[t_s:t_e]}$ as possible worlds that arrive at the state $q \in Q$ in the time interval $[t_s : t_e]$. Let "$\times$" denote the product of possible worlds, as in Definition 3. $W^q_{[t_s:t_e]}$ is calculated by the following recurrence formula:

$$W^q_{[t_s:t_e]} = \bigcup_{q' \in Q, \alpha \in \Sigma \,:\, \delta(q',\alpha)=q} W^{q'}_{[t_s:t_e-1]} \times \{\alpha_{t_e}\} \tag{8}$$

For example, we consider the possible worlds that arrive at state 4 in Fig. 8. States 3 and 4 have a transition to state 4. That is, the possible worlds have to arrive at state 3 or 4 to reach state 4 at the next time step. Thus, the set of possible worlds $W^4_{[1:5]}$ is represented as follows:

$$W^4_{[1:5]} = \left(W^3_{[1:4]} \times \{\mathtt{b}_5\}\right) \cup \left(W^4_{[1:4]} \times \{\cdot_5\}\right)$$

We calculate the sum of the probabilities of possible worlds that arrive at each state. Let $V^q_{[t_s:t_e]}$ be the sum of the probabilities of possible worlds $W^q_{[t_s:t_e]}$. We can derive the following equation from Eq. (8):

$$V^q_{[t_s:t_e]} = \sum_{q' \in Q, \alpha \in \Sigma \,:\, \delta(q',\alpha)=q} V^{q'}_{[t_s:t_e-1]} \cdot P(\alpha_{t_e}) \tag{9}$$

Thus, we can calculate every $V^q_{[t_s:t_e]}$ by applying Eq. (9) recursively, where the probability of the initial state $V^{q_0}_{[t_s:t_s-1]}$ is initialized to 1.0. For example, suppose that the DFA in Fig. 8 and the stream in Fig. 1 are given. Figure 9 shows the change of $V^q_{[t_s:t_e]}$ over the time interval $[1 : 6]$. The probability of each state is calculated using Eq. (9) at each time step as follows:

$$\begin{aligned} V^4_{[1:5]} &= V^3_{[1:4]} \cdot P(\mathtt{b}_5) + V^4_{[1:4]} \cdot P(\cdot_5) \\ &\simeq 0.56 \cdot 0.1 + 0.21 \cdot 1.0 \\ &\simeq 0.27 \end{aligned}$$

The probability of a group $g_{[t_s:t_e]}$ is the sum of the probabilities of possible worlds that arrive at the final states:

$$P(g_{[t_s:t_e]}) = \sum_{q \in F} V^q_{[t_s:t_e]} \tag{10}$$

For example, suppose that the group $g_{[1:6]}$ is generated from the pattern $p = \langle \mathtt{a}^+ \ \mathtt{h}^+ \ \mathtt{b}^+ \rangle$ for the stream in Fig. 1. Figure 9 also shows the process of calculation of $P(g_{[1:6]})$. As the only final state is state 4, $P(g_{[1:6]})$ is about 0.92.

7. Experiments

We analyze the performance of our approach by performing experiments. We constructed a Java-based system that implements the described methods and performed all measurements with the settings given in Table 2.

Table 2 Experiment environment.

OS	Ubuntu 14.04.1 LTS
CPU	Intel Xeon CPU E5620 @ 2.40GHz
Version of JVM	1.8
Memory allocation	8.0GB

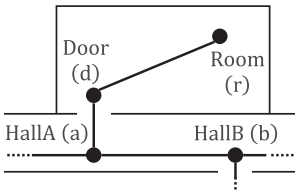


Fig. 10 Graph structure in the real dataset.

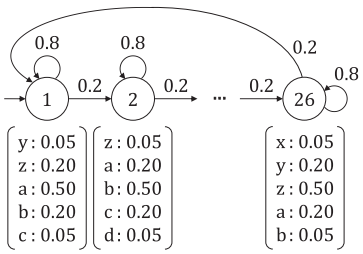


Fig. 11 Markov model for the synthetic dataset.

We use real and synthetic datasets in the experiments. We evaluate the effectiveness of our grouping approach, using the real dataset provided by the Lahar project [4]. The dataset represents an indoor space as a graph, as in Fig. 10, and records the estimated locations of a person walking around the space. For every second, the probability that a person is located at the node is estimated on the basis of sensor data. The dataset also provides the ground truth, a non-probabilistic data stream of user locations. There are nine room entry/exit events in the dataset. In addition, we use a synthetic dataset. We generate a probabilistic data stream that comprises a million events, using the Markov model in Fig. 11 with $\Sigma = \{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$. The model shows the cyclic event occurrence from "a" to "z". The generation starts from the initial state and shifts to the next state according to the probability of each edge. Each state corresponds to each event symbol, such as state 1 and "a", and outputs a probabilistic event that contains five symbols with their probabilities. The probability of each symbol is maximized at the corresponding state, and decreases as the current state moves away from it.

In the following, we first evaluate the effectiveness of grouping. Then, we study the effect of the parameters and policies: the threshold of occurrence probabilities θ, the threshold of overlap ratios τ, and the use of complete or single overlaps. We also evaluate the efficiency of the DFA-based group probability calculation.

Table 3 Outputs corresponding to the sequence c.

Match	Complete overlap	Single overlap
$P(\langle \mathtt{d}_{92}, \mathtt{r}_{93}, \mathtt{d}_{94} \rangle) \simeq 0.06$	$P(g_{[90:149]}) \simeq 0.96$	$P(g_{[90:149]}) \simeq 0.96$
$P(\langle \mathtt{d}_{95}, \mathtt{r}_{96}, \mathtt{d}_{97} \rangle) \simeq 0.03$	$P(g_{[90:148]}) \simeq 0.96$	
$P(\langle \mathtt{d}_{111}, \mathtt{r}_{112}, \mathtt{d}_{113} \rangle) \simeq 0.02$	$P(g_{[90:147]}) \simeq 0.96$	
$P(\langle \mathtt{d}_{95}, \mathtt{r}_{96}, \mathtt{r}_{97}, \mathtt{d}_{98} \rangle) \simeq 0.02$	$P(g_{[101:149]}) \simeq 0.93$	
...		
(about 210,000 matches are obtained)		

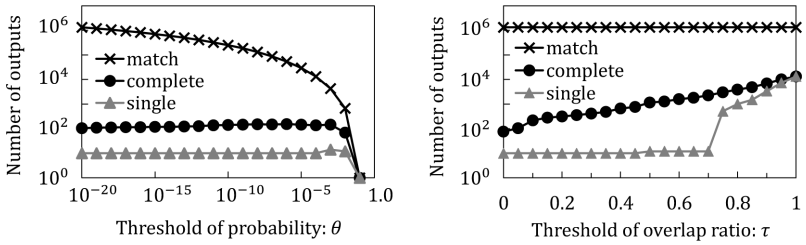


Fig. 12 The number of output matches and groups.

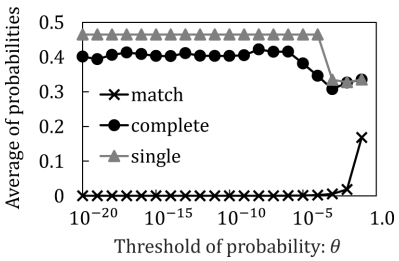


Fig. 13 Averages of occurrence probabilities.

7.1 Effect of Grouping

We evaluate the effectiveness of grouping using the real dataset. We use $p = \langle \mathtt{d}^+ \, \mathtt{r}^+ \, \mathtt{d}^+ \rangle, \theta = 10^{-20}$, and $\tau = 0.05$ as the default settings. In the following, we use the term *correct sequences* to denote agreement with the ground truth.

To show the general trend, Table 3 shows the instances of output matches and groups in descending order of their probabilities. Note that we enumerate only the matches and groups that overlap with the following correct sequence c, because there are a lot of matches and groups, as shown in Fig. 12:

$$c = \langle \mathtt{d}_{94}, \ldots, \mathtt{d}_{108}, \mathtt{r}_{109}, \ldots, \mathtt{r}_{123}, \mathtt{d}_{124}, \ldots, \mathtt{d}_{131} \rangle$$

Table 3 indicates that grouping can detect c with high probability. By contrast, we get too many matches with small probabilities without grouping.

Figure 12 shows the number of output matches and groups. The result indicates that the number of outputs decreases by grouping. There are approximately 100,000 matches that correspond to each correct sequence. It is not realistic to check so many matches. By contrast, we can get small number of groups with a small threshold τ. This gives a realistic number of possibilities to confirm by examining the final results directly.

Figure 13 shows the averages of occurrence probabilities of matches and groups. The result indicates that the occurrence probabilities greatly increase by grouping. As the average probabilities of the groups are rather large, the detected groups imply the occurrence of the corresponding real-world events. In contrast with grouping, simple matching is not useful for detection because the probabilities are extremely small.

7.2 Effect of Parameters

We analyze the effect of parameters on grouping quality. We also use $p = \langle \mathtt{d}^+ \, \mathtt{r}^+ \, \mathtt{d}^+ \rangle, \theta = 10^{-20}$, and $\tau = 0.05$ as the default settings. First, we explain the evaluation metrics. Then, we evaluate the effects of θ, τ, and the two overlap policies, in this order.

7.2.1 Evaluation Metrics

We use precision and recall to analyze grouping quality. Let G be the set of groups and C be the set of correct sequences. Precision indicates the accuracy of detection, and is usually calculated by $|G \cap C|/|G|$. However, we cannot calculate the precision for groups because the form of $g \in G$ differs from that of $c \in C$. We therefore introduce a precision formula for this context based on the overlap ratios. Let us use $\max_{c \in C}(\text{overlap}(g, c))$ as the overlap ratio between g and C. We regard the expected value of overlap ratios of G as the precision score:

$$\begin{aligned} &\text{precision}(G, C) \\ &= \sum_{g \in G} P(g|G) \cdot \max_{c \in C}(\text{overlap}(g, c)), \end{aligned} \tag{11}$$

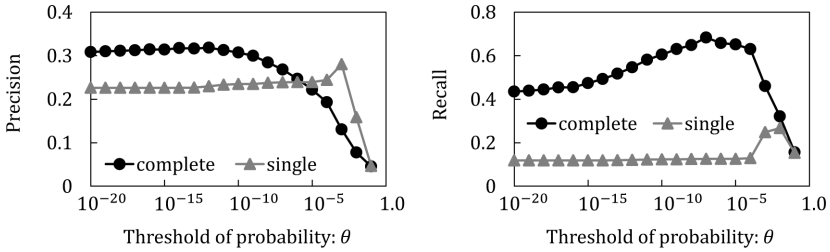


Fig. 14 Precision and recall of $\langle \mathtt{d}^+ \rangle$ for different values of θ.

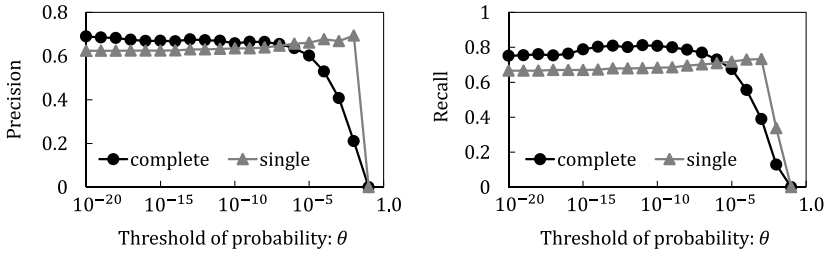


Fig. 15 Precision and recall of $\langle \mathtt{d}^+\ \mathtt{r}^+\ \mathtt{d}^+ \rangle$ for different values of θ.

where $P(g|G)$ is the normalized probability of g in G:

$$P(g|G) = \frac{P(g)}{\sum_{g' \in G} P(g')} \tag{12}$$

Recall indicates the completeness of detection, and is usually calculated by $|G \cap C|/|C|$. In this paper, we regard the average of the overlap ratios of C as recall:

$$\text{recall}(G, C) = \sum_{c \in C} \frac{1}{|C|} \cdot \max_{g \in G}(\text{overlap}(g, c)) \tag{13}$$

7.2.2 Changing the Threshold of Probability θ

Figures 14 and 15 show the precision and recall scores of two patterns $\langle \mathtt{d}^+ \rangle$ (Fig. 14) and $\langle \mathtt{d}^+\ \mathtt{r}^+\ \mathtt{d}^+ \rangle$ (Fig. 15). Note that we reduce θ from 10^{-1} to 10^{-20}. That is, we explain the figures from right to left. The precision and recall initially increase by reducing θ because necessary matches are not detected with high θ. The scores, however, become flat or decrease after reducing θ to some extent. It indicates that unnecessary matches are detected and grouped because of too small θ. As unnecessary matches generate a group $g_{[t_s:t_e]}$ with larger time interval compared with the correct sequence, such as $g_{[90:149]}$ in Table 3, the precision and recall decrease. Note that precision and recall are lower for simple patterns such as $\langle \mathtt{d}^+ \rangle$ because unnecessary matches are more often detected. Additionally, precision and recall rapidly increase and decrease with single overlap because it groups more matches than complete overlap does.

Figure 16 shows the throughputs for different θ values. We use the synthetic dataset and the settings $p = \langle \mathtt{a}^+\ \mathtt{b}^+\ \mathtt{c}^+ \rangle$ and $\tau = 0.05$. We also explain the figure from right to left. The throughputs decrease as we reduce θ. The computation time for a grouping depends on the number of matches. Thus, the throughputs decrease with small θ because more matches are detected, as shown in Fig. 12.

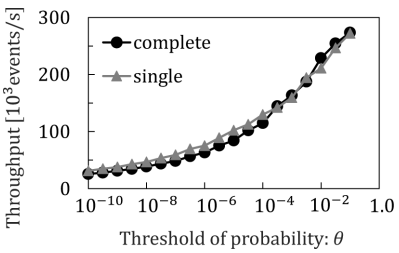


Fig. 16 Throughputs for different values of θ.

From the results above, we can conclude that the threshold θ should set to detect enough matches for grouping. We can estimate such a θ by using the uncertainty of an input data stream and the lengths of assumed correct sequences. For example, consider the real dataset. The maximum probability is about 0.7 for every time, and the lengths of the correct sequences are 30 to 50 for $p = \langle \mathtt{d}^+\ \mathtt{r}^+\ \mathtt{d}^+ \rangle$. Thus, we have to reduce θ to at least $0.7^{50} \simeq 10^{-8}$. This approximately corresponds to the precision and recall results in Fig. 15.

7.2.3 Changing Threshold of Overlap Ratio τ

Figure 17 shows the precision and recall scores for different τ values. The precision decreases as we increase τ. The reason is that smaller groups are generated by increasing τ. For example, consider the case of Table 3. When we use large values of τ, short matches, such as $\langle \mathtt{d}_{90}, \mathtt{r}_{91}, \mathtt{d}_{92} \rangle$, fall apart from $g_{[90:149]}$. Such short matches generate smaller groups, such as $g_{[90:118]}$ and $g_{[90:92]}$. Because the lengths of groups decrease as we increase τ, the overlap ratios (i.e., precision) between the groups and the correct sequences also decrease. However, the precision improves by raising τ close to 1.0 because, conversely, we generate more groups that almost

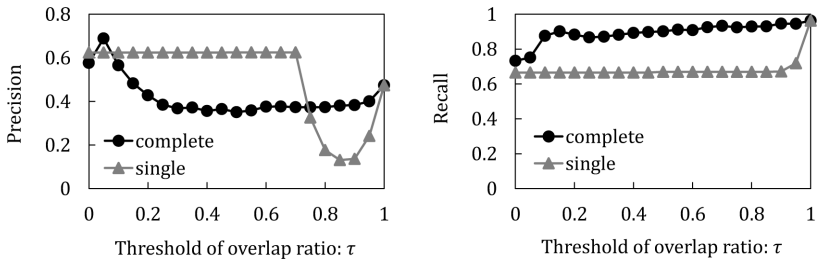


Fig. 17 Precision and recall for different values of τ.

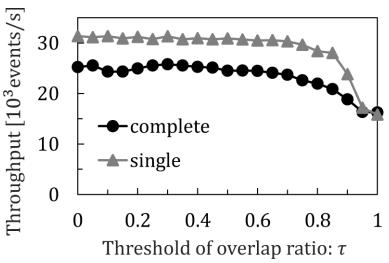


Fig. 18 Throughputs for different values of τ.

overlap with the correct sequences, such as $g_{[94:131]}$. Note that single overlap is slow to respond to changes in τ, as shown in Fig. 12. Thus, the precision increases/decreases suddenly and significantly.

On the other hand, the recall increases as we increase τ. When we use small values of τ, we generate groups with a longer time interval compared with the correct sequences, as shown in Table 3. Because we can generate more appropriate groups, such as $g_{[94:131]}$, by raising τ, recall improves.

Figure 18 shows the throughputs for different τ values. We use the synthetic dataset and the settings $p = \langle \mathsf{a}^+ \, \mathsf{b}^+ \, \mathsf{c}^+ \rangle$ and $\theta = 10^{-10}$. The throughputs decrease by increasing τ, owing to the number of groups. As the number of groups increases with high τ, as shown in Fig. 12, we need more computation time to add the matches to the groups.

From the results above, we conclude that we should choose a threshold τ according to the user's requirement. When we need to detect groups that are similar to correct sequences, a large value is required for τ. Such a τ, however, decreases precision and increases the number of output groups. Thus, we should use a smaller value for τ if we want to look over the results of pattern matching.

7.2.4 Comparison of Complete Overlap and Single Overlap

We examine the difference between complete overlap and single overlap. Because the condition of single overlap is not strict in contrast to that of complete overlap, more matches are put together into one group. When we use a small value for τ, single overlap can generate groups that correspond to correct sequences at a one-to-one level, as in Table 3. However, we may group matches excessively by using single overlap when real-world events occur within short intervals. For example, consider the recall of $p = \langle \mathsf{d}^+ \rangle$. Two real-world events of p occur when the person enters and exits the room. Because single overlap puts together all the matches that correspond to the two real-world events, the recall of single overlap is extremely low, as shown in Fig. 14. In contrast, complete overlap can distinguish the two events because the condition of complete overlap is strict. Thus, the recall of complete overlap is greater than that of single overlap. The number of groups, however, increases with complete overlap, as shown in Fig. 12.

We suggest using the two overlap policies according to the desired treatment. Single overlap is better at looking over the entire data stream, while complete overlap can analyze the stream in detail. Because single overlap generates groups that correspond to real-world events at a one-to-one level, we can guess when the events have occurred. Then, we can apply complete overlap to the time intervals of the detected groups for the detailed analysis.

7.3 Efficiency of DFA-Based Calculation

We study the efficiency of the DFA-based calculation in Sect. 6. We compare the proposed method with the naïve one based on the computation time for calculating the probability of a group. Given a group $g_{[t_s:t_e]}$, our method uses Eq. (9) over the time interval $[t_s : t_e]$ for the calculation. Thus, the computation time of our method depends on the length of the time interval $|[t_s : t_e]| = t_e - t_s + 1$. The computation time of the naïve method also depends on $|[t_s : t_e]|$ because the number of possible worlds relies on $|[t_s : t_e]|$. In the experiments, we therefore measure the computation time for calculating the probability of a group over the given time interval. We first detect a group $g_{[t_s:t_e]}$, and change the end of the time interval of the group from t_s to t_e. That is, we generate groups $g_{[t_s:t_s]}, g_{[t_s:t_s+1]}, \ldots, g_{[t_s:t_e]}$. We then measure the computation time for calculating the probability of each group with the proposed and naïve methods.

Figure 19 shows the computation time where the synthetic dataset and $p = \langle \mathsf{a}^+ \, \mathsf{b}^+ \, \mathsf{c}^+ \rangle$ are used. We omit the computation time of $|[t_s : t_e]| = 1$ and 2 because there are no matches in these time intervals. The result indicates that the computation time of our method is markedly smaller than that of the naïve method. Because the number of possible worlds increases exponentially, the computation time of the naïve method also increases rapidly. In contrast, the computation time of our method increases slowly, because $|[t_s : t_e]|$ affects only the number of uses of Eq. (9).

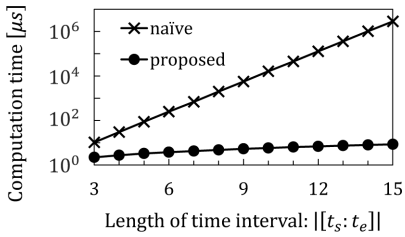


Fig. 19 Computation time for $P(g_{[t_s:t_e]})$.

8. Conclusion

We proposed methods of grouping matches from pattern matching over probabilistic data streams. We defined the overlap ratio as the grouping criterion, and proposed complete overlap and single overlap as grouping policies. Then, we explained the grouping algorithms based on the two overlap policies, and proposed DFA-based calculation of the occurrence probabilities of groups. We evaluated the effectiveness and efficiency of our approach by performing experiments. Future work includes the refinement of the grouping policies and support for detailed conditions for pattern matching.

Our methods extend applications of complex event processing based on regular expressions from non-probabilistic data to uncertain data. Many researchers have proposed pattern matching methods to detect complex events from non-probabilistic data streams, such as stock data [12]–[19]. However, we have to deal with probabilistic data streams, such as human activity data, in some applications. For example, suppose lifelog services using smartphones. Although we can predict user's activity by using sensing data and machine learning techniques, the predictions are uncertain because of the noise of sensing data and the limitation of machine learning techniques. In probabilistic data streams, even if simple queries, such as $\langle \texttt{walk jog}^{+} \texttt{walk} \rangle$, are given, existing methods may detect inappropriate matches and calculate misleading occurrence probabilities. In contrast, our methods detect real-world events with their exact occurrence probabilities using the grouping methods. As probabilistic data streams appear in other applications, such as speech recognition and activity recognition in video, our methods have a potential for various utilization.

Acknowledgments

This research was partially supported by KAKENHI (16H01722, 26540043, 15K21069) and the Center of Innovation Program from Japan Science and Technology Agency (JST).

References

[1] J. Yin, Q. Yang, and J.J. Pan, "Sensor-based abnormal human-activity detection," IEEE Trans. Knowl. Data Eng., vol.20, no.8, pp.1082–1090, 2008.

[2] L. Chen, C. Nugent, and H. Wang, "A knowledge-driven approach to activity recognition in smart homes," IEEE Trans. Knowl. Data Eng., vol.24, no.6, pp.961–974, 2012.

[3] Z. Li, T. Ge, and C.X. Chen, "ε-matching: Event processing over noisy sequences in real time," Proc. 2013 ACM SIGMOD Int. Conf. Management Data, pp.601–612, 2013.

[4] C. Ré, J. Letchner, M. Balazinska, and D. Suciu, "Event queries on correlated probabilistic streams," Proc. 2008 ACM SIGMOD Int. Conf. Management Data, pp.715–728, 2008.

[5] T.T.L. Tran, L. Peng, Y. Diao, A. McGregor, and A. Liu, "CLARO: Modeling and processing uncertain data streams," VLDB J., vol.21, no.5, pp.651–676, 2012.

[6] G. Cormode and M. Garofalakis, "Sketching probabilistic data streams," Proc. 2007 ACM SIGMOD Int. Conf. Management Data, pp.281–292, 2007.

[7] C. Jin, K. Yi, L. Chen, J.X. Yu, and X. Lin, "Sliding-window top-k queries on uncertain streams," Proc. VLDB Endow., vol.1, no.1, pp.301–312, 2008.

[8] Q. Zhang, F. Li, and K. Yi, "Finding frequent items in probabilistic data," Proc. 2008 ACM SIGMOD Int. Conf. Management Data, pp.819–832, 2008.

[9] C.C. Aggarwal and P.S. Yu, "A framework for clustering uncertain data streams," 2008 IEEE 24th Int. Conf. Data Eng., pp.150–159, 2008.

[10] H. Zhang, Y. Diao, and N. Immerman, "Recognizing patterns in streams with imprecise timestamps," Proc. VLDB Endow., vol.3, no.1-2, pp.244–255, 2010.

[11] K. Sugiura, Y. Ishikawa, and Y. Sasaki, "Grouping methods for pattern matching in probabilistic data streams," 20th Int. Conf. Database Syst. Advanced Appl., pp.92–107, 2015.

[12] E. Wu, Y. Diao, and S. Rizvi, "High-performance complex event processing over streams," Proc. 2006 ACM SIGMOD Int. Conf. Management Data, pp.407–418, 2006.

[13] H. Zhang, Y. Diao, and N. Immerman, "On complexity and optimization of expensive queries in complex event processing," Proc. 2014 ACM SIGMOD Int. Conf. Management Data, pp.217–228, 2014.

[14] M. Liu, M. Li, D. Golovnya, E.A. Rundensteiner, and K.T. Claypool, "Sequence pattern query processing over out-of-order event streams," 2009 IEEE 25th Int. Conf. Data Eng., pp.784–795, 2009.

[15] B. Chandramouli, J. Goldstein, and D. Maier, "High-performance dynamic pattern matching over disordered streams," Proc. VLDB Endow., vol.3, no.1-2, pp.220–231, 2010.

[16] M. Akdere, U. Çetintemel, and N. Tatbul, "Plan-based complex event detection across distributed sources," Proc. VLDB Endow., vol.1, no.1, pp.66–77, 2008.

[17] Y. Mei and S. Madden, "ZStream: A cost-based query processor for adaptively detecting composite events," Proc. 2009 ACM SIGMOD Int. Conf. Management Data, pp.193–206, 2009.

[18] L. Woods, J. Teubner, and G. Alonso, "Complex event detection at wire speed with FPGAs," Proc. VLDB Endow., vol.3, no.1-2, pp.660–669, 2010.

[19] S. Santini, "Querying streams using regular expressions: Some semantics, decidability, and efficiency issues," VLDB J., vol.24, no.6, pp.801–821, 2015.

[20] K. Thompson, "Programming techniques: Regular expression search algorithm," Commun. ACM, vol.11, no.6, pp.419–422, 1968.

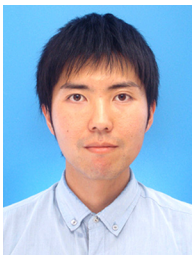
[21] D.E. Knuth, J.H. Morris, Jr., and V.R. Pratt, "Fast pattern matching in strings," SIAM J. Comput., vol.6, no.2, pp.323–350, 1977.

[22] A.V. Aho and M.J. Corasick, "Efficient string matching: An aid to bibliographic search," Commun. ACM, vol.18, no.6, pp.333–340, 1975.

[23] I. Nakata, "Generation of pattern-matching algorithms by extended regular expressions," Japan Soc. Softw. Sci. Tech., vol.5, pp.1–9, 1993.

[24] J.E. Hopcroft, "An n log n algorithm for minimizing states in a finite

automaton," Tech. Rep., Stanford University, 1971.
[25] A.K. Jain, M.N. Murty, and P.J. Flynn, "Data clustering: A review," ACM Comput. Surv., vol.31, no.3, pp.264–323, 1999.



Kento Sugiura is a Ph.D. student in Graduate School of Information Science, Nagoya University. He received the B.S. in Engineering and M.S. in Information Science degrees from Nagoya University in 2013 and 2015, respectively. His research interests include data stream processing, uncertain data management, and simulation data warehouses. He is a student member of DBSJ and IPSJ.

Yoshiharu Ishikawa is a professor in Graduate School of Information Science, Nagoya University. His research interests include spatio-temporal databases, mobile databases, scientific databases, data mining, and Web information systems. He is a member of the Database Society of Japan, IPSJ, IEICE, JSAI, ACM, and IEEE Computer Society.

Yuya Sasaki received the B.E., M.E., and Ph.D. degrees from Osaka University, Japan, in 2009, 2011, and 2014, respectively. He is currently an Assistant Professor in Graduate school of Information Science and Technology, Osaka University. His research interests include database system, mobile environments, and information retrieval.