Mining Uncertain Sequential Patterns in iterative MapReduce

Jiaqi Ge[1], Yuni Xia[1], and Jian Wang[2]

[1] Department of Computer & Information Science, Indiana University Purdue University Indianapolis, Indiana, USA, 46202
[2] School of Electronic Science and Engineering, Nanjing University, Jiangsu, China, 210023
[1] `{jiaqige,yxia}@cs.iupui.edu`, [2] `wangjnju@nju.edu.cn`

Abstract. This paper proposes a sequential pattern mining (SPM) algorithm in large scale uncertain databases. Uncertain sequence databases are widely used to model inaccurate or imprecise timestamped data in many real applications, where traditional SPM algorithms are inapplicable because of data uncertainty and scalability. In this paper, we develop an efficient approach to manage data uncertainty in SPM and design an iterative MapReduce framework to execute the uncertain SPM algorithm in parallel. We conduct extensive experiments in both synthetic and real uncertain datasets. And the experimental results prove that our algorithm is efficient and scalable.

Keywords: Uncertain Databases, Sequential Pattern Mining

1 Introduction

Sequential pattern mining (SPM) is an important data mining application. It provides inter-transactional analysis for timestamped data which are modeled by sequence databases. In real applications, uncertainty is almost everywhere and it may cause probabilistic event existence in sequence databases. For example, in an employee tracking RFID network, the tag read by sensors are modeled by a relation *see(t, aId, tId)*, which denotes that the RFID tag tId is detected by an antenna aId at time t. Since an RFID sensor can only identify a tag with a certain probability within its working range, the PEEX system [10] outputs an uncertain event such as $meet$(100, Alice, Bob, 0.4), which indicates that the event that Alice and Bob meet at time 100 happens with probability 0.4.

Possible world semantics is widely used to interpret uncertain databases[5, 17]; however, it also brings efficiency and scalability challenges to uncertain SPM problems. Meanwhile, applications in the areas of biology, Internet and business informatics encounter limitations due to large scale datasets. While MapReduce is a widely used programming framework for processing big data in parallel, its basic framework can not directly be used in SPM because it does not support the iterative computing model which is required by most SPM algorithms.

In this paper, we propose a sequential pattern mining algorithm in iterative MapReduce for large scale uncertain databases. Our main contributions are summarized as follows:
(1) We use possible world semantics to interpret uncertain sequence databases and analyze the naturally correlated possible worlds.
(2) We design a vertical format of uncertain sequence databases in which we save and reuse intermediate computational results to significantly reduces the time complexity.
(3) We design an iterative MapReduce framework to execute our uncertain algorithm in parallel.
(4) Extensive experiments are conducted in both synthetic and real uncertain datasets, which prove the efficiency and scalability of our algorithm.

2 Related works

A lot of traditional database and data mining techniques have been extended to be applied to uncertain data [2]. Muzammal and Raman propose the SPM algorithm in probabilistic database using expected support to measure pattern frequentness, which has weakness in mining high quality sequential patterns[14, 15]. Zhao et al. define probabilistic frequent sequential patterns using possible world semantics and propose their complimentary uncertain SPM algorithm UPrefixSpan [17, 18]; however, it uses the depth-first strategy to search frequent patterns and cannot be directly extended to MapReduce framework. A dynamic programming approach of mining probabilistic spatial-temporal frequent sequential patterns is introduced in [11]; Wan et al. [16] propose a dynamic programming algorithm of mining frequent serial episodes within an uncertain sequence. However, dynamic programming also cannot be directly extended to MapReduce.

Jeong et al. propose a MapReduce framework for mining sequential patterns in DNA sequences with only four distinct items [8], in contrast to this paper where unlimited number of items are allowed; Chen et al. extend the classic SPAM algorithm to its MapReduce version SPAMC [7]. However, SPAMC relies on a global bitmap and it is still not scalable enough for mining extremely large databases. Miliaraki et al. propose a gap-constraint frequent sequence mining algorithm in MapReduce [13]. However, all these algorithms are applied in the context of deterministic data, while our work aims to solve large scale uncertain SPM problems.

3 Problem statement

3.1 Uncertain Model

An uncertain database contains a collection of uncertain sequences. An uncertain sequence is an ordered list of uncertain events. An uncertain event is represented by $e = \langle sid, eid, I, p_e \rangle$. Here sid is the sequence id and eid is the event id. $\langle sid, eid \rangle$ identifies a unique event. I is an itemset that describes event e, and p_e is

sid	eid	I	Pe
1	1	{A,B}	0.8
1	2	{C}	0.8
2	1	{B}	1
2	2	{C}	0.8
2	3	{C}	0.4

Fig. 1. An example of uncertain database

wid	Possible world
1	<e_{11}><e_{21}, e_{22}, e_{23}>
2	<e_{11}, e_{12}><e_{21}, e_{22}>
3	<e_{11}, e_{12}><e_{21}, e_{23}>
4	<e_{11}, e_{12}><e_{21}, e_{22}, e_{23}>
...	...

Fig. 2. possible worlds table

the existential probability of event e. Figure 1 shows an example of an uncertain sequence database. Here, for instance, the uncertain event $e_{11} = \langle 1, 1, \{AB\}, 0.8\rangle$ indicates that the itemset $\{AB\}$ occurs in e_{11} with probability 0.8.

We use possible world semantics to interpret uncertain sequence databases. A possible world is instantiated by generating every event according to its existential probability. The number of possible worlds grows exponentially to the number of sequences and events. It is widely assumed that uncertain sequences in the sequence database are mutually independent, which is known as the *tuple-level independence* [9, 2] in probabilistic databases. Events are also assumed to be independent of each other [5, 17], which can be justified by the assumption that events are often observed independently in real world applications. Therefore, we can compute the existential probability of a possible world w in Equation (1).

$$P_e(w) = \prod_{\forall d_i \in w} \{ \prod_{\forall e_{ij} \in d_i} P(e_{ij}) * \prod_{e_{ij} \notin d_i} (1 - P(e_{ij}))\} \quad (1)$$

Where $d_i \in w$ is a sequence in w and $e_{ij} \in d_i$ is an event in d_i. Here e_{ij} is instantiated from the original database and $P(e_{ij})$ is its existential probability. Figure 2 is a table which contains four possible worlds of the uncertain sequence database in Figure 1. Then, for example, we can compute the existential probability of possible world w_1 by $P(w_1) = (0.8 * 0.2) * (1 * 0.8 * 0.4) = 0.0512$.

When one or more uncertain event occurs multiple times in the sequence such as C in e_{22} and e_{23}, some possible worlds are correlated even under the independent assumptions. For example, w_1 and w_3 in Figure 2 are correlated in supporting a pattern, because each event in w_1 is also present in w_3.

3.2 Uncertain SPM problem

A sequential pattern $\alpha = \langle X_1 \cdots X_n\rangle$ is *supported* by a sequence $\beta = \langle Y_1 \cdots Y_m\rangle$, denoted by $\alpha \sqsubseteq \beta$, if and only if there exists integers $1 \leq k_1 < \cdots < k_n \leq m$ so that $X_i.I \subseteq Y_{k_i}.I, \forall i \in [1, n]$. In deterministic databases, a sequential pattern s is frequent if and only if it satisfies $sup(s) \geq t_s$, where $sup(s)$ is the total number of sequences that support s and t_s is the user-defined minimal threshold. In an uncertain database D, the frequentness of s is probabilistic and it can be computed by Equation (2).

$$P(sup(s) \geq t_s) = \sum_{\forall w, sup(s|w) \geq t_s} P(w) \tag{2}$$

Where w is a possible world in which s is frequent and $P(w)$ is the existential probability of w.

Then the uncertain sequential pattern mining problem is defined as follows. *Given an uncertain sequence database D, a minimal support threshold t_s and a minimal frequentness probability threshold t_p, find every probabilistic frequent sequential pattern s in D which has $P(sup(s) \geq t_s) \geq t_p$.*

4 Solution

4.1 Approximation of Frequentness Probability

Suppose $D = \{d_1, \ldots, d_n\}$ is an uncertain database and s is a sequential pattern. Because $d_1, \ldots, d_n$ in D are mutually independent, the probabilistic support of s in D, denoted by $sup(s)$, can be computed by Equation (3).

$$sup(s) = \sum_{i=1}^{n} sup(s|d_i) \tag{3}$$

Where $sup(s|d_i)$ $(i = 1, \ldots, n)$ are Bernoulli random variables, whose success probabilities are $P(sup(s|d_i) = 1) = P(s \sqsubseteq d_i)$. And we will discuss the computation of $P(s \sqsubseteq d_i)$ in section 4.2.

We find that $sup(s)$ is a Poisson-Binomial random variable, because it is the sum of n independent but non-identical Bernoulli random variables. And $sup(s)$ can be modeled by its probability mass function (pmf), denoted by $sup(s) = \{sup(s)|0 : p_0, 1 : p_1, \ldots, n : p_n\}$. Here $n = |D|$ is the number of sequences in D.

According to *central limit theorem*, $sup(s)$ converges to the Gaussian distribution when n goes to infinity. Therefore, in the large scale database D, we can approximate the distribution of $sup(s)$ by Equation (4).

$$sup(s) = \sum_{i=1}^{n} X_i \longrightarrow N(\sum_{i=1}^{n} p_i, \sum_{i=1}^{n} p_i * (1 - p_i)) \tag{4}$$

Here we approximate $sup(s)$ by the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, and then the approximated frequentness probability $P(sup(s) \geq t_s)$ can be computed in linear time.

4.2 Support Probability

The support probability $P(s \sqsubseteq d)$ is the probability that a sequential pattern s is supported by an uncertain sequence d and it can be computed in (5) according to possible world semantics.

$$P(s \sqsubseteq d) = \sum_{\forall w, s \sqsubseteq w} P(w) \tag{5}$$

Where w is a possible world of d which supports s and $p(w)$ is its existential probability. However, suppose each item in a k-length pattern s has m multiple occurrences in d in average, there are $O(k^m)$ possible worlds that may support s in the worst case. And directly enumerating all of them is usually too complex in practice.

Therefore, we design an incremental approach to compute support probability efficiently. Let l be the last item of sequential pattern s. In uncertain sequence d, suppose there are q possible occurrences of l in events $e_{k_1}, \ldots, e_{k_q}$, then all the possible worlds that may support s can be divided into q disjoint subsets $(g_1, \ldots, g_q)$ by the most recent occurrence of item l.

Let $P(g_i)$ be the probability that the latest occurrence of item l (the last item of s) is in e_{k_i}, then it can be computed by Equation (6).

$$P(g_i) = P(l \in e_{k_i}) * \prod_{t=i+1}^{q} P(l \notin e_{k_t}) \tag{6}$$

The amortized cost of Equation (6) is $O(1)$, when events are pre-sorted by their *eid*s. And the support probability $P(s \sqsubseteq d)$ can be computed in (7).

$$P(s \sqsubseteq d) = \sum_{i=1}^{q} P(s \sqsubseteq d|g_i) * P(g_i) = P(s \sqsubseteq d \cap g_i) \tag{7}$$

For example, given $d = \big\langle(B : 0.5)(C^1 : 0.4)(C^2 : 0.4)\big\rangle$ and $s = \langle BC\rangle$, according to possible world semantics, there are three possible worlds of d that may support s: $w_1 = \{BC^1\}$, $w_2 = \{BC^2\}$ and $w_3 = \{BC^1C^2\}$, and we divide them into two disjoint groups by the latest occurrence of item C in the possible worlds as $g_1 = \{w_1\}$ and $g_2 = \{w_2, w_3\}$. We first compute $P(g_1) = 0.4 * 0.6 = 0.24$ and $P(g_2) = 0.4$, then we have $P(s \sqsubseteq d) = 0.5 * 0.24 + 0.5 * 0.4 = 0.22$.

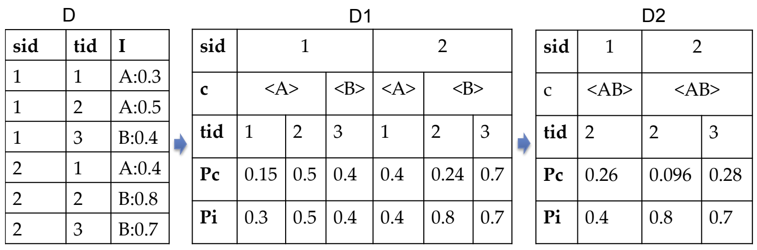
Suppose l is the last item of s, then $s' = s - \{l\}$ is a $(k-1)$-length sequential pattern. $P(s \sqsubseteq d|g_i)$ in (7) can be computed by (8).

$$P(s \sqsubseteq d|g_i) = \sum_{j=1}^{p} P(s' \sqsubseteq d|g_j) * P(g_j|g_i) = \sum_{j=1}^{p} P(s' \sqsubseteq d \cap g_j) * \delta(g_i, g_j) \tag{8}$$

Where g_j ($\forall j \in [1, p]$) are p disjoint subsets of possible worlds in which the latest occurrence of the last item of s' in the event e_{k_j}. And $\delta(g_j, g_i) = 1$, if the last item of s' occurs before the last item of s; otherwise, $\delta(g_j, g_i) = 0$.

By substituting (8) into (7), we can compute the support probability in (9).

$$P(s \sqsubseteq d) = \sum_{i=1}^{q} \sum_{j=1}^{p} P(s' \sqsubseteq d \cap g_j) * P(g_i) * \delta(g_i, g_j) \tag{9}$$



D

sid	tid	I
1	1	A:0.3
1	2	A:0.5
1	3	B:0.4
2	1	A:0.4
2	2	B:0.8
2	3	B:0.7

D1

sid	1			2		
c	<A>		<B>	<A>	<B>	
tid	1	2	3	1	2	3
Pc	0.15	0.5	0.4	0.4	0.24	0.7
Pi	0.3	0.5	0.4	0.4	0.8	0.7

D2

sid	1	2	
c	<AB>	<AB>	
tid	2	2	3
Pc	0.26	0.096	0.28
Pi	0.4	0.8	0.7

Fig. 3. An example of constructing the vertical data structure

Therefore, if we save and reuse the values of $P(s' \sqsubseteq d \cap g_j)$, we can avoid repeated computation which reduces the time complexity of support probability computation from exponential to $O(p * q)$.

4.3 Vertical Data Structure

We develop a vertical data format D_k to save occurrences of k-length candidate patterns. The schema of D_k is $\langle sid, c, tid, P_c, P_i \rangle$, where sid identifies an uncertain sequence d, c is a candidate pattern and $\langle tid, P_c, P_i \rangle$ records an occurrence of c in d. Suppose i is the last item of c and e is the event identified by (sid, tid), then we have $P_c = P(c \sqsubseteq d \cap g_i)$, where g_i is a subset of possible worlds in which the latest occurrence of item i locates in event e. And $P_i = P(i \in e)$ is the existential probability of i in e.

We transform the original sequence database into its vertical format which is a set of candidate occurrences. Figure 3 shows an example of constructing the vertical data format D_k. Here D is the original database, and D_1 is transformed from D. For example, let $s = \langle A \rangle$, then we have two groups g_1 and g_2 of occurrences of s in sequence d_1. We compute $P_{c1}(s) = 1 * P(g_1) = 0.3 * 0.5 = 0.15$ and $P_{c2}(s) = 0.5$ and save the results in D_1. Thereafter, we can compute the support probabilities $P(s \sqsubseteq d_1) = 0.65$ and $P(s \sqsubseteq d_2) = 0.4$ from D_1, which are used to calculate the frequentness probability. In this example, if we set $minsup = 1$ and $minprob = 0.5$, then $\langle A \rangle$ and $\langle B \rangle$ are two frequent patterns. 2-length candidates are generated by self-joining 1-length frequent patterns, and their occurrences are saved in D_2. For example, let $s' = \langle AB \rangle$, then $P(s' \sqsubseteq d_1) = 0.65 * 0.4 = 0.26$. Since there are two occurrences of item B in d_2, we first compute $P(g_1) = 0.8 * 0.3 = 0.24$ and $P(g_2) = 0.7$, then we have $P_{c1}(s') = 0.4 * 0.24 = 0.096$ and $P_{c2}(s') = 0.4 * 0.7 = 0.28$. Thereafter, the support probability $P(s' \sqsubseteq d_2) = 0.376$.

In our approach, we only refer to D_k in searching k-length frequent patterns. And D_k is usually in a much smaller size than the original database because it only contains occurrences of potential frequent candidate patterns.

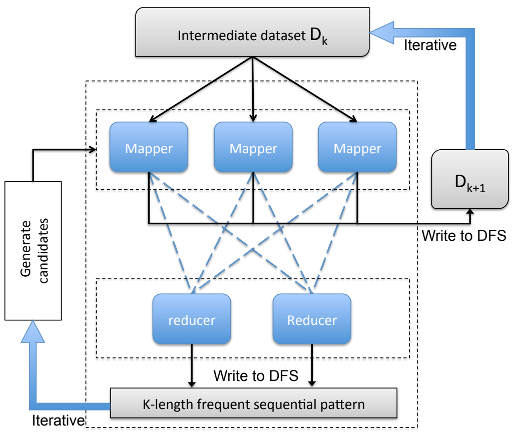


Fig. 4. Iterative MapReduce framework for uncertain sequential pattern mining

4.4 Uncertain SPM in Iterative MapReduce

Our iterative MapReduce framework helps to traverse a huge sequence tree[4] in searching frequent patterns in parallel. In each iteration, we start a MapReduce job to search k-length frequent patterns on a cluster of computers.

Fig. 4 shows our iterative MapReduce framework for uncertain SPM. In the first iteration, the original database is split and input to mappers; in the k^{th} $(k > 1)$ iteration, the input data of a mapper is a chunk of D_{k-1}. We modify the data split function in MapReduce to make sure that all occurrences in one sequence are input to the same map function. A set of k-length candidate patterns are distributed to mappers, which is denote by C_k.

(1) **Mapper function**: The mapper function is shown in Algorithm 1. It first constructs d_k from d_{k-1} and C_k, where $d_{k-1} \in D_{k-1}$ contains occurrences of $(k-1)$-length frequent patterns in one uncertain sequence. Given a candidate pattern c, the mapper computes the support probability $p = P(c \sqsubseteq d_k)$ using the newly updated data structure and outputs a key-value pair $\langle c, \langle \mu, \sigma^2 \rangle \rangle$ if $p = P(c \sqsubseteq d) > 0$. Here $\mu = p$ and $\sigma^2 = p * (1-p)$ are the mean and variance of the Bernoulli random variable $sup(c|d_k)$. Thereafter, d_k is written to distributed file system (DFS) to be used in the next iteration.

(2) **Combiner function**: We design a combiner function in Algorithm 2 to help improve the performance. Suppose a mapper function emits n key-value pairs $\langle c, \langle \mu_i, \sigma_i^2 \rangle \rangle$ $(i = 1, \ldots, n)$ which are associated with the identical pattern c. As the value filed of the mapper output is associative and commutative, they can be condensed to a single pair $\langle c, \langle \sum_1^n u_i, \sum_1^n \sigma_i^2 \rangle \rangle$. Then each mapper sends only one key-value pair to the reducer for each candidate pattern, which dramatically reduce the total bandwidth cost of data shuffling.

(3) **Reducer function**: Algorithm 3 shows the reducer function. The input key-value pair of the reducer is in the form of $\langle c, \langle \mu_i, \sigma_i^2 \rangle \rangle$, where $\mu_i = \sum p$ and $\sigma_i^2 =$

ALGORITHM 1: Map(Key *key*, Value *value*, Context *context*)

```
$d_{k-1} \leftarrow$ pase($value$)                    /* $d_{k-1} \in D_{k-1}$ parsed from value */
$C_k \leftarrow$ DistributedCache.file
$d_k \leftarrow$ construct from $C_k$ and $d_{k-1}$
foreach $c \in C_k$ do
    $p \leftarrow P(c \sqsubseteq d_k)$               /* computed by summing $P_c(c)$ in $d_k$ */
    $key \leftarrow c$;
    $value \leftarrow \langle p, p * (1-p) \rangle$   /* composited value */
    context.collect($key$, $value$)
end
DFS.file $f$ = new DFSFile("$D_k$");
$f$.append($d$);
```

ALGORITHM 2: Combine(Key *key*, Iterable *values*, Context *context*)

```
$\mu \leftarrow 0$, $\sigma^2 \leftarrow 0$
foreach $value \in values$ do
    $\mu = \mu + value.\mu$
    $\sigma^2 = \sigma^2 + value.\sigma^2$
end
$context.collect(key, \langle \mu, \sigma^2 \rangle)$
```

$\sum p * (1-p)$ are the partially aggregated mean and variance of the probabilistic support of candidate c. The reducer function accumulates the overall mean and variance of c in the entire uncertain database and uses the Gaussian distribution to approximate the distribution of overall support $sup(c)$. Given $minsup = t_s$ and $minprob = t_p$, the reducer outputs the probabilistic frequent sequential patterns to the file, if $P(sup(c) \geq t_s) \geq t_p$; otherwise, c is not probabilistic frequent and is discarded by the reducer.

A MapReduce iteration is finished after all k-length probabilistic frequent sequential patterns are discovered and written to DFS files. After that, we self-join k-length frequent patterns to generate $(k+1)$-length candidate patterns for the next iteration. This process continues until all frequent patterns are discovered.

5 Evaluation

In this section, we implement our uncertain SPM algorithm in iterative MapReduce, denoted by *IMRSPM*, and evaluate its performance using both synthetic and real world datasets in a 10-node Hadoop cluster.

A naive method directly enumerates possible worlds table without reusing previous computational results. We implement this naive approach in Iterative MapReduce as *baseline*, which is denoted by *BL* here. We also compare our algorithm with the single-machine uncertain sequential pattern mining algorithm, denoted by *UPrefix* [17, 18], to show the benefit from parallel computing.

ALGORITHM 3: Reduce(Key *key*, Iterable *values*, Context *context*)

```
$c \leftarrow key$
$\mu \leftarrow 0$, $\sigma^2 \leftarrow 0$
**foreach** $value \in values$ **do**
    $\mu = \mu + value.\mu$
    $\sigma^2 = \sigma^2 + value.\sigma^2$
**end**
$sup(c) \sim N(\mu, \sigma^2)$
$t_s \leftarrow context.minsup$, $t_p \leftarrow context.minprob$
**if** $P(sup(c) \geq t_s) \geq t_p$ **then**
    DFS.file f = new DFSFile("frequent pattern");
    $f$.append($c$);
**end**
```

5.1 Synthetic Dataset Generation

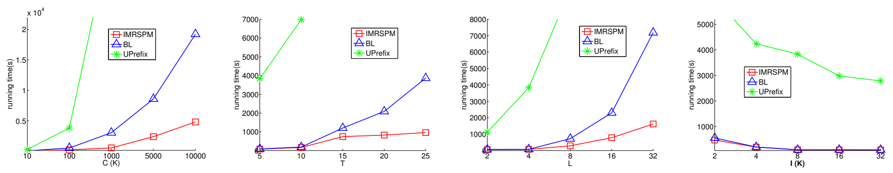
The IBM market-basket data generator [3] uses the following parameters to generate sequence datasets in various scales: (1) C: number of customers; (2) T: average number of transactions per sequence; (3) L: average number of items per transaction per sequence; (4) I: number of different items.

We assume that an event existential probability follows normal distribution $t \sim N(\mu, \sigma^2)$, where μ is randomly drawn from range $[0.7, 0.9]$ and σ is randomly drawn from range $[1/21, 1/12]$. Then we draw a value from t and assign it to an event in the original synthetic datasets as its existential probability. This approach has been used in previous work [1] to generate synthetic uncertain datasets. We name a synthetic uncertain dataset by its parameters. For example, a dataset T4L10I10C10 indicates $T = 4$, $L = 10$, $I = 10*1000$ and $C = 10*1000$.

5.2 Scalability

In Figure 5, we evaluate the scalability of IMRSPM on synthetic datasets generated by different parameters. Here we set $minsup = 0.2\%$ and $minprob = 0.7$. Fig. 5(a) shows the running time variations of IMRSPM when C varies from 10 000 to 10 000 000, where $T = 4$, $L = 4$, $I = 10\,000$. Fig. 5(b) shows the running time variations of IMRSPM when T varies from 5 to 25, where $C = 100\,000$, $L = 4$, $I = 10\,000$. Fig. 5(c) shows the running time variations of IMRSPM when L varies from 2 to 32, where $C = 100\,000$, $T = 4$, $I = 10\,000$. Fig. 5(d) shows the running time variations of IMRSPM when I varies from 2 000 to 32 000, where $C = 100\,000$, $T = 4$, $L = 4$.

In Figure 5, we observe the following phenomenons:
(1) IMRSPM outperforms BL under every setting of the parameters, which proves the effectiveness of our incremental temporal uncertainty management approach; meanwhile, IMRSPM is much more scalable than UPrefix, which demonstrates the advantage of using iterative MapReduce framework.
(2) The running time increase with the increment of C, T, L, as increasing these



(a) scale of C (b) scale of T (c) scale of L (d) scale of I

Fig. 5. Scalability of IMRSPM-A algorithm

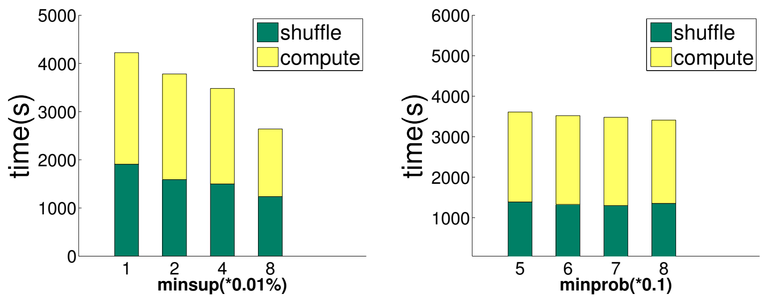
parameters generates larger scale datasets. Furthermore, when T or L are set to larger values, there are more repeated items in uncertain sequences. And our incremental uncertainty management approach shows its effectiveness in improving the efficiency especially in such cases.
(3) The running time slightly drops with the increment of I. When the value of I grows, the number of repeated item in one sequence become less because items are randomly selected from a fixed set of items.

5.3 Mining Customer Behavior Patterns from Amazon Reviews

We apply our IMRSPM algorithm in Amazon review dataset[12] to discover customer behavior patterns. The Amazon review dataset includes 34 686 770 reviews of 2 441 053 products from 6 643 669 customers between June 1995 to March 2013. Each review is scored by an integer between 1 to 5, which indicates a user opinion toward a product. However, this score is a lose measurement of subjective satisfaction. Suppose a customer gives a score t to a product, then we believe that the probability that this customer likes this product is $p = t/5$. An ordered list of user reviews is regarded as an uncertain sequence. A probabilistic frequent sequential pattern $\langle A, B\rangle$ mined from this database can be explained as: if a customer likes product A, then it is very likely that he/she will like product B in the future.

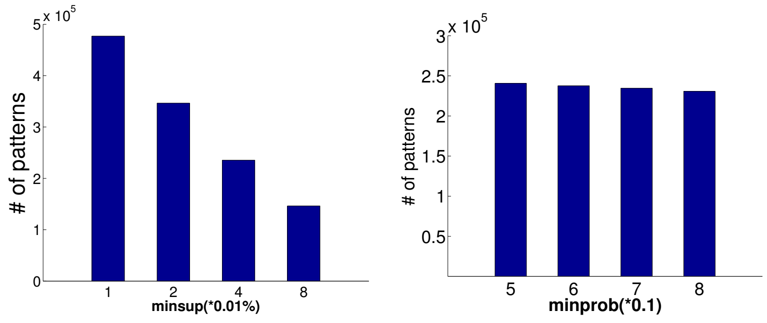
For example, given $minsup = 0.005\%$ and $minprob = 0.7$, we have discovered the sequential pattern $\langle$B000TZ19TC $\rightarrow$ B000GL8UMI$\rangle$. Here B000TZ19TC is the Amazon Standard Identification Number (ASIN) of the book *Fahrenheit 451* published in 1953. And this pattern reveals that users who now like product B000TZ19TC may also like B000GL8UMI in the future, which is a newer edition of the same book published in 1963. We also discover other non-trivial patterns as $\langle$B000MZWXNA $\rightarrow$ B000PBZH6Q$\rangle$, where B000MZWXNA is associated with the book *The Martian Way* and ASIN B000PBZH6Q identifies the book *Foundation*.

Figure 6 and Figure 7 show the effect of user-defined parameters $minsup$ and $minprob$ in Amazon dataset. We initially set $minprob = 0.7$ and $minsup = 0.04\%$. In Figure 6(a) and 7(a), we vary the value of $minsup$ from 0.02% to 0.04%; while $minprob$ is varied from 0.5 to 0.8 in Figure 6(b) and 7(b). From Figure 6 and Figure 7 , we observe that: (1) In Figure 6(a), the running time



(a) vary *minsup* (b) vary *minprob*

Fig. 6. Effect of user-define parameters in efficiency



(a) vary *minsup* (b) vary *minprob*

Fig. 7. Effect of user-define parameters in number of patterns

of IMRSPM decreases with the increment of *minsup*; meanwhile, the effect of *minsup* to the computing time is more significantly than that to the shuffling time. The reason is that fewer frequent patterns are mined when *minsup* is larger, which can be proved by Figure 7(a).
(2) The performance remains relatively stable to the variation of *minprob*. The probabilistic support of a sequential pattern is bounded to its expected value (*Chernoff bound*). Thus, the frequentness of a large number of candidate patterns becomes deterministic, and this explains why the running time and the number of frequent patterns do not significantly fluctuate in Figure 6(b) and 7(b).

6 Conclusions

In this paper, we propose a SPM algorithm in an iterative MapReduce framework for large scale uncertain databases to discover customer behavior patterns in Amazon review dataset. In the future, we will continue to explore the facilitation of other distributed platforms in solving uncertain SPM problems.

References

1. C. C. Aggarwal, Y. Li, J. Wang, and J. Wang. Frequent pattern mining with uncertain data. In *SIGKDD*, pages 29–38, 2009.
2. C. C. Aggarwal and P. S. Yu. A survey of uncertain data algorithms and applications. *IEEE Trans. on Knowl. and Data Eng.*, 21(5):609–623, May 2009.
3. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, pages 487–499, 1994.
4. J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation. In *SIGKDD*, pages 429–435, 2002.
5. T. Bernecker, H.-P. Kriegel, M. Renz, F. Verhein, and A. Zuefle. Probabilistic frequent itemset mining in uncertain databases. In *SIGKDD*, pages 119–128. ACM, 2009.
6. H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. In *Annals of Mathematical Statistics*, volume 23, pages 493–507, 1952.
7. M.-S. C. Chun-Chieh Chen, CHi-Yao Tseng. Highly scalable sequential pattern mining based on mapreduce model on the cloud. In *BigData Congress*, pages 310–317, 2013.
8. B.-S. Jeong, H.-J. Choi, M. A. Hossain, M. M. Rashid, and M. R. Karim. A mapreduce framework for mining maximal contiguous frequent patterns in large dna sequence datasets. In *IETE Technical Review*, volume 29, pages 162–168, 2012.
9. J. Jestes, G. Cormode, F. Li, and K. Yi. Semantics of ranking queries for probabilistic data. *IEEE Transactions on Knowledge and Data Engineering*, 23(12):1903–1917, 2011.
10. N. Khoussainova, M. Balazinska, and D. Suciu. Probabilistic event extraction from rfid data. In *In Proceedings of the 24th IEEE International Conference on Data Engineering*, pages 1480–1482, 2008.
11. Y. Li, J. Bailey, L. Kulik, and J. Pei. Mining probabilistic frequent spatio-temporal sequential patterns with gap constraints from uncertain databases. In *IEEE International Conference on Data Mining*, pages 448–457, 2013.
12. J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, 2013.
13. I. Miliaraki, K. Berberich, R. Gemulla, and S. Zoupanos. Mind the gap: Large-scale frequent sequence mining. In *SIGKDD*, pages 797–808, 2013.
14. M. Muzammal and R. Raman. Mining sequential patterns from probabilistic databases. In *PAKDD*, pages 210–221, 2011.
15. Y. Tong, L. Chen, Y. Cheng, and P. S. Yu. Mining frequent itemsets over uncertain databases. In *Proceeding of the VLDB Endowment*, volume 5, pages 1650–1661, 2012.
16. L. Wan, L. Chen, and C. Zhang. Mining frequent serial episodes over uncertain sequence data. In *EDBT*, pages 215–226, 2013.
17. Z. Zhao, D. Yan, and W. Ng. Mining probabilistically frequent sequential patterns in uncertain databases. In *EDBT*, pages 74–85, 2012.
18. Z. Zhao, D. Yan, and W. Ng. Mining probabilistically frequent sequential patterns in large uncertain databases. In *IEEE Transactions on Knowledge and Data Engineering*, volume 26, pages 1171–1184, 2013.