2012 IEEE 13th International Conference on Mobile Data Management

BP-Growth: Searching Strategies for Efficient Behavior Pattern Mining

Xueying Li[1], Huanhuan Cao[2], Enhong Chen[1], Hui Xiong[3], Jilei Tian[2]
[1]School of Computer Science and Technology, University of Science and Technology of China
lxying@mail.ustc.edu.cn, cheneh@ustc.edu.cn
[2]Nokia Research Center Beijing, happia.cao@nokia.com, jilei.tian@nokia.com
[3]Rutgers Business School, Rutgers University, USA, hxiong@rutgers.edu

***Abstract*—User habit mining plays an important role in user understanding, which is critical for improving a wide range of personalized intelligence services. Recently, some researchers proposed to mine user *behavior patterns* which characterize the habits of mobile users and account for the associations between user interactions and context captured by mobile devices. However, the existing approaches for mining these behavior patterns are not practical in mobile environments due to limited computing resources on mobile devices. To fulfill this crucial void, we investigate optimizing strategies which can be used for improving the efficiency of behavior pattern mining in terms of computing and memory needs. Specifically, we examine typical optimizing strategies for association rule mining and study the feasibility of applying them to behavior pattern mining, since these two problems are similar in many aspects. Moreover, we develop an efficient algorithm, named BP-Growth, for behavior pattern mining by combining two promising strategies. Finally, experimental results show that BP-Growth outperforms benchmark methods with a significant margin in terms of both computing and memory cost.**

***Keywords*-behavior pattern mining; optimizing strategies;**

I. Introduction

In recent years, the development of mobile devices progressed at an ever fastest pace to make possible supporting various applications and services beyond the traditional speech-centric service, such as music, videos, web browsing, gaming and camera shooting, just to name a few. The rich user interaction information captured by the mobile device can be used to understand user habits, which can bring a great business value, such as data-driven user studies for marketing, targeted advertising and personalized recommendation. Consequently, studying the habits of mobile users through their interactions attracts many researchers' attention, e.g., [1–3].

A distinct property of the user interactions with mobile devices is that they are usually associated with volatile contexts, such as waiting a bus, driving a car, or doing shopping. Intuitively, some user interactions are context-aware, that is, the occurrences of these user interactions are influenced by the contexts of users. For example, some users would like to listen to music with their smart phones when taking a bus to the workplace but rarely do the same thing on other contexts. The context-awareness of user interactions reflects the habits of mobile users. Therefore, Cao et al. [3] proposed to characterize user habits by the associations between user interaction records and the corresponding contexts.

This kind of associations is referred to as *behavior patterns*. A wide range of context-aware services, such as context-aware recommendation, context-aware UI adaption [4], can be improved by understanding the habits of mobile users from their behavior patterns. Moreover, behavior patterns can be used to build user profiles and segment users for marketing analysis.

Context logs collect the history context data and interaction records of mobile users, and thus can be used as data sources for mining behavior patterns. However, mining behavior patterns is not a trivial problem because it can not be addressed by the traditional association rule mining [5–7]. On one hand, we should not collect context data only when an interaction happens, because it loses the discriminative information on how likely no interaction happens with a given context. On the other hand, if we always collect context data no matter whether any interaction happens, the occurrences of contexts and user interaction records in context logs will be very unbalanced [3], which makes it difficult to mine meaningful patterns through the traditional association rule mining approach. Therefore, Cao et al. [3] defined the problem of behavior pattern mining from a different perspective which takes context logs as time ordered sequences of context records and calculates the support of a context by taking into account its time ranges of appearances. Moreover, they proposed an effective algorithm named GCPM to solve the problem.

To protect user privacy and save the data flow, it may be desirable to directly perform behavior pattern mining on users' mobile devices instead of transiting the raw context data to the back end server and then performing mining. However, we find it is difficult to apply GCPM in practical mobile computing environment due to the limited computing resource of mobile devices, which motivates us to search effective strategies for improving the efficiency of behavior pattern mining. To this end, we investigate several typical optimizing strategies for association rule mining and discuss the feasibility of applying them to behavior pattern mining since the two problems are similar in some aspects.

The contributions of this paper are summarized as follows.

First, to the best of our knowledge, it is the first attempt to systematically study the feasibility of applying typical optimizing strategies used in association rule mining to behavior pattern mining.

978-0-7695-4713-8/12 $26.00 © 2012 IEEE
DOI 10.1109/MDM.2012.14

Second, we propose a novel and efficient algorithm named BP-Growth (**B**ehavior **P**attern **G**rowth) for mining behavior patterns. BP-Growth combines two optimizing strategies for association rule mining and the experimental results on real context data clearly show that it significantly outperforms GCPM and other two baselines in terms of both running time and memory cost.

Last, we firstly make a systematic evaluation of several algorithms for behavior pattern mining in a practical mobile computing environment, which may be inspiring for the developers who are interested in developing relevant applications based on behavior pattern mining.

The rest of this paper is organized as follows. First, in Section 2, we review the problem statement of behavior pattern mining. Then, we briefly review some related work and especially introduce the GCPM algorithm in Section 3. In Section 4, we study the feasibility of applying several widely used optimizing strategies for association rule mining to behavior pattern mining. In Section 5, we propose a novel and efficient algorithm named BP-Growth for mining behavior patterns by combing two promising strategies. Next in Section 6, the experimental results in practical mobile computing environment are summarized and discussed. Finally, in Section 7 we conclude this paper and point out the direction of future work.

II. Preliminaries

Before reviewing the related work, we firstly briefly review the related notions of association rule mining [6–8] and behavior pattern mining [3, 9] as follows.

A. Association Rule Mining

A transaction database TDB is a set of transactions, where each transaction, denoted as a tuple $< Tid, X >$, contains a set of items (i.e., X) and is associated with a unique transaction identifier Tid. A transaction $< Tid, X >$ is said to contain itemset Y if $Y \subset X$. The number of transactions in TDB containing itemset Y is called the support of itemset Y, denoted as $Sup(Y)$. Given a minimum support threshold min_sup and a minimum confidence threshold min_conf, $A \Longrightarrow B$ is an association rule if $Sup(A \cup B) \geq min\_sup$ and $\frac{Sup(A \cup B)}{Sup(A)} \geq min\_conf$, where A, B denote two non-overlapped itemsets, A is called the antecedent, and B is called the consequent.

B. Behavior Pattern Mining

The problem of behavior pattern mining is proposed by Cao et al. [3] for mining the habits of mobile users from their context logs which record their historical context data and interaction records. A context log R contains several *context records* $r_1 r_2 ... r_n$, and each context record $r =< Tid, C, I >$ consists of a timestamp Tid, the most detailed available context at that time C, and the corresponding user interaction record I. A context C consists of several contextual feature-value pairs $\{(f_1 : v_1), (f_2 : v_2), ..., (f_l : v_l)\}$ where $(f_i : v_i)$ denotes a context data type f_i and the corresponding value v_i, such as (*Time range: AM8:00-9:00*). It is worth noting that we mention "available" because a context record may miss some context data though which context data should be collected is usually predefined. For example, the GPS coordinate is not available when the user stays indoors. Moreover, interaction records can be empty (denoted as "Null") because user interactions do not always happen. Table I illustrates a toy context log which contains 10 context records.

According to [3], for the associations between user contexts and the user interactions with mobile devices, they are regarded as behavior patterns if both their supports and confidences are bigger than predefined thresholds. Compared with the traditional association rule mining, behavior pattern mining can address the unbalanced occurrences of context data and user interaction records well due to its different way of calculating the supports of contexts. To be specific, it counts the support of a context by taking into account how many times it continuously appears in several adjacent context records, which constitute a *context range* of the context. For a context range which contains non-empty interaction records, the number of non-empty interaction records is regarded as the support of the corresponding context in this context range. Otherwise, for a context range which only contains empty interaction records, the support of the corresponding context is regarded as one in this context range. Finally, the support of the corresponding context is calculated by summing the supports in each of its context ranges. Take the context log in Table I for example, considering the context "{*(Is a holiday?: No)*,(*Time range: AM8:00-9:00*)}", though it appears in seven context records, its support is not seven but three because it has three context ranges, i.e., (t_1, t_2, t_3), (t_{38}, t_{39}), and (t_{58}, t_{59}), respectively, and each context range at most has one non-empty interaction record. Moreover, since {(*Is a holiday?: No*),(*Time range: AM8:00-9:00*)} and "*Playing Music Player*" co-occur two times, the confidence of "{(*Is a holiday?: No*),(*Time range: AM8:00-9:00*)} $\longrightarrow$ *Playing Music Player*" is 2/3 = 0.66. According to [3], the definition of behavior patterns is formulated as follows.

Definition 1 (Support, Confidence): Given a context C_i and an interaction record I, the **support** of C_i w.r.t. I (denoted as $Sup(C_i \Longrightarrow I)$) is $\sum_m Count_m(I)$, where $Count_m(I)$ denotes the occurrence number of I in the m-th context range of C_i.

Given a complete interaction set Γ, the **support** of C_i (denoted as $Sup(C_i)$) is $\sum_{I \in \Gamma} Sup(C_i \Longrightarrow I) + N_0$, where N_0 denotes the number of C_i's context ranges which only contain empty interaction records. Moreover, the **confidence** of C_i w.r.t. I denoted as $(Conf(C_i \Longrightarrow I))$ is $\frac{Sup(C_i \Longrightarrow I)}{Sup(C_i)}$.

Definition 2 (Promising Context, Behavior Pattern): Given a context C_i, if $\exists_I Sup(C_i \Longrightarrow I) \geq min\_sup$, C_i is called a **promising context**. Moreover, if $Sup(C_i \Longrightarrow I) \geq min\_sup$ and $Conf(C_i \Longrightarrow I) \geq min\_conf$, $C_i \Longrightarrow I$ is called a **behavior pattern**.

TABLE I
A TOY CONTEXT LOG.

Timestamp	Context	Interaction record
t_1	{(Is a holiday?: No),(Time range: AM8:00-9:00),(Cell ID: 2341-42344)}	Null
t_2	{(Is a holiday?: No),(Time range: AM8:00-9:00),(Cell ID: 2341-22347)}	Playing Music Player
t_3	{(Is a holiday?: No),(Time range: AM8:00-9:00),(Cell ID: 2341-79901)}	Null
.....		
t_{38}	{(Is a holiday?: No),(Time range: AM8:00-9:00),(Cell ID: 2341-32044)}	Null
t_{39}	{(Is a holiday?: No),(Time range: AM8:00-9:00),(Cell ID: 2341-2501)}	Null
.....		
t_{58}	{(Is a holiday?: No),(Time range: AM8:00-9:00),(Cell ID: 2341-42344)}	Playing Music Player
t_{59}	{(Is a holiday?: No),(Time range: AM8:00-9:00),(Cell ID: 2341-42344)}	Null

III. RELATED WORK

Cao et al. [3] firstly proposed to use behavior patterns to characterize mobile users' habits and gave a corresponding algorithm named GCPM for behavior pattern mining. As introduced in the previous section, behavior patterns denote associations between user interactions and contexts captured by mobile devices. However, though traditional association rule mining is widely used for mining associations between items that usually co-occurred in same transactions from a transaction data base [6–8], it can not be used to mine behavior patterns because of the unbalanced occurrences of contexts and interaction records [3]. Therefore, Cao et al. defined the problem of behavior pattern mining by using different metrics of *support* and *confidence*, which are two key notions of association.

GCPM is an Apriori-like algorithm [8] which iteratively generates candidate behavior patterns by joining shorter promising contexts and checks their correctness from the original context log. Though it can effectively find meaningful behavior patterns, we find it is still difficult to apply it to a practical mobile computing environment since it has a relatively high requirement of main memory and big computation cost. To this end, we are motivated to find a much more efficient algorithm for behavior pattern mining which fits limited computation resource.

The problem of association rule mining has been studied for more than a decade and a lot of optimizing strategies are proposed to improve the mining efficiency. These strategies can be roughly grouped into four categories. The first category is to reduce the redundant data before performing mining. The second category is to compress the original data set by FP-Trees. The third category is to adopt "divide and conquer" principle and partition the original data sets into several small data sets. The last category is to discover patterns by directly comparing candidate patterns in the sorted data set. Most of successful algorithms take advantage of more than one optimizing strategy. For example, FP-Growth [10] and CLOSET+ [11] take advantage of the strategies of removing redundant data, FP-Trees, and data set partition. FreeSpan [7] and PrefixSpan [12] take advantage of the strategies of removing redundant data and data set partition. Disc [13] takes advantage of the strategies of removing redundant data, data set partition, and directly comparing candidate patterns.

Since the problem of behavior pattern is largely related to association rule mining, the optimizing strategies for association rule mining may be used for improving the efficiency of behavior pattern mining. It motivates us to systematically study these strategies and discuss the feasibility of applying them to behavior pattern mining.

IV. CANDIDATE OPTIMIZING STRATEGIES FOR BEHAVIOR PATTERN MINING

A. Removing Redundant Data

In the problem of traditional association rule mining, an item is infrequent if and only if its support is less than min_sup. A widely used optimizing strategy is removing the infrequent items of the original database before performing association rule mining. It is easy to prove that removing infrequent items will not affect the mined association rules according to the Apriori-Theory [8]. In this way, both the memory requirement and mining space of association rule mining are reduced.

Similarly, in the problem of behavior pattern mining, we can safely remove the *unpromising* contextual feature-value pairs from the context log. The support of a contextual feature-value pair is calculated by taking it as a 1-context. To prove this statement, we firstly introduce the following lemmas.

Lemma 4.1: Given a promising context C_i, for any unpromising contextual feature-value pair p, we can conclude $p \notin C_i$.

Proof: Assume that p is unpromising and $p \in C_i$. According to Definition 2, $\exists_I Sup(C_i \Longrightarrow I) \geq min\_sup$. Moreover, according to the Context-Apriori theorem [3], $\exists_I Sup(p \Longrightarrow I) \geq min\_sup$. However, the conclusion is in conflict with the statement that p is unpromising. Therefore, the assumption is false and the lemma is true. ■

Lemma 4.2: Given a context C_i and a context log R, removing a contextual feature-value pair p from R where $p \notin C_i$ does not affect $Sup(C_i)$ and $\forall_I Sup(C_i \Longrightarrow I)$.

Proof: First, given a context C_i, for each context record in R denoted as $r =< Tid, C, I >$, removing p does not affect whether C_i appears in C . Therefore, removing p does not affect the context ranges of C_i. Considering that the interaction records are not affected by removing p too, it is obvious that removing p does not affect N_0 and $\forall_I Sup(C_i \Longrightarrow I)$ according to Definition 1. Moreover, since $Sup(C_i) = \sum_{I \in \Gamma} Sup(C_i \Longrightarrow I) + N_0$, we can conclude that removing p does not affect $Sup(C_i)$. ■

The above lemmas imply that for any promising context C_i, removing unpromising contextual feature-value pairs does not affect $\forall_I Sup(C_i \Longrightarrow I)$ and $Sup(C_i)$. Therefore, we can safely remove unpromising contextual feature-value pairs for

TABLE II
AN EXAMPLE OF CONTEXT LOG

Tid	Context	Interaction
t_1	$\{(a:2),(b:3),(c:5)\}$	Null
t_2	$\{(a:2),(b:3),(c:5),(d:1)\}$	I_1
t_3	$\{(a:2),(b:3),(c:5),(d:1)\}$	Null
t_4	$\{(a:3),(b:2),(c:5),(d:0),(e:1)\}$	I_2
t_5	$\{(a:2),(b:2),(c:5),(d:1)\}$	Null
t_6	$\{(a:2),(b:3),(c:5),(d:0),(e:1)\}$	Null
t_7	$\{(a:2),(b:3),(c:5),(d:1)\}$	Null
t_8	$\{(a:2),(b:3),(c:5),(d:1)\}$	Null
t_9	$\{(a:2),(b:3),(c:4),(d:1)\}$	I_2
t_{10}	$\{(a:2),(b:3),(c:5)\}$	I_1

reducing the size of original context log and thus the mining space.

In the problem of behavior pattern mining, the size of the original context log can be further safely reduced by removing the adjacent duplicate context records which only contain empty interaction records according to the following theorems.

Theorem 1: Given a context log $R = r_1 r_2 ... r_n$, a context C_i, if $\exists_{1 \le s < m \le n} r_s.C = r_{s+1}.C = ... = r_m.C$ where $r.C$ denotes the context of r and $\forall_{s \le k \le m, I} I \notin r_k$, removing $r_{s+1}...r_m$ does not affect $Sup(C_i)$ and $\forall_I Sup(C_i \Longrightarrow I)$.

Proof: Because no context record of $r_{s+1}...r_m$ contains a non-empty interaction record, removing $r_{s+1}...r_m$ does not affect $\forall_I Sup(C_i \Longrightarrow I)$ according to Definition 1. We only need to prove removing $r_{s+1}...r_m$ does not affect N_0, i.e., the number of C_i's context ranges which only contain empty interaction records.

Firstly, suppose C_i is a sub-set of $r_k.C$. Thus, $\forall_{s \le k \le m} C_i \subseteq r_k.C$ because $r_s.C = r_{s+1}.C = ... = r_m.C$. According to the definition of context range, $r_{s+1}...r_m$ must be in a context range of C_i. In this case, removing $r_{s+1}...r_m$ just modifies the context range but does not remove it. Thus, N_0 won't be affected.

Secondly, suppose C_i is not a sub-set of $r_k.C$. Thus, $\forall_{s \le k \le m} C_i \not\subseteq r_k.C$ because $r_s.C = r_{s+1}.C = ... = r_m.C$. According to the definition of context range, $r_{s+1}...r_m$ must not be in any context range of C_i. In this case, removing $r_{s+1}...r_m$ does not affect any context range of C_i can thus does not affect N_0. ■

Similarly, we have the following theorem.

Theorem 2: Given a context log $R = r_1 r_2 ... r_n$, a context C_i, if $\exists_{1 \le s < m \le n} r_s.C = r_{s+1}.C = ... = r_m.C$ and $\exists_{s \le k \le m, I} I \in r_k$, removing r_k $(s \le k \le m)$ does not affect $Sup(C_i)$ and $\forall_I Sup(C_i \Longrightarrow I)$ if r_k only contains an empty interaction record.

B. Data Compaction through FP-Trees

FP-Tree [10] is also widely used for reducing the size of the original transaction database in association rule mining. FP-Tree is a prefix tree of the lists of frequent items of the original transaction database. A FP-Tree is usually much smaller than the original transaction database because transactions usually share the same prefix. The following figure gives an example of FP-Tree for the context log in Table II by taking contextual feature-value pairs as items. It is worth noting that each node is associated with a support vector for recording the supports of the context derived from the path to the root node w.r.t. each

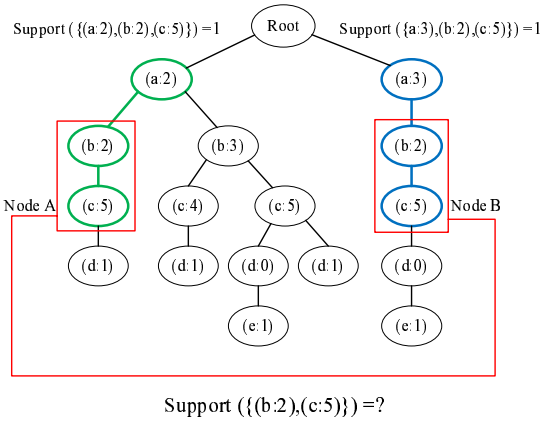

Fig. 1. The FP-Tree for the context log in Table II.

interaction record and the corresponding N_0. For example, Node A contains a support vector for recording the supports of $\{(a:3),(b:2),(c:5)\}$ w.r.t. I_1, I_2, and N_0. In contrast, in association rule mining, each node of FP-Tree only records the frequency that the item set derived from the path to the root node occurs as prefixes of transactions [10].

However, we find that FP-Tree cannot be applied to behavior pattern mining because the problem of behavior pattern mining has not an important property that association rule mining has. To ease introducing the property, we introduce the following notion firstly.

Definition 3 (Minimum Coverage Path): Given an item set α, a FP-Tree T, if 1) N is a node of T where the item set derived from the path from the root node of T to N is a super set of α; and 2) there exists no a path from the root node of T to an ancestor node of N which can also derive a super set of α, the path between N and the root of T is called a **minimum coverage path** of α.

Property 1 (Minimum Coverage Path Completeness): In the problem of association rule mining, given an item set α, a FP-Tree T, we have $Sup(\alpha) = \sum_N N.freq$, where the path between N and the root of T makes a minimum covering path of α, and $N.freq$ indicates the frequency that the item set derived from the corresponding minimum covering path occurs as prefixes of original transactions.

This property is very important for the success of FP-Tree to maintain all necessary information for association rule mining. For example, if we take the context log in Table II as a transaction database and consider the FP-Tree in Figure 1 in the view of association rule mining, we can calculate the frequency of $\{(b:2),(c:5)\}$ by summing the frequencies of $\{(a:2),(b:2),(c:5)\}$ and $\{(a:3)(b:2),(c:5)\}$, which are recorded by node A and node B, respectively.

By contrast, in the problem of behavior pattern mining, we can not calculate the support of a context in a similar way. Take the FP-Tree in Figure 1 for example, given a context $C_i = \{(b:2),(c:5)\}$, node A and node B contribute two minimum covering pathes of C_i which derive $\{(a:2),(b:2),(c:5)\}$ (denoted as C_j) and $\{(a:3),(b:2),(c:5)\}$ (denoted as C_k), respectively. As mentioned

above, node A records $N_0 = 1$, $Sup(C_j \Longrightarrow I_1) = 0$, $Sup(C_j \Longrightarrow I_2) = 0$, and node B records $N_0 = 0$, $Sup(C_k \Longrightarrow I_1) = 0$, $Sup(C_k \Longrightarrow I_2) = 1$. Thus, both $Sup(C_j)$ and $Sup(C_k)$ are 1. However, we can not calculate $Sup(C_i)$ since $Sup(C_i) = 1 \neq [Sup(C_j) + Sup(C_k)]$. The challenge of applying FP-Tree to behavior pattern mining is that FP-Tree can not maintain all necessary information for mining behavior patterns, which is caused by the different way of calculating supports.

C. Data Set Partition

The major computation cost of Apriori-like algorithms comes from the generation of candidate patterns. In association rule mining, the strategy of data set partition is usually used to avoid candidate pattern generation. For example, FP-Growth [10] and CLOSET+ [11] recursively generate candidate FP-Trees from the original FP-Tree. The main idea of data set partition is to recursively partition the original data set into smaller sub-datasets and then mine patterns on the basis of the seed patterns of each sub-data set.

Since in behavior pattern mining we can not use a FP-Tree to compress the original context log, we should study the association rule mining algorithms which take advantage of data set partition strategy but do not use FP-Trees. The typical algorithms of them are PrefixSpan [12] and PTAC [14]. Instead of performing FP-Tree based partition, they take advantage of an alternative data partition approach named *Prefix Projection*. The main idea of prefix projection is to recursively partition the original data set with prefixes as projected data sets and make the new mined patterns from the projected data sets as new prefixes. To depict how to apply prefix projection to behavior pattern mining, let's introduce the following related notions.

Definition 4 (Prefix, Projection, and Suffix): Given a context $C_i = \{(x_1 : v_1), (x_2 : v_2), ..., (x_l : v_l)\}$, one of its sub-contexts $C_j = \{(y_1 : u_1), (y_2 : u_2), ..., (y_m : u_m)\}$ is called a **prefix** of C_i if and only if $\forall_{1 \leq k \leq m}(x_k = y_k) \wedge (v_k = u_k)$; Given a context record $r =< Tid, C_i, I >$ and one context $C_j \subseteq C_i$, the **projection** of r w.r.t. prefix C_j is $rp =< Tid, C_p, I >$, where C_p is the longest sub-context of C_i which has a prefix C_j. Moreover, the **suffix** of r w.r.t. prefix C_i is $rs =< Tid, C_s, I >$, where $C_s = C_p - C_j$;

For example, given a context record $r =< t_2, \{(a : 2), (b : 3), (c : 4), (d : 1)\}, I_2 >$ and a context $(b : 3)$, the projection of r w.r.t. prefix $\{(b : 3)\}$ is $rp =< t_2, \{(b : 3), (c : 4), (d : 1)\}, I_2 >$ and the suffix of r w.r.t. prefix $\{(b : 3)\}$ is $rs =< t_2, \{(c : 4), (d : 1)\}, I_2 >$.

Definition 5 (Projected Context Log): Given a context log R and a context C_j, the **projected context log** of R w.r.t. prefix C_j is a timestamp ordered suffix sequence $R|C_j$ which contains all suffixes of context records in R w.r.t. prefix C_j.

For example, Table III shows the projected context log of the context log in Table II w.r.t. prefix $\{(b : 3)\}$.

The suffix of a context record can be taken as a special context record. The related notions about context record also fit suffix. Similarly, the related notions about context log, such as context range, support, and confidence, also fit projected context log. Because projected context logs are usually dramatically smaller than the original context log, the search space of mining behavior patterns will be reduced if we can mine behavior patterns from the projected context logs.

TABLE III
THE PROJECTED CONTEXT LOG OF THE CONTEXT LOG IN TABLE II W.R.T. PREFIX $\{(b : 3)\}$

Tid	Context	Interaction
t_1	$\{(c : 5)\}$	Null
t_2	$\{(c : 5), (d : 1)\}$	I_1
t_3	$\{(c : 5), (d : 1)\}$	Null
t_6	$\{(c : 5), (d : 0), (e; 1)\}$	Null
t_7	$\{(c : 5), (d : 1)\}$	Null
t_8	$\{(c : 5), (d : 1)\}$	Null
t_9	$\{(c : 4), (d : 1)\}$	I_2
t_{10}	$\{(c : 5))\}$	I_1

Since the factors of indicating a behavior pattern $C_i \Longrightarrow I$ are $Sup(C_i \Longrightarrow I)$ and $Sup(C_i)$, if we can calculate the two factors from projected context logs, we will be able to mine behavior patterns from projected context logs.

Theorem 3: Given a context $C_j = \{(x_1 : v_1), (x_2 : v_2), ..., (x_l : v_l)\}$ and one of its super contexts $C_i = \{(x_1 : v_1), (x_2 : v_2), ..., (x_l : v_l), (x_{l+1} : v_{l+1})\}$, $\forall_I Sup(C_i \Longrightarrow I) = Sup|C_j((x_{l+1} : v_{l+1}) \Longrightarrow I)$, where $Sup|C_j(*)$ indicates the support of $*$ in $R|C_j$.

This theorem is easy to prove. It implies that given a $l+1$-context C_i, denoting the last context feature-value pair of C_i as $(x_{l+1} : v_{l+1})$, we can calculate $Sup(C_i \Longrightarrow I)$ by calculating $Sup|C_j((x_{l+1} : v_{l+1}) \Longrightarrow I)$, where C_j denotes the context generated by removing $(x_{l+1} : v_{l+1})$ from C_i.

However, we cannot calculate $Sup(C_i)$ in a similar way. Take the projected context log in Table III for example, $Sup(\{(b : 3), (c : 5)\}) = 3 \neq Sup|C_j((c : 5)) = 2$ where $C_j = (b : 3)$. It is because the projected context log may lose the information of context range contained by the original context log. Fortunately, we can solve this problem by extracting the context ranges of the prefix C_j from the original context log firstly and then build the projected context logs w.r.t. prefix C_j in each context range of C_j.

Theorem 4: Given a context $C_j = \{(x_1 : v_1), (x_2 : v_2), ..., (x_l : v_l)\}$ and one of its super contexts $C_i = \{(x_1 : v_1), (x_2 : v_2), ..., (x_l : v_l), (x_{l+1} : v_{l+1})\}$, 1)$\forall_I Sup(C_i \Longrightarrow I) = \sum_k Sup_k|C_j((x_{l+1}, v_{l+1}) \Longrightarrow I)$; 2)$Sup(C_i) = \sum_k Sup_k|C_j((x_{l+1}, v_{l+1}))$, where $Sup_k|C_j(*)$ indicates the support of $*$ in the projected context log from the k-th context range of C_j.

This theorem is also easy to prove. With this theorem, we can perform behavior pattern mining in projected context logs. For example, Table IV shows the projected context logs of the context log in Table II w.r.t. prefix $\{(b : 3)\}$ for each context range of $\{(b : 3)\}$. From the projected context logs in Table IV, we can calculate both $\forall_I(Sup(\{(b : 3), (c : 5)\} \Longrightarrow I)$ and $Sup(\{(b : 3), (c : 5)\})$. For instance, denoting $\{(b : 3)\}$ as C_j, $Sup(\{(b : 3), (c : 5)\}) = Sup_1|C_j((c : 5)) + Sup_2|C_j((c : 5)) = 1 + 2 = 3$.

TABLE IV
CONTEXT RANGE VIEW: THE PROJECTED CONTEXT LOGS OF THE CONTEXT LOG IN TABLE II W.R.T. PREFIX $\{(b:3)\}$

Context Range 1		
Tid	Context	Interaction
t_1	$\{(c:5)\}$	Null
t_2	$\{(c:5),(d:1)\}$	I_1
t_3	$\{(c:5),(d:1)\}$	Null

Context Range 2		
Tid	Context	Interaction
t_6	$\{(c:5),(d:0),(e:1)\}$	Null
t_7	$\{(c:5),(d:1)\}$	Null
t_8	$\{(c:5),(d:1)\}$	Null
t_9	$\{(c:4),(d:1)\}$	I_2
t_{10}	$\{(c:5)\}$	I_1

V. BP-GROWTH: COMBINING PROMISING OPTIMIZING STRATEGIES

In above discussions, we find that the strategies of reducing redundant data and data set partition can be applied to behavior pattern mining. In this section, we propose a novel algorithm for behavior pattern mining called BP-Growth (**B**ehavior **P**attern **Growth**) by combining the two strategies. The main steps of BP-Growth are shown in Algorithm 1, where *redundant context records* denote the context records which have duplicated contexts with adjacent context records and only contain empty interaction records.

Algorithm 1 BP-Growth

Input1: a context log $R = r_1 r_2 ... r_n$;
Input2: an interaction set $\Gamma = \{I_1, I_2, ... I_Q\}$;
Input3: min_sup, min_conf;

1: Find all promising 1-context set $\mathsf{C}_p^1 = \{C_p^1 | \exists_I Sup(C_p^1 \Longrightarrow I) \geq min\_sup\}$.
2: Remove $(y:u)$ from R where $(y:u) \notin \mathsf{C}_p^1$.
3: Remove redundant context records.
4: **for each** 1-context C_p^1 **in** C_p^1 **do**
5: **for each** I **in** Γ **do**
6: **if** $Sup(C_p^1 \Longrightarrow I) \geq min\_sup$ **and** $Conf(C_p^1 \Longrightarrow I) \geq min\_conf$ **then**
7: Output $C_p^1 \Longrightarrow I$;
8: $SearchBP(C_p^1, \mathsf{C}_p^1, R_1|C_p^1, ..., R_n|C_p^1)$;

The procedure of $SearchBP()$ is outlined as follows. Firstly, the method $SearchBP()$ receives a promising context C_p^l, a set of promising 1-contexts C_p^1, and a set of projected context logs $\{R_1|C_p^l, ..., R_n|C_p^l\}$ where $R_i|C_p^l$ denotes the projected context log of the i-th context range of C_p^l $(1 \leq i \leq n)$. Then it tries to find all promising 1-contexts in $R|C_p^l$ as C_p^{*1} for extending C_p^l. For each extended $l+1$-context C_p^{l+1}, the method $SearchBP()$ firstly checks C_p^{l+1} and then recursively calls $SearchBP()$ for C_p^{l+1} if C_p^{l+1} is promising.

The pseudo code of $SearchBP()$ is shown in Algorithm 2, where for each promising 1-context C_p^1, an array is constructed for recording its support w.r.t. each interaction record in Γ, denoted as $C_p^1.support[]$. $C_p^1.support[0]$ is used for counting N_0, i.e., the number of context ranges of C_p^1 which only contain empty interaction records. Moreover, $C_p^l \cdot C_p^1$ denotes the context generated by appending C_p^1 to C_p^l.

Algorithm 2 SearchBP

Input1: a promising l-context C_p^l;
Input2: a set of promising 1-contexts C_p^1;
Input3: a set of projected context logs $\{R_1|C_p^l, ..., R_n|C_p^l\}$;

1: //Init
2: Init $\mathsf{C}_p^{*1} = \phi$;
3: **for each** C_p^1 **in** C_p^1 **do**
4: Init $C_p^1.support[]$;
5: //Count support
6: **for each** projected context log $R_i|C_p^l (1 \leq i \leq n)$ **do**
7: Remove redundant suffixes.
8: Scan $R_i|C_p^l$ to update $C_p^1.support[]$.
9: //Count confidence
10: **for each** C_p^1 **in** C_p^1 **do**
11: **for** $1 \leq n \leq Q$ **do**
12: **if** $C_p^1.support[n] \geq min\_sup$ **then**
13: $\mathsf{C}_p^{*1} \cup = C_p^1$;
14: **if** $\frac{C_p^1.support[n]}{\sum_{n=0}^{Q} C_p^1.support[n]} \geq min\_conf$ **then**
15: Output $C_p^l \cdot C_p^1 \Longrightarrow I_n$;
16: //Partition data set
17: **for each** 1-context C_p^{*1} **in** C_p^{*1} **do**
18: $C_p^{l+1} = C_p^l \cdot C_p^{*1}$;
19: **for each** $R_i|C_p^{l+1}$ **do**
20: Remove $(y:u)$ from $R_i|C_p^{l+1}$ where $(y:u) \notin \mathsf{C}_p^{*1}$.
21: $SearchBP(C_p^{l+1}, \mathsf{C}_p^{*1}, R_1|C_p^{l+1}, ..., R_{n'}|C_p^{l+1})$;

TABLE V
THE PRUNED CONTEXT LOG IN TABLE II.

Tid	Context	Interaction
t_2	$\{(a:2),(b:3),(c:5)\}$	I_1
t_4	$\{(c:5)\}$	I_2
t_5	$\{(a:2),(c:5)\}$	Null
t_6	$\{(a:2),(b:3),(c:5)\}$	Null
t_9	$\{(a:2),(b:3)\}$	I_2
t_{10}	$\{(a:2),(b:3),(c:5)\}$	I_1

The following example illustrates the process of mining behavior patterns from the context log in Table II by BP-Growth for easing understanding how BP-Growth works. The min_sup and min_conf are set to be 2 and 0.6, respectively.

Example 1 (Mining behavior patterns): Firstly, we remove the unpromising contextual feature-value pairs $(a:3)$, $(b:2)$, $(c:4)$, $(d:0)$, $(d:1)$ and $(e:1)$, and then remove all redundant context records, i.e., the context records with timestamps t_1, t_3, t_7, and t_8. The pruned context log is shown in Table V.

Secondly, we find all behavior patterns with 1-contexts from the pruned context log as follows: $\{(a:2)\} \Longrightarrow I_1$ (support: 2, confidence: 0.66), $\{(b:3)\} \Longrightarrow I_1$ (support: 2, confidence: 0.66), and $\{(c:5)\} \Longrightarrow I_1$ (support: 2, confidence: 0.66). Then for each promising contextual feature-value pair (f_i, v_i), we build projected context logs w.r.t. prefix (f_i, v_i) in its each context range, as illustrated in Table VI. Notice that 1) an empty suffix in the second projected context log w.r.t. prefix $\{(b:3)\}$ is maintained to indicate that the suffixes with timestamps t_6 and t_{10} are not in the same context range of $\{(b:3),(c:5)\}$; and 2) all the projected context logs w.r.t.

TABLE VI
CONTEXT RANGE VIEW: THE PROJECTED CONTEXT LOGS OF THE CONTEXT LOG IN TABLE V W.R.T. (A) PREFIX $\{(a:2)\}$, (B) PREFIX $\{(a:2),(b:3)\}$, AND (C) PREFIX $\{(b:3)\}$.

(a)

Context Range 1		
Tid	Context	Interaction
t_2	$\{(b:3),(c:5)\}$	I_1

Context Range 2		
Tid	Context	Interaction
t_5	$\{(c:5)\}$	Null
t_6	$\{(b:3),(c:5)\}$	Null
t_9	$\{(b:3)\}$	I_2
t_{10}	$\{(b:3),(c:5)\}$	I_1

(b)

Context Range 1		
Tid	Context	Interaction
t_2	$\{(c:5)\}$	I_1

Context Range 2		
Tid	Context	Interaction
t_6	$\{(c:5)\}$	Null
t_9	$\{\}$	Null
t_{10}	$\{(c:5)\}$	I_1

(c)

Context Range 1		
Tid	Context	Interaction
t_2	$\{(c:5)\}$	I_1

Context Range 2		
Tid	Context	Interaction
t_6	$\{(c:5)\}$	Null
t_9	$\{\}$	Null
t_{10}	$\{(c:5)\}$	I_1

prefix $(c:5)$ for each context range of $(c:5)$ are empty and avoided to be generated.

Thirdly, we find the promising contextual feature-value pairs from the projected context log in Table VI (a), i.e.,$(b:3)$ and $(c:5)$, then output the behavior patterns $\{(a:2),(b:3)\} \Longrightarrow I_1$ (support: 2, confidence: 0.66) and $\{(a:2),(c:5)\} \Longrightarrow I_1$ (support: 2, confidence: 0.66). Then we build the projected context logs w.r.t. prefix $\{(a:2),(b:3)\}$ for each context range of $\{(a:2),(b:3)\}$ as illustrated in Table VI (b). The projected context logs w.r.t. prefix $\{(a:2),(c:5)\}$ are avoided to built since they are empty.

Fourthly, we find the promising contextual feature-value pairs from the projected context log in Table VI (b), i.e., $(c:5)$, and output the behavior patterns $\{(a:2),(b:3),(c:5)\} \Longrightarrow I_1$ (support: 2, confidence: 0.66).

Finally, since the projected context logs w.r.t. $\{(a:2),(b:3),(c:5)\}$ are empty, the mining in the projected context logs w.r.t. prefix $\{(a:2)\}$ is end and we come back to the projected context logs w.r.t. prefix $\{(b:3)\}$ (Table VI (c)). Specially, we find the only one promising contextual feature-value pair $(c:5)$, and output the behavior pattern $\{(b:3),(c:5)\} \Longrightarrow I_1$ (support: 2, confidence: 0.66).

In BP-Growth, generating projected context logs needs additional memory cost. Wisely, we can avoid generating physical projected context logs by taking advantage of the *pseudo projection* technology [12]. The main idea of pseudo projection is to replace the physical projected data by a group of pointers when the memory can hold the original data to be projected. To be specific, a pseudo projected context log consists of a series of pseudo suffixes. Each pseudo suffix consists of a Tid which indicates a context record in the original context log and an index which indicates the beginning position to separate a suffix from the corresponding context record. For example, Table VII shows the pseudo projected context logs w.r.t. prefix $\{(a:2)\}$ for each context range of $(a:2)$. In the first pseudo suffix of the pseudo projected context log from context range 1, the Tid t_2 and index 2 indicate a suffix beginning from the second contextual feature-value pair of the context record with timestamp t_2, i.e., $(b:3)$, to the end of the context record. Similarly, in the first pseudo suffix of the pseudo projected log from context range 2, the Tid t_5 and index 2 indicate a suffix beginning from the second contextual feature-value pair of the context record with timestamp t_5, i.e., $(c:5)$, to the end of the context record, which is also $(c:5)$.

TABLE VII
CONTEXT RANGE VIEW: THE PSEUDO PROJECTED LOGS OF THE CONTEXT LOG IN TABLE V W.R.T. PREFIX $\{(a:2)\}$

Context Range 1	
Tid	Index
t_2	2

Context Range 2	
Tid	Index
t_5	2
t_6	2
t_9	2
t_{10}	2

With pseudo projected context logs, we can still find promising contextual feature-value pairs and behavior patterns through BP-Growth but do not need to generate physical projected context logs. A constraint of pseudo projection technology is that we can not reduce the mining space for behavior pattern mining by removing the redundant context data in pseudo projected context logs as we do in physical projected context logs. It is because each accessed contextual feature-value pairs from a pseudo projected context log is substantially located in the pruned original context log and only has one copy. An unpromising contextual feature-value pair (f_i, v_i) in one pseudo projected context log S_1 may be promising in another. Thus, if we remove (f_i, v_i) when operating S_1, the operation for other pseudo projected context logs may be affected.

VI. EXPERIMENTS

To evaluate the efficiency of BP-Growth for behavior pattern mining in terms of running time and memory cost, we conduct extensive experiments on real context log collected from mobile users. The experimental data sets include 10 university volunteers' context logs spanning for one month and contain rich types of context data and user interaction records such as day name, battery level, cell ID, listening to music, playing games, email and so on. These data are collected by Cao et al. [9] and can be downloaded from http://dm.ustc.edu.cn/paperlist.html. For more details of the data sets, please refer to [9].

A. Experimental Set Up

To effectively evaluate the efficiency of BP-Growth for behavior pattern mining, we select three algorithms for behavior pattern mining as baselines.

- **GCPM-H**: An implementation of GCPM which takes advantage of CH-Trees instead of context sets to store

and access candidate promising contexts. The experimental results show that the memory cost of GCPM-H is comparable to the original GCPM while the running time drops more than 10 ten times averagely [3, 9].

- **GCPM-H-R**: An optimization of GCPM-H which adopts the strategy of removing redundant data. In other words, it firstly removes redundant data and then performs GCPM-H in the pruned context log.
- **BP-Growth-NR**: An algorithm which takes advantage of the same strategy of data set partition as BP-Growth but does not remove redundant data in the original context log and projected context logs.

As mentioned above, it is desirable to directly perform behavior pattern mining on users' smart devices for protecting user privacy and saving data flow, which motivates us to find an efficient algorithm for behavior pattern mining which can work well in practical mobile computing environments. Therefore, all experiments in this paper are conducted in a practical mobile computing environment. To be specific, in the following experiments, the BP-Growth algorithm and all baselines are implemented by Qt on a Nokia N97 smart phone with S60v5 operating system, 128MB main memory and 434MHZ CPU. It is worth noting that the operating system needs more than 70MB main memory for running. Consequently, the development guideline claims the available main memory for third party programs are less than 40MB.

B. Cost of Running Memory

Since the available memory of smart phones for third part programs is usually much less than PC, memory cost is an important indicator for the efficiency of algorithms which run in mobile computing environments. Therefore, in this section, we firstly compare the memory cost of BP-Growth and the baselines on varying context logs. Limited by space, it is not feasible to show the detailed experimental results for all collected context logs. Instead, we randomly select two context logs, namely, R_A and R_B to show the detailed experimental results and report part of experimental results for other context logs. Finally, we summarize the observations from experimental results.

Figure 2 (a) and Figure 2 (b) compare the memory cost of BP-Growth, the optimization of BP-Growth which takes advantage of pseudo projection technology (denoted as *BP-Growth-P*), and the baseline algorithms with varying settings of min_sup on R_A and R_B, respectively. The min_conf is set to be 0.5. The memory cost of GCPM-H and GCPM-H-R with $min\_sup = 2$ is not available because in this case its memory cost is over the memory limit (40MB) of the experimental smart phone and the program terminates with an exception. From the figures we can see that the memory cost of BP-Growth, BP-Growth-P and BP-Growth-NR is dramatically less than GCPM-H and GCPM-H-R with relatively small min_sup. To be specific, they reduce more than 75% memory cost compared with GCPM-H in the best case. In contrast, when min_sup increases bigger and bigger their memory cost becomes comparable to GCPM-H.

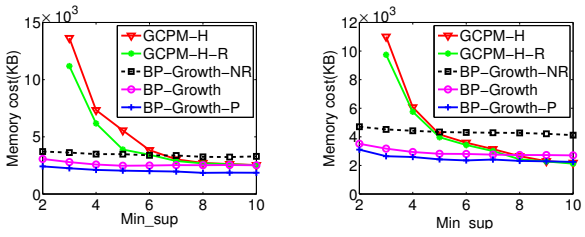

Fig. 2. Memory cost of varying algorithms for context logs (a) R_A and (b) R_B with varying min_sup and the fixed $min\_conf = 0.5$.

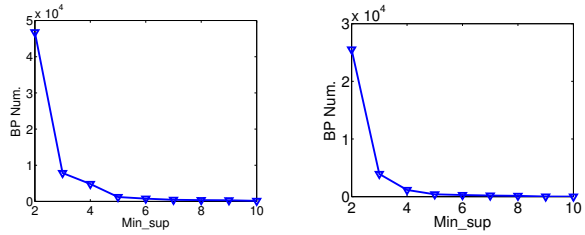

Fig. 3. Numbers of mined behavior patterns for context logs (a) R_A and (b) R_B with varying min_sup and the fixed $min\_conf = 0.5$.

It is worth noting that the relatively good performance of GCPM in terms of memory cost with big min_sup makes no sense because rare behavior patterns are mined in this case, and all comparing algorithms quickly terminate. Figure 3 (a) and Figure 3 (b) show the numbers of mined behavior patterns with varying settings of min_sup and the fixed $min\_conf = 0.5$ on R_A and R_B, respectively. We can see that the number of mined behavior patterns drops quickly when the min_sup increases. Thus, it is more meaningful to compare the memory cost of different algorithms by considering their performance with relatively small min_sup.

Moreover, from Figure 2 (a) and Figure 2 (b) we can see that the memory cost of BP-Growth, BP-Growth-P, and BP-Growth-NR is stable with varying settings of min_sup while GCPM-H and GCPM-H-R's memory cost is sensitive to the decrease of min_sup.

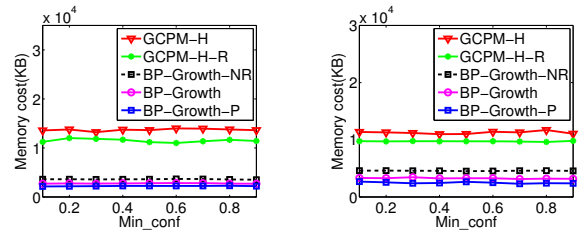

Fig. 4. Memory cost of varying algorithms for context logs (a) R_A and (b) R_B with varying min_conf and the fixed $min\_sup = 3$.

Figure 4 (a) and Figure 4 (b) compare the memory cost of BP-Growth, BP-Growth-P, and the baseline algorithms with varying settings of min_conf on R_A and R_B, respectively. The min_sup is set to be 3. From the figures we can see that the memory cost of BP-Growth, BP-Growth-P, and BP-Growth-NR is dramatically less than GCPM-H and GCPM-H-R with varying settings of min_conf. Moreover, the memory

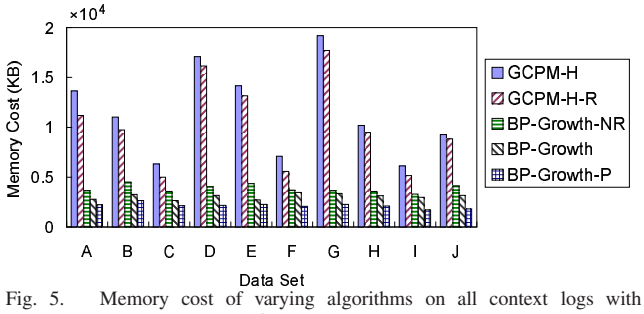

Fig. 5. Memory cost of varying algorithms on all context logs with $min\_sup = 3$ and $min\_conf = 0.5$.

cost of all algorithms is stable with the change of min_conf. The experiments on other context logs show the similar phenomenon.

Figure 5 shows the memory cost of all algorithms with $min\_sup = 3$ and $min\_conf = 0.5$ on all context logs. Such a setting of min_sup leads to the worst case of memory cost for all algorithms on the basis of ensuring that their running memory is not over the memory limit of experimental device. From this figure we can see that the memory cost of BP-Growth, BP-Growth-P, and BP-Growth-NR is dramatically less than GCPM-H and GCPM-H-R on all context logs, and BP-Growth-P always performs best in terms of memory cost.

Summary: by comprehensively analyzing the experimental results for memory cost, we summarize the observations as follows.

a) BP-Growth, BP-Growth-P and BP-Growth-NR dramatically outperform GCPM-H and GCPM-H-R in terms of memory cost and can run well in a middle-end smart phone with limited memory. It implies that the strategy of data set partition plays a more important role in reducing the memory cost for behavior pattern mining than the strategy of removing redundant data.

b) GCPM-H-R outperforms GCPM-H slightly in terms of memory cost while BP-Growth also outperforms BP-Growth-NR slightly in terms of memory cost. It implies that the strategy of removing redundant data can reduce the memory cost for behavior pattern mining though the effect is limited.

c) BP-Growth-P outperforms BP-Growth slightly in terms of memory cost. It implies that the pseudo projection technology can be integrated with the strategy of data set partition for reducing the memory cost for behavior pattern though the effect is limited.

C. Computation Efficiency

Except for memory cost, computation efficiency is another important performance indicator of the algorithms for behavior pattern mining. In this section, we evaluate the computation efficiency of BP-Growth and other baselines in terms of running time. Similar to the experiments of memory cost, we firstly show the detailed experimental results on two randomly selected context logs, namely, R_A and R_B, and then report part of experimental results for other context logs. Finally, we summarize the observations from experimental results.

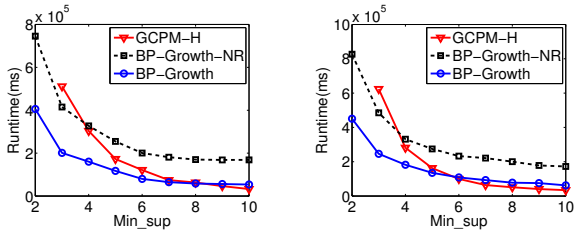

Fig. 6. Running time of varying algorithms with varying min_sup for context logs (a) R_A and (b) R_B with $min\_conf = 0.5$.

Figure 6 (a) and Figure 6 (b) compare the running time of BP-Growth, BP-Growth-NR and GCPM-H with varying settings of min_sup on R_A and R_B, respectively. The min_conf is set to be 0.5. The running time of GCPM-H with $min\_sup = 2$ is not available because its memory cost is over the memory limit (40MB) of the experimental smart phone in this case. From the figures we can see that the running time of BP-Growth is dramatically less than BP-Growth-NR and GCPM-H with relatively small min_sup. To be specific, it reduces more than 60% running time compared with GCPM-H in the best case. In contrast, when min_sup increases bigger and bigger their running time becomes to be comparable to GCPM-H. Moreover, BP-Growth-NR outperforms GCPM-H in terms of running time with small min_sup and becomes to under-perform the latter when the min_sup increases. As mentioned in Section VI-B, we should consider the case of small min_sup more when comparing the efficiency of different algorithms for behavior pattern mining since there are rare mining results with big min_sup. Therefore, we can roughly conclude that the running time of BP-Growth-NR is at least comparable to GCPM-H.

The two figures do not show the running time of GCPM-H-R and BP-Growth-P because we observe that the running time of GCPM-H-R is very close to GCPM-H and the running time of BP-Growth-P is very close to BP-Growth with varying min_sup. Showing their running time will cause bad clarity of these figures.

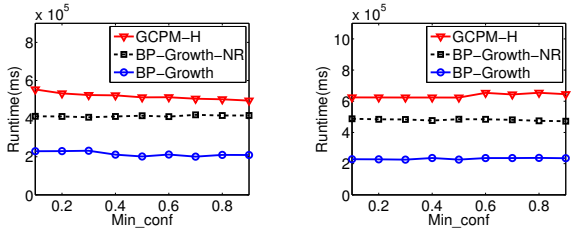

Fig. 7. Running time of varying algorithms for context logs (a) R_A and (b) R_B with varying min_conf and the fixed $min\_sup = 3$.

Figure 7 (a) and Figure 7 (b) compare the running time of BP-Growth, BP-Growth-NR and GCPM-H with varying settings of min_conf on R_A and R_B, respectively. The min_sup is set to be 3. From the figures we can see that the running time of BP-Growth is dramatically less than GCPM-H and BP-Growth-NR with varying settings of min_conf.

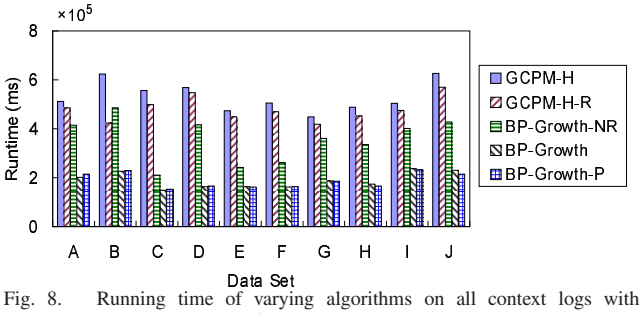

Fig. 8. Running time of varying algorithms on all context logs with $min\_sup = 3$ and $min\_conf = 0.5$.

Moreover, the running time of all algorithms is stable with the change of min_conf. The experiments on other context logs show the similar phenomenon.

Figure 8 shows the running time of all algorithms with $min\_sup = 3$ and $min\_conf = 0.5$ on all context logs. From this figure we can see that the running time of BP-Growth, BP-Growth-P is dramatically less than BP-Growth-NR, GCPM-H and GCPM-H-R on all context logs.

Summary: by comprehensively analyzing the experimental results for running time, we summarize the observations as follows.

a) BP-Growth, BP-Growth-P dramatically outperform BP-Growth-NR, GCPM-H and GCPM-H-R in terms of running time. It implies that the strategy of removing redundant data plays a more important role in reducing the running time for behavior pattern mining than the strategy of data set partition, and the improvement for data set partition based algorithms is more dramatic.
b) GCPM-H-R is roughly comparable to GCPM-H in terms of running time. It implies that applying the strategy of removing redundant data to an Apriori-like algorithm only leads to very limited improvement of running time.
c) BP-Growth-P is roughly comparable to BP-Growth in terms of running time. It implies that the pseudo projection technology only has very limited effect on reducing the running time of BP-Growth.

VII. Conclusion and Future Work

In this paper, we studied several strategies which may be used for improving the efficiency of behavior pattern mining in terms of computing efficiency and memory cost. To be specific, we searched some typical optimizing strategies for association rule mining and discuss the feasibility of applying them to behavior pattern mining since the two problems are similar in many aspects. Moreover, we proposed a novel and efficient algorithm named BP-Growth for behavior pattern mining by combining two promising strategies. The extensive experiments in a practical mobile computing environment clearly show that BP-Growth and its optimization significantly outperform GCPM and other two baselines which adopt one of the two promising strategies in terms of both running time and memory cost.

One challenge of utilizing behavior patterns is that there usually exists thousands of behavior patterns for an individual mobile user. Usually many of them are semantically similar or related and may be summarized by several representative patterns. As for future work, we plan to study the effective approaches to summarize lots of behavior patterns for easing the representation of user habits while maintaining as much information as possible.

VIII. Acknowledgements

The research was supported by grants from Nokia Research Center China, Natural Science Foundation of China (Grant No.s 60775037, 70890082 and 71028002), The National Major Special Science & Technology Projects (Grant No. 2011ZX04016-071), National Science Foundation (CCF-1018151), Research Fund for the Doctoral Program of Higher Education of China (20093402110017). The authors would like to thank Nokia for giving permission to use their data collection platform.

References

[1] Palen, L., Salzman, M. and Youngs, E., "Going wireless: behavior & practice of new mobile phone users." in *CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work*. New York, NY, USA: ACM Press, 2000, pp. 201–210.
[2] Tseng, V. S. and Lin, K. V., "Efficient mining and prediction of user behavior patterns in mobile web systems." *Information and Software Technology.*, pp. 357–369, 2006.
[3] Cao, H., Bao, T., and Yang, Q. et al., "An effective approach for mining mobile user habits." in *Proceedings of the 19th ACM international conference on Information and knowledge management (CIKM'10)*, 2010, pp. 1677–1680.
[4] Schilit, B., Adams, N., and Want, R., "Context-aware computing applications." in *Proceedings of the Workshop on Mobile Computing Systems and Applications.* IEEE Computer Society, 1994, pp. 85–90.
[5] Agrawal, R., Imieliński, T. and Swami, A., "Mining association rules between sets of items in large databases." in *Proceedings of the 1993 ACM SIGMOD international conference on Management of data (SIGMOD '93)*. ACM, 1993, pp. 207–216.
[6] Fukuda, T., Morimoto, Y. and Morishita, S. et al., "Mining optimized association rules for numeric attributes," in *Proceedings of the 5th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'96)*, 1996, pp. 182–191.
[7] Han, J., Pei, J. and Yin, Y., "Mining frequent patterns without candidate generation." in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*. ACM, 2000, pp. 1–12.
[8] Agrawal, R. and Srikant, R., "Fast algorithms for mining association rules." in *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*, 1994, pp. 487–499.
[9] Cao, H., Bao, T., and Chen, E., "An effective approach for mining mobile user habits." University of Science and Technology of China., Tech. Rep., 2010. [Online]. Available: http://dm.ustc.edu.cn/paperlist.html
[10] Han, J., Pei, J. and Yin, Y. et al., "Mining frequent patterns without candidate generation: a frequent-pattern tree approach." *Data Min. Knowl. Discov.*, vol. 8, no. 1, pp. 53–87, 2004.
[11] Wang, J., Han, J., and Pei, J., "Closet+: Searching for the best strategies for mining frequent closed itemsets." in *Proceedings of the 9rd ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'03).*, 2003, pp. 236–245.
[12] Pei, J., and Han, J., and Mortazavi-Aslet, B. et al., "Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth." in *Proceedings of the 16th IEEE International Conference on Data Engineering (ICDE'01)*, 2001, p. 215C226.
[13] Chiu, D., Wu, Y., and Chen, A. L. P., "An efficient algorithm for mining frequent sequences by a new strategy without support counting." in *Proceedings of the 20th International Conference on Data Engineering (ICDE04).* IEEE Computer Society, 2004, pp. 375–386.
[14] Chen, E., and Cao, H., and Li, Q. et al., "Efficient strategies for tough aggregate constraint-based sequential pattern mining," *Inf. Sci.*, vol. 178, no. 6, pp. 1498–1518, 2008.