Interactive Constraint-Based Sequential Pattern Mining ⋆

Marek Wojciechowski

Poznan University of Technology
Institute of Computing Science
ul. Piotrowo 3a, 60-965 Poznan, Poland
Marek.Wojciechowski@cs.put.poznan.pl

Abstract. Data mining is an interactive and iterative process. It is very likely that a user will execute a series of similar queries differing in pattern constraints and mining parameters, before he or she gets satisfying results. Unfortunately, data mining algorithms currently available suffer from long processing times, which is unacceptable in case of interactive mining. In this paper we discuss efficient processing of sequential pattern queries utilizing cached results of other sequential pattern queries. We analyze differences between sequential pattern queries and propose algorithms that in many cases can be used instead of time-consuming mining algorithms.

1 Introduction

Data mining aims at discovery of useful patterns from large databases or warehouses. One of the most popular data mining methods is sequential pattern discovery introduced in [2]. Informally, sequential patterns are the most frequently occurring subsequences in sequences of sets of items. The initial formulation of the problem was significantly extended in [10], where a taxonomy on items was added to support discovery of so called generalized sequential patterns, and three time constraints (min-gap, max-gap, and time window) were introduced to be used when checking if a given source sequence contains a given pattern. For that extended problem formulation, an efficient algorithm called *GSP* was proposed. Applications of sequential patterns include analysis of telecommunication systems, discovering frequent buying patterns, analysis of patients' medical records, etc.

From a user's point of view, data mining can be seen as an interactive and iterative process of advanced querying: a user specifies the source dataset and the requested class of patterns, the system chooses the appropriate data mining algorithm and returns discovered patterns to the user [4][6]. A user interacting with a data mining system has to specify several constraints on patterns to be discovered. However, usually it is not trivial to find a set of constraints leading

⋆ This work was partially supported by the grant no. KBN 43-1309 from the State Committee for Scientific Research (KBN), Poland.

to the satisfying set of patterns. Thus, users are very likely to execute a series of similar data mining queries before they find what they need. Unfortunately, data mining algorithms require long processing times, which makes such interaction difficult.

In this paper, we discuss efficient sequential pattern discovery in the presence of materialized results of previous sequential pattern queries. We claim that a data mining system should exploit the fact that a user is very likely to execute a number of similar sequential pattern queries during a single session. We propose caching results of mining queries by materializing their results on disk (we assume that a data mining system is going to be assigned a certain amount of disk space for that purpose). It is obvious that materialized results of a query can be used to answer an identical query, therefore we concentrate on processing queries different from those whose results are available. The possibility of answering a query using known results of another query depends on the differences between the two queries. Our goal is to provide criteria for determining if cached results of a given query can be used to answer the current query without running a complete mining algorithm, and introduce efficient sequential pattern query processing algorithms exploiting materialized patterns.

Exploiting cached results of previous mining queries has been studied in the context of association rules [3][7]. However, direct application of methods and techniques introduced for association rules to sequential pattern discovery problem is not possible since different types of constraints are available in the two problems. Nevertheless, it seems that the general ideas should stay unchanged.

In has been observed [3] that the three particularly interesting relationships between two mining queries DMQ_1 and DMQ_2 extracting patterns from the same data are equivalence, inclusion, and dominance. The three relationships are interesting since they represent situations, where one data mining query can be efficiently answered using the results of another query. Differences between mining queries leading to these relationships were analyzed only in the context of association rules. In this paper we present analogous analysis concerning sequential patterns. Thus, most of our work can be regarded as the extension of the approach from [3] into sequential pattern discovery.

1.1 Sequential Patterns

Let $L = l_1, l_2, ..., l_m$ be a set of literals called items. An *itemset* is a non-empty set of items. A *sequence* is an ordered list of itemsets and is denoted as $< X_1X_2...X_n >$, where X_i is an itemset ($X_i \subseteq L$). X_i is called an *element* of the sequence. The *size* of a sequence is the number of items in the sequence. The *length* of a sequence is the number of elements in the sequence. Let D be a set of variable length sequences (called *data-sequences*), where for each sequence $S =< X_1X_2...X_n >$, a timestamp is associated with each X_i.

With no time constraints we say that a sequence $X =< X_1X_2...X_n >$ is *contained* in a data-sequence $Y =< Y_1Y_2...Y_m >$ if there exist integers $i_1 < i_2 < ... < i_n$ such that $X_1 \subseteq Y_{i_1}, X_2 \subseteq Y_{i_2}, ..., X_n \subseteq Y_{i_n}$. We call $< Y_{i_1}Y_{i_2}...Y_{i_n} >$ an *occurrence* of X in Y. We consider the following user-specified time constraints while looking for occurrences of a given sequence: minimal and maximal gap allowed between consecutive elements of an occurrence of the sequence (called *min-gap* and *max-gap*), and time window that allows a group of consecutive elements of a data-sequence to be merged and treated as a single element as long as their timestamps are within the user-specified *window-size*.

The *support* of a sequence $< X_1 X_2 ... X_n >$ in D is the fraction of data-sequences in D that contain the sequence. A *sequential pattern* is a sequence whose support in D is above the user-specified threshold.

1.2 Relationships between Results of Data Mining Queries

Two data mining queries are *equivalent* if for all datasets they both return the same set of patterns and the values of statistical significance measures (e.g. support) for each pattern are the same in both cases. A data mining query DMQ_1 *includes* a data mining query DMQ_2 if for all datasets each pattern in the results of DMQ_2 is also returned by DMQ_1 with the same values of the statistical significance measures. A data mining query DMQ_1 *dominates* a data mining query DMQ_2 if for all datasets each pattern in the results of DMQ_2 is also returned by DMQ_1, and for each pattern returned by both queries its values of the statistical significance measures evaluated by DMQ_1 are not less than is case of DMQ_2. Equivalence is a particular case of inclusion, and inclusion is a particular case of dominance. Equivalence, inclusion, and dominance meet the transitivity property.

If for a given query, results of a query equivalent to it, including it, or dominating it are available, the query can be answered without running a costly mining algorithm. In case of equivalence no processing is necessary, since the queries have the same results. In case of inclusion, one scan of the materialized query results is necessary to filter out patterns that do not satisfy constraints of the included query. In case of dominance, one verifying scan of the source dataset is necessary to evaluate the statistical significance of materialized patterns (filtering out the patterns that do not satisfy constraints of the dominated query is also required).

1.3 Related Work

To facilitate interactive and iterative pattern discovery, [8] proposed to materialize patterns discovered with the least restrictive selection criteria, and answer incoming queries by filtering the materialized pattern collection. This approach is not a perfect solution of the problem since pattern mining with very low minimum support thresholds might lead to collections of frequent patterns even larger than the original database. Moreover, restricting certain constraints (e.g. time constraints in the context of sequential pattern mining) not only makes some patterns infrequent but also changes the support of patterns that remain frequent.

Much more reasonable and flexible solutions supporting interactive and iterative mining were presented in [7], in the context of association rules. The solutions presented there consisted in caching results of mining queries. In the approach, materialization of frequent itemsets instead of rules was proposed. However, in some cases it was required to materialize also some of the infrequent itemsets.

Most of the research on sequential patterns focused on introducing new algorithms, more efficient than *GSP* (e.g. [5][9]). However, the novel methods do not handle time constraints and taxonomies. Thus, *GSP* still remains the most general sequential pattern discovery algorithm and the reference point for new methods and techniques.

1.4 Organization of the Paper

The paper is organized as follows. Section 2 presents constraints that can be specified in sequential pattern mining. In Sect.3, relationships between sequential pattern queries are discussed. Section 4 contains efficient sequential pattern query processing algorithms. Experimental results concerning the proposed algorithms are presented in Sect.5. We conclude with a summary in Sect.6.

2 Constraint-Based Sequential Pattern Mining

In constraint-based sequential pattern mining, we identify the following classes of constraints: database constraints, pattern constraints, and time constraints. Database constraints are used to specify the source dataset. Pattern constraints specify which patterns are interesting and should be returned by the query. Finally, time constraints influence the process of checking whether a given data-sequence contains a given pattern.

The basic formulation of the sequential pattern discovery problem introduces three time constraints: max-gap, min-gap, and time window, and assumes only one pattern constraint (expressed by means of the minimum support threshold). We model pattern constraints as complex Boolean predicates having the form of a conjunction of basic Boolean predicates on patterns presented below:

– π(**SPG**, α, pattern) - true if pattern support is greater than α, false otherwise;
– π(**SL**, α, pattern) - true if pattern size is less than α, false otherwise;
– π(**SG**, α, pattern) - true if pattern size is greater than α, false otherwise;
– π(**LL**, α, pattern) - true if pattern length is less than α, false otherwise;
– π(**LG**, α, pattern) - true if pattern length is greater than α, false otherwise;
– π(**C**, β, pattern) - true if β is a subsequence of the pattern, false otherwise;
– π(**NC**, β, pattern) - true if β is not a subsequence of the pattern, false otherwise.

We believe that the above list of predicates is sufficient to allow users to express their pattern selection criteria. For simplicity's sake, in length and size predicates we consider only sharp inequalities.

3 Relationships between Sequential Pattern Queries

Inclusion and dominance relationships between two data mining queries are defined for queries operating on the same dataset. Therefore, analyzing differences between sequential pattern queries, we consider only differences in time and pattern constraints.

Definition 1. *Given two basic Boolean pattern predicates b_1 and b_2, we say that b_2 is* stronger *than b_1 if one of the following conditions holds:*

1. *$b_1 = \pi($**SPG**, α_1, pattern$)$ and $b_2 = \pi($**SPG**, α_2, pattern$)$, where $\alpha_2 > \alpha_1$,*
2. *$b_1 = \pi($**SG**, α_1, pattern$)$ and $b_2 = \pi($**SG**, α_2, pattern$)$, where $\alpha_2 > \alpha_1$,*
3. *$b_1 = \pi($**SL**, α_1, pattern$)$ and $b_2 = \pi($**SL**, α_2, pattern$)$, where $\alpha_2 < \alpha_1$,*
4. *$b_1 = \pi($**LG**, α_1, pattern$)$ and $b_2 = \pi($**LG**, α_2, pattern$)$, where $\alpha_2 > \alpha_1$,*
5. *$b_1 = \pi($**LL**, α_1, pattern$)$ and $b_2 = \pi($**LL**, α_2, pattern$)$, where $\alpha_2 < \alpha_1$,*
6. *$b_1 = \pi($**C**, β_1, pattern$)$ and $b_2 = \pi($**C**, β_2, pattern$)$, where a pattern β_1 is a subsequence of the pattern β_2 and the size of β_1 is less than the size of β_2,*
7. *$b_1 = \pi($**NC**, β_1, pattern$)$ and $b_2 = \pi($**NC**, β_2, pattern$)$, where pattern β_2 is a subsequence of the pattern β_1 and the size of β_2 is less than the size of β_1.*

Definition 2. *We say that a data mining query DMQ_2* extends pattern constraints *of a data mining query DMQ_1 if any of the following conditions holds:*

1. *Pattern constraints of DMQ_1 have a form of a conjunction of n basic Boolean pattern predicates, pattern constraints of DMQ_2 have a form of a conjunction of $n+1$ basic Boolean pattern predicates ($n \geq 0$), and each basic Boolean pattern predicates in DMQ_1 also appears in DMQ_2;*
2. *DMQ_1 and DMQ_2 have pattern constraints p_1 and p_2 respectively, where p_1 and p_2 are conjunctions of n basic Boolean pattern predicates ($n \geq 1$), $p_1 = p \wedge b_1$, $p_2 = p \wedge b_2$ (p is a conjunction of $n-1$ basic Boolean pattern predicates), and b_2 is stronger than b_1;*
3. *It is possible to formulate a data mining query DMQ_3 such that DMQ_2 extends pattern constraints of DMQ_3 and DMQ_3 extends pattern constraints of DMQ_1. (The relationship of extending pattern constraints is transitive.)*

In other words, a data mining query DMQ_2 extends pattern constraints of a data mining query DMQ_1 if pattern constraints of DMQ_1 can be transformed into pattern constraints of DMQ_2 by appending new basic Boolean pattern predicates or replacing basic Boolean pattern predicates with stronger ones.

Given two sequential pattern queries, there are four cases possible regarding pattern constrains: DMQ_1 and DMQ_2 have the same pattern constraints, DMQ_1 extends pattern constraints of DMQ_2, DMQ_2 extends pattern constraints of DMQ_1, or pattern constraints of DMQ_1 and DMQ_2 are not comparable.

Definition 3. *We say that a data mining query DMQ_2* extends time constraints *of a data mining query DMQ_1 if any of the following conditions holds:*

1. *The value of the max-gap parameter in DMQ_2 is less than in DMQ_1 and both queries have the same value of the min-gap parameter, and the same value of the window-size parameter;*
2. *The value of the min-gap parameter in DMQ_2 is greater than in DMQ_1 and both queries have the same value of the max-gap parameter, and the same value of the window-size parameter;*
3. *The value of the window-size parameter in DMQ_2 is less than in DMQ_1 and both queries have the same value of the max-gap parameter, and the same value of the min-gap parameter;*
4. *It is possible to formulate a data mining query DMQ_3 such that DMQ_2 extends time constraints of DMQ_3 and DMQ_3 extends time constraints of DMQ_1. (The relationship of extending time constraints is transitive.)*

In other words, a data mining query DMQ_2 extends time constraints of a data mining query DMQ_1 if it restricts at least one of the time parameters (max-gap, min-gap, window-size) and does not relax any time parameters.

Given two sequential pattern queries, there are four cases possible regarding time constrains: DMQ_1 and DMQ_2 have the same time constraints, DMQ_1 extends time constraints of DMQ_2, DMQ_2 extends time constraints of DMQ_1, or time constraints of DMQ_1 and DMQ_2 are not comparable.

Example 1. Let us consider the following three sequential pattern queries, operating on the same dataset:

DMQ_1 = {max-gap: 100, min-gap: 0, window-size: 1, π(**SPG**, 0.2, pattern)}
DMQ_2 = {max-gap: 100, min-gap: 0, window-size: 1, π(**SPG**, 0.1, pattern) $\wedge$ π(**SG**, 3, pattern)}
DMQ_3 = {max-gap: 100, min-gap: 7, window-size: 1, π(**SPG**, 0.2, pattern) $\wedge$ π(**SG**, 3, pattern)}

DMQ_3 extends pattern constraints of DMQ_1 and DMQ_2, while pattern constraints of DMQ_1 and DMQ_2 are not comparable. DMQ_3 extends time constraints of DMQ_1 and DMQ_2, while time constraints DMQ_1 and DMQ_2 are the same.

The two relationships defined above concern the syntax of queries, while the general inclusion and dominance relationships refer to results of queries. Below we introduce three theorems regarding dependence of relationships between results of two queries on syntactic differences between the two queries. We also introduce several lemmas on which the proofs of theorems are based. For brevity, we do not include proofs of the lemmas since they come straight from the above definitions and inherent properties of pattern and time constraints.

Lemma 1. *Let b_1 and b_2 be basic Boolean pattern predicates such that b_2 is stronger than b_1. For each pattern p, if p satisfies b_2 then p satisfies b_1.*

Lemma 2. *Let DMQ_1 and DMQ_2 be two sequential pattern queries, operating on the same dataset and having the same time constraints. Let p_1 and p_2 denote pattern constraints of DMQ_1 and DMQ_2 respectively. If $p_2 = p_1 \wedge b$, where b is a basic Boolean pattern predicate, then DMQ_1 includes DMQ_2.*

Lemma 3. *Let DMQ_1 and DMQ_2 be two sequential pattern queries, operating on the same dataset and having the same time constraints. Let p_1 and p_2 denote pattern constraints of DMQ_1 and DMQ_2 respectively. If $p_1 = p \wedge b_1$ and $p_2 = p \wedge b_2$, where p is a conjunction of n basic Boolean pattern predicates ($n \geq 0$) and b_2 is stronger than b_1, then DMQ_1 includes DMQ_2.*

Theorem 1. *Let DMQ_1 and DMQ_2 be two sequential pattern queries, operating on the same dataset and having the same time constraints. If DMQ_2 extends pattern constraints of DMQ_1, then DMQ_1 includes DMQ_2.*

Proof. From the Definition 2, we know that if DMQ_2 extends pattern constraints of DMQ_1, then it is possible to formulate a sequence of sequential pattern queries DMQ_{i_1}, DMQ_{i_2}, ..., DMQ_{i_n} operating on the same dataset and having the same time constraints as DMQ_1 and DMQ_2, such that $DMQ_{i_1} = DMQ_1$ and $DMQ_{i_n} = DMQ_2$, and for $j = 2..n$ one of the following conditions holds:

1. pattern constraints of $DMQ_{i_{j-1}}$ have a form of a conjunction of n basic Boolean pattern predicates, pattern constraints of DMQ_{i_j} have a form of a conjunction of $n+1$ basic Boolean pattern predicates ($n \geq 0$), and each basic Boolean pattern predicates in $DMQ_{i_{j-1}}$ also appears in DMQ_{i_j};
2. $DMQ_{i_{j-1}}$ and DMQ_{i_j} have pattern constraints p_1 and p_2 respectively, where p_1 and p_2 are conjunction of n basic Boolean pattern predicates ($n \geq 1$), $p_1 = p \wedge b_1$, $p_2 = p \wedge b_2$ (p is a conjunction of $n-1$ basic Boolean pattern predicates), and b_2 is stronger than b_1.

From the Lemmas 2 and 3 and the transitivity property of the inclusion relationship, we have DMQ_1 includes DMQ_2.

Lemma 4. *Let DMQ_1 and DMQ_2 be two sequential pattern queries, operating on the same dataset and having the same pattern constraints. Let max_1, min_1, and win_1 denote values of max-gap, min-gap, and window-size parameters of DMQ_1, and max_2, min_2, and win_2 values of max-gap, min-gap, and window-size parameters of DMQ_2. If one of the following conditions holds:*

1. *$max_2 < max_1$ and $min_2 = min_1$ and $win_2 = win_1$,*
2. *$min_2 > min_1$ and $max_2 = max_1$ and $win_2 = win_1$,*
3. *$win_2 < win_1$ and $max_2 = max_1$ and $min_2 = min_1$*

then DMQ_1 dominates DMQ_2.

Theorem 2. *Let DMQ_1 and DMQ_2 be two sequential pattern queries, operating on the same dataset and having the same pattern constraints. If DMQ_2 extends time constraints of DMQ_1, then DMQ_1 dominates DMQ_2.*

Proof. Let max_1, min_1, and win_1 denote values of max-gap, min-gap, and window-size parameters of DMQ_1, and max_2, min_2, and win_2 values of max-gap, min-gap, and window-size parameters of DMQ_2. Since DMQ_2 extends time constraints of DMQ_1, we have: $win_2 \leq win_1$, $max_2 \leq max_1$ and $min_2 \geq min_1$. Let DMQ_3 and DMQ_4 be sequential pattern queries operating on the same dataset and having the same pattern constraints as DMQ_1 and DMQ_2, Let the values of max-gap, min-gap, and window-size parameters be max_2, min_1, and win_1 in case of DMQ_3, and max_2, min_2, and win_1 in case of DMQ_4. Thus, from the Lemma 4, DMQ_1 dominates DMQ_3, DMQ_3 dominates DMQ_4, and DMQ_4 dominates DMQ_2 (in fact, in each of the three cases equivalence is possible but equivalence is a particular case of dominance). Since the dominance relationship is transitive, DMQ_1 dominates DMQ_2.

Theorem 3. *Let DMQ_1 and DMQ_2 be two sequential pattern queries, operating on the same dataset. If DMQ_2 extends pattern constraints of DMQ_1 and DMQ_2 extends time constraints of DMQ_1, then DMQ_1 dominates DMQ_2.*

Proof. Let DMQ_3 be a sequential pattern query operating on the same dataset as DMQ_1 and DMQ_2, having pattern constraints of DMQ_1 and time constraints of DMQ_2. Thus, DMQ_2 extends pattern constraints of DMQ_3 and DMQ_3 extends time constraints of DMQ_1. From the Theorems 1 and 2 we have: DMQ_1 dominates DMQ_3 and DMQ_3 includes DMQ_2. Since inclusion is a particular case of dominance and the dominance relationship is transitive, DMQ_1 dominates DMQ_2.

4 Algorithms for Efficient Sequential Pattern Query Processing in the Presence of Materialized Results of Previous Queries

Given a sequential pattern query DMQ and materialized results of a sequential pattern query DMQ_V, in the general case, even if DMQ_V and DMQ operate on the same dataset but differ in pattern and time constraints, it is not possible to answer DMQ without running a sequential pattern mining algorithm. However, there are four particular cases where DMQ can be answered efficiently using the materialized results of DMQ_V since they correspond to equivalence, inclusion, and dominance relationships between DMQ_V and DMQ. These cases are listed below:

1. If DMQ_V and DMQ have the same pattern and time constraints, then the results of DMQ are equal to the results of DMQ_V (the two queries are equivalent since they are identical);
2. If DMQ_V and DMQ have the same time constraints and DMQ extends pattern constraints of DMQ_V, then DMQ can be answered by filtering out the patterns returned by DMQ_V not satisfying pattern constraints of DMQ (DMQ_V includes DMQ according to the Theorem 1);
3. If DMQ_V and DMQ have the same pattern constraints and DMQ extends time constraints of DMQ_V, then DMQ can be answered by evaluating the support of the patterns returned by DMQ_V using the time constraints of DMQ, and filtering out patterns not satisfying the minimum support threshold of DMQ. (DMQ_V dominates DMQ according to the Theorem 2);

4. If DMQ extends pattern constraints of DMQ_V and DMQ extends time constraints of DMQ_V, then DMQ can be answered by evaluating the support of the patterns returned by DMQ_V using the time constraints of DMQ, and filtering out patterns not satisfying the pattern constraints of DMQ. (DMQ_V dominates DMQ according to the Theorem 3).

Answering the query in the first case (the case of equivalence) is trivial, therefore we concentrate on details concerning inclusion and dominance relationships.

For the second case we propose an algorithm that performs one sequential scan of the materialized patterns, processing one pattern at a time (main memory requirements are minimal). Each pattern is tested if it satisfies these basic Boolean pattern predicates from the pattern constraints of DMQ that were not in DMQ_V. All the basic Boolean pattern predicates of DMQ that were in DMQ_V must be satisfied by all the materialized patterns since pattern constraints in our model have the form of a conjunction of basic predicates. The algorithm for the second case is presented below.

Algorithm 1 Answering a sequential pattern query in case of inclusion due to extending pattern constraints (Result Filtering)
Input: A sequential pattern query issued by a user (DMQ) and results of a sequential pattern query DMQ_V including DMQ.
Output: The results of DMQ.
Method:

```
begin
    Answer = results of DMQ_V;
    for each p ∈ results of DMQ_V do
    begin
        for each basic Boolean pattern predicate b such that
            b is in pattern constraints of DMQ and
            b is not in pattern constraints of DMQ_V do
            begin
                if not (p satisfies b) then
                    Answer = Answer \ {p};
                    break;
                end if;
            end;
        end;
    output Answer;
end.
```

For the third and fourth cases we propose one uniform algorithm (both cases result in the dominance relationship). Conceptually, the algorithm has to scan the source dataset once in order to re-evaluate the support of materialized patterns and then prune the patterns that do not satisfy pattern constraints of DMQ. However, for the fourth case, we apply one optimization to reduce the cost of the support re-evaluation phase that is proportional to the number of patterns to be verified. Before scanning the source dataset, we filter out patterns that do not satisfy pattern constraints of DMQ using Algorithm 1. After the scan of the dataset, we only test the predicate representing the minimum support threshold (the only one that for a given pattern could by true before the support re-evaluation, and false after that operation). The effects of this optimization will be discussed in the next section.

During the support re-evaluation phase, when testing whether a currently processed data-sequence contains a given pattern, all time constraints of DMQ have to be taken into account, even if only one of them has been restricted compared to DMQ_V. This is motivated by the observation that a given pattern may occur several times in a given data-sequence. As a result, if we checked only one of the time constraints, we might find a different occurrence satisfying the constraint than the occurrence previously found as valid with respect to the other two time constraints.

The algorithm in the form presented below assumes that the set of materialized patterns supporting pattern constraints of DMQ fits into main memory. If this is not the case, the set of materialized patterns has to be partitioned into portions that fit into main memory and the algorithm has to be run on each of the partitions.

Algorithm 2 Answering a sequential pattern query in case of dominance due to extending time constraints (Result Verification)
Input: A sequential pattern query issued by a user (DMQ), a collection of data-sequences D, and results of a sequential pattern query DMQ_V dominating DMQ.
Output: The results of DMQ.
Method:

```
begin
    if DMQ extends pattern constraints of DMQ_V then
        Answer = patterns in results of DMQ_V satisfying
        pattern constraints of DMQ; /* Algorithm 1 */
    else Answer = results of DMQ_V;
    end if;
    scan D once evaluating the support of patterns
    in Answer using time constraints of DMQ;
    for each p ∈ Answer do
    begin
        if p exceeds the minimum support threshold of DMQ
        then output p; end if;
    end;
end.
```

Having provided sequential pattern query processing algorithms for the cases leading to equivalence, inclusion and dominance relationships, we have to address situations where for a given query issued by a user (DMQ), there are many materialized query results that could be used to answer the query without running a complete data mining algorithm. In general, the set of applicable materialized query results consists of results of queries equivalent to DMQ, including DMQ, and dominating DMQ. It is clear that in the first place the data mining system should look for a query identical to DMQ (the case of equivalence) since in that case the results of DMQ are directly available. Then, the system should look for query results that could be used by Algorithm 1 (returned by a query DMQ_V having the same time constraints as DMQ, such that DMQ extends pattern constraints of DMQ_V). If no query satisfying the above criteria could be found, the system should try to find query results that could be used by Algorithm 2 (returned by a query DMQ_V, such that DMQ extends time constraints of DMQ_V and either DMQ_V and DMQ have the same pattern constraints or DMQ extends pattern constraints of DMQ_V). Finally, if again no appropriate query criteria could be found, a complete data mining algorithm has to be run.

We believe that in majority of cases Algorithm 1 will be more efficient than Algorithm 2 since the former requires one scan of the pattern set and no scan of the source dataset, while the latter scans the source dataset once and during this scan for each data-sequence processes all the patterns. However, it has to be noted that in certain cases application of Algorithm 2 may be more efficient than application of Algorithm 1 (for example, if the source dataset and the materialized set of patterns to be used by Algorithm 2 are extremely small, whereas the materialized pattern set to be used by Algorithm 1 is huge).

The final issue that has to be addressed is the selection of the materialized query results to be used by Algorithms 1 and 2 if there is more than one query including or dominating the query to be answered. We observe that it is not possible to provide selection criteria always leading to the minimal processing time, because the processing time depends not only on the syntax of the queries but also on the contents of the source dataset. Therefore, we decide to optimize the space requirements by choosing the materialized pattern set of the smallest size. We believe that this solution will also lead to minimal processing time in many situations, since smaller size of the pattern set leads to the smaller number or size of patterns that have to be filtered or verified against the database. It is not guaranteed, however, since the processing time is affected also by the number of predicates that have to be tested for each pattern, which depends on the pattern structure (subsequent predicates are tested until one of them is found to be false).

Example 2. Let us consider the following three queries discovering sequential patterns from the same dataset:

DMQ_1 = {max-gap: 100, min-gap: 0, window-size: 1, π(**SPG**, 0.2, pattern)}
DMQ_2 = {max-gap: 100, min-gap: 7, window-size: 1, π(**SPG**, 0.1, pattern) $\wedge$ π(**SG**, 3, pattern)}
DMQ = {max-gap: 100, min-gap: 7, window-size: 1, π(**SPG**, 0.2, pattern) $\wedge$ π(**SG**, 3, pattern)}

Let us assume that results of DMQ_1 and DMQ_2 are stored in cache, and DMQ is the query to be answered. Since neither DMQ_1 nor DMQ_2 is identical to

DMQ, the data mining system would choose to answer DMQ using Algorithm 1 exploiting cached results of DMQ_2 (returning those patterns from the results of DMQ_2 that exceed the minimum support threshold of 0.2). If results of DMQ_2 were not available, the system would answer DMQ using Algorithm 2 exploiting the results of DMQ_1 (selecting patterns from the results of DMQ_1 whose size is greater than 3, re-evaluating their support in one scan of the source dataset using max-gap of 100, min-gap of 7, and window-size of 1, and returning those patterns that exceed the minimum support threshold of 0.2 after support re-evaluation).

5 Experimental Results

In order to evaluate performance gains offered by our sequential pattern query processing algorithms, we performed several experiments on a synthetic dataset generated by means of the *GEN* generator from the *Quest* project [1]. We treated transaction identifiers generated by *GEN* as transaction times. Thus, the time gap between two adjacent elements of each data-sequence was always equal to one time unit. The dataset used in the experiments consisted of 1000 data-sequences. *GEN* parameter values were chosen so that for the minimum support thresholds used in queries there were a reasonable number of sequential patterns varying in size and length to be discovered.

In the first step we materialized the results of the query discovering all sequential patterns whose support was above 0.5% using max-gap of 1000, min-gap of 0, and window-size of 1. The materialized set of patterns consisted of about 3500 sequential patterns. Next, we tested several queries adding additional pattern constraints (concerning pattern support, size, length, or contents) and restricting time constraints. For each query, we compared execution times of our algorithms exploiting materialized patterns and the *GSP* algorithm with the post-processing pattern filtering phase. For the queries included by the materialized query, Algorithm 1 was on average more than 400 times faster than *GSP*. For the queries dominated by the materialized query, Algorithm 2 was used, and its processing time was on average more than 100 times shorter than in case of *GSP*. We also tested the effects of our optimization used in case of queries extending both pattern and time constraints of the materialized query (filtering out patterns that do not satisfy pattern constraints before re-evaluating the support of materialized patterns). Experiments show that the optimization reduces processing time by about 33%.

6 Concluding Remarks

We proved experimentally that our sequential pattern query processing schemes can reduce processing time by several orders of magnitude when materialized results of previous queries are available. However, theoretically it is possible to imagine situations, where a complete mining algorithm could be more efficient than our techniques. While we believe that in typical situations our methods should outperform mining algorithms, in the future we plan to focus on cost-based optimization of sequential pattern queries (and data mining queries in general) using certain statistics of the source dataset in order to choose an optimal query execution plan.

In the paper, we did not discuss cache management schemes, which could certainly influence the overall performance of the system. We believe that general purpose cache management algorithms could be used, possibly with simple optimizations such as removing included or dominated queries first, and not materializing results of queries equivalent to queries whose results are already in cache.

References

1. Agrawal R., Mehta M., Shafer J., Srikant R., Arning A., Bollinger T.: The Quest Data Mining System. Proc. of the 2nd KDD Conference (1996)
2. Agrawal R., Srikant R.: Mining Sequential Patterns. Proc. of the 11th ICDE Conf. (1995)
3. Baralis E., Psaila G.: Incremental Refinement of Mining Queries. Proc. of the 1st DaWaK Conference (1999)
4. Han J., Lakshmanan L., Ng R.: Constraint-Based Multidimensional Data Mining. IEEE Computer, Vol. 32, No. 8 (1999)
5. Han J., Pei J., Mortazavi-Asl B., Chen Q., Dayal U., Hsu M-C.: FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining. Proc. of the 6th KDD Conference (2000)
6. Imielinski T., Mannila H.: A Database Perspective on Knowledge Discovery. Communications of the ACM, Vol. 39, No. 11 (1996)
7. Nag B., Deshpande P.M., DeWitt D.J.: Using a Knowledge Cache for Interactive Discovery of Association Rules. Proc. of the 5th KDD Conference (1999)
8. Parthasarathy S., Zaki M.J., Ogihara M., Dwarkadas S.: Incremental and Interactive Sequence Mining. Proc. of the 8th CIKM Conference (1999)
9. Pei J., Han J., Mortazavi-Asl B., Pinto H., Chen Q., Dayal U., Hsu M-C.: PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. Proc. of the 17th ICDE Conference (2001)
10. Srikant R., Agrawal R.: Mining Sequential Patterns: Generalizations and Performance Improvements. Proc. of the 5th EDBT Conference (1996)