Plan Mining by Divide-and-Conquer*

Jiawei Han, Qiang Yang, and Edward Kim
School of Computing Science, Simon Fraser University
Burnaby, BC Canada V5A 1S6
E-mail: {han, qyang, kime}@cs.sfu.ca

Abstract

Plans or sequences of actions are an important form of data. With the proliferation of database technology, plan databases (or planbases) are increasingly common. Efficient discovery of important patterns of actions in plan databases presents a challenge to data mining. In this paper, we present a method for mining significant patterns of successful actions in a large planbase using a *divide-and-conquer* strategy. The method exploits multi-dimensional generalization of sequences of actions and extracts the inherent hierarchical structure and sequential patterns of plans at different levels of abstraction. These patterns are used in turn to subsequently narrow down the search for more specific patterns. The process is analogous to the use of divide-and-conquer methods in hierarchical planning. We illustrate our approach using a travel planning database.

1 Introduction

In recent years, data mining has achieved great success in a variety of application domains [11, 9, 5, 7]. Besides relational data, much of the world's recorded knowledge is in the form of sequences of actions, or plans, and their goals. This is especially true for the increasingly successful applications of Internet-based electronic commerce. It can therefore be expected that plan and goal mining will become an important theme in data mining research.

Plan mining is the task of discovering significant knowledge from a large database of plans. Some examples are: a) discover travel patterns of business travelers in an airline database, b) in a medical domain, uncover the effects in treatment processes, c) find out the highly critical actions that affect the success or failure of plans in relation to their goals and environmental conditions, and d) find the significant patterns that exist in the maintenance or repair plans for a mechanical device, such as a vehicle or aircraft.

* Research was supported in part by research grants from the Natural Sciences and Engineering Research Council of Canada, a grant NCE:IRIS/Precarn from the Networks of Centres of Excellence of Canada, and a B.C. Science Council GREAT Scholarship Award.

Previous work in this area can be divided into several categories. In the area of plan recognition, the objective is to uncover the intention and goal of a series of observations in actions. Examples are works by Allen [2], and Woods, et al. [13]. There is a large body of work in plan-generation on how to generate plans from action descriptions and goals [1, 14, 12]. An example is the search-control rule learning methods [4, 10]. Zaki, et al. [15] developed a technique for plan mining, which first performs a brute-force expansion of all the nodes and then mines sequence patterns using the Apriori technique [11] on the data which indicate high incidence of plan failure. They then added this knowledge to the planner so that the paths which often result in failure are avoided in the planner.

Plan mining presents several challenges for data mining. First, mining total order sequences as in [15] may not be in general feasible if there is a large number of distinct operators or a large set of alphabets used for describing a plan's actions since it may lead to very low support for each pattern as reported in [15]. In addition, this may lead to a large number of patterns of actions and a potentially very large planbase. It is thus unrealistic to apply a brute-force approach to verify the significance of *every* pattern in such a planbase.

In this work, we focus on mining successful plans that consist of sequences of actions. We are interested in mining important patterns of actions at different levels of abstraction. To account for the possibility of very large planbases and vast possibilities of potential patterns, we adopt a generalization-based *divide and conquer* approach, whereby we first abstract the planbase to a sufficiently high level, and apply various operators to obtain interesting patterns of actions with significant support. Then these high-level patterns can serve as a guide to partition the planbase and divide the mining target into subsequences that can be mined effectively and efficiently. In the remaining of the paper, we will first present an air-travel planbase to motivate our problem. We then present our algorithms and analyses.

2 Problem Outline

A plan consists of a variable sequence of actions, and a planbase is a large collection of plans. Plan mining extracts important or significant patterns from the planbase which may aid a domain expert or be used in an intelligent planner. We motivate our work using an air-travel planning example.

2.1 An Air Flight Planbase

Suppose an air-travel planbase, as shown in Table 1[1], stores customers' flight sequences, where each record corresponds to an action in a sequential database, and a sequence of records sharing the same plan number are considered as one plan with a sequence of actions. Also, *D-Loc* and *A-Loc* specify the airport code, and Table 2 stores information about airports.

P#	A#	D-Loc	D-Time	A-Loc	A-Time	A_line	...
1	1	ALB	800	JFK	900	TWA	...
1	2	JFK	1000	ORD	1230	UA	...
1	3	ORD	1300	LAX	1600	UA	...
1	4	LAX	1710	SAN	1800	DAL	...
2	1	SPI	900	ORD	950	AA	...
2	2	ORD	1100	JFK	1230	AA	...
2	3	JFK	1300	SYR	1400	CAL	...
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

Table 1: A database of travel plans: A travel planbase

A_Code	City	State	Region	A_Size	...
ORD	Chicago	Illinois	Mid-West	100000	...
CMI	Champaign	Illinois	Mid-West	10000	...
SPI	Springfield	Illinois	Mid-West	10000	...
LAX	Los Angeles	California	Pacific	80000	...
BUR	Burbank	California	Pacific	10000	...
SAN	Sandiego	California	Pacific	20000	...
JFK	New York	New York	Eastern	70000	...
ALB	Albany	New York	Eastern	20000	...
SYR	Syracuse	New York	Eastern	10000	...
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

Table 2: An airport information table

There could be many patterns minable from a planbase like Table 1. For example, one may find that most flights from the cities in the eastern United States to mid-western cities have a stop over in ORD, which could be due to the fact that ORD is the major hub for several major airlines. Notice that the information about airline hubs like, Los Angeles (LAX), Chicago (ORD), and New York (JFK) can be easily derived from Table 2, based on the airport size.

However, there may be hundreds of major hubs in a travel database. Indiscriminate mining of such kind of "rules" may result in a large number of scattered rules without substantial support but still missing a clear overall picture. Obviously, this is not our intention. What we would like to find is a small number of general (sequential) patterns which may cover a substantial portion if not most of the plans, and then we can divide our search efforts based on such mined sequences.

[1] In all the tables, P# for Plan#, A# for Action#, D-Loc for Dep-Loc, D-Time for Dep-Time, A-Loc for Arr-Loc, A-Time for Arr-Time, A_line for Airline, A_Code for Airport code, and A_Size for Airport-Size.

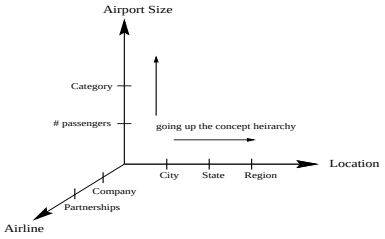

Figure 1: A multi-dimensional view of a database

The key to mine such very general (sequential) patterns is to generalize the plans in our planbase to a sufficiently high level. A multi-dimensional database model [3], such as the one shown in Figure 1 for the air-flight planbase, can be used to facilitate such plan generalization. Notice that low level information may never share enough commonality to form succinct plans. With multidimensional model, one may generalize the planbase in different directions and observe when the generalized plans may share some common, interesting, sequential patterns with substantial support. Such shared generalized patterns will lead to the derivation of high level, concise plans.

2.2 Plan Mining by Divide-and-Conquer

Let's examine the air-travel plan mining in more detail.

Example 2.1 We examine how to derive generalized plans from the detail plans in the planbase. The actual plans may appear as follows.

Springfield - Chicago - New York - Albany

San Diego - Los Angeles - Denver - Atlanta - Savannah

These plans may look very different and they can be generalized in many different ways. However, when the routes are generalized based on the airport size dimension, one may observe some interesting sequential patterns like **S-L-L-S**, where L represents a large airport (hub), and S represents a relatively small regional airport. The generalization of a large number of air travel plans may lead to a rather general but highly regular pattern in the form of **[S]-L^+-[S]**, where L^+ indicates one or more repetition of L, and the square brackets [] means an optional step.

In planning, the pattern of flying to a major hub then to another major hub before reaching the final destination, is analogous to achieving *plan subgoals*. A major hub would correspond to a common subgoal in a plan space. Subgoal identification is an important problem in planning [8, 14]. Once mined, this knowledge could

be used in some planners to reduce the search space and improve plan efficiency.

In our approach, the common high-level sequences can be mined using a sequential pattern mining technique, which may involve operations like "merge", "option (collapsing)", etc. After the significant sequential pattern is identified, the pattern can be used to partition the planbase and then mine each partitioned planbase to find common characteristics. For example, the following characteristics can be mined from a partitioned planbase.

if Flight(x, y) & AirportSize(x, S) & AirportSize(y, L)
then Region(x) = Region(y) [conf]

where *conf* (confidence) is a commonly used rule strength measure [11]. This rule says for a flight from a small airport x to a large airport y, there is a *conf* probability that x and y belong to the same region. □

3 Plan Mining Methods

Now we examine how to derive multi-dimensional, generalization-based plan mining algorithms. Similar to our example, we assume there are two tables: a *planbase* table and a *dimension* table. A multi-dimensional data model, such as star model [3], can be used to link the two tables via some key components. Our divide and conquer method will combine sequential pattern mining with generalization to produce significant patterns of interest. Divide and conquer implies high level patterns are first obtained and the derived patterns are then used to guide the search for patterns in the partitioned planbase.

3.1 Generalization-based overall sequential pattern mining

The first task of generalization-based divide-and-conquer mining is to explore multidimensional generalization in order to find high level, highly regular sequential patterns. The Attribute-Oriented Induction (AOI) method developed in [6] can be used to generalize the planbase according to the concept hierarchy provided in the *dimension* table. This is similar to the rollup operation in OLAP [3], however, AOI works on a plan database which consists of sequences of actions; whereas the rollup in OLAP works only in a highly structured multidimensional database. Notice that the generalization is performed in multiple directions (dimensions) and at multiple levels in order to catch highly regular sequential patterns when it occurs. However, this does not mean that a user needs to observe and control every step of multi-dimensional generalization. An attribute threshold control method proposed in [6] can be used to estimate the levels to "jump": When there exist a large number of distinct values in an attribute, further generalization is needed in order to generate short patterns. Therefore, the system can calculate the minimum levels to be reached directly in order to produce promising patterns to avoid wasting time to perform detailed generalization level-by-level. For example, as we will see in Example 3.1, it is promising to generalize the location sequence to size-sequence which contains only two distinct values: (*Large* and *Small*) rather than region-sequence which contains many distinct values.

Algorithm 3.1 (Overall sequential pattern mining)

Input: A planbase and its associated dimension table(s), as shown in the Problem Outline section.

Output: The overall, generalized sequential pattern(s) of the planbase.

Method:

1. Transform the planbase into a sequence planbase (called *seq-planbase*) by grouping together the actions in the same plan according to its sequence number.
2. Based on the data in the associated dimension table of the planbase, estimate the number of distinct values which may be generalized from the seq-planbase, and determine whether it is useful to examine the generalized plan at a particular level of abstraction.
3. Generalize the seq-planbase in multiple directions (but only to sufficiently high levels) in the similar manner as Attribute-Oriented Induction (AOI) [6].
4. Apply a *merge* operator to merge the consecutive identical symbols (each such symbol represents a particular action), which yields a sequence of actions noted with the $^+$ notation.
5. Calculate the support of each such sequence in the generalized planbase. Retain the sequential patterns whose support is no less than a predefined minimum support threshold, $overall\_pattern\_min\_support$ (which could be defined and/or adjusted by an expert or a user).
6. Further merge the sequential patterns derived above by applying the option operator [], and output the merged patterns.

Complexity analysis. For a sequential table with N tuples, creating the transformed sequence table, with K distinct plans (i.e., $K < N$), is a linear operation, $O(N)$. The complexity of the AOI algorithm [6] for the seq-planbase of size K, is $O(KlogK)$. The $merge()$ and $optional()$ operations are linear time operations on the number of plans, $O(K)$. If the sequence length is on average c, then $N = cK$, and $O(K) = O(N)$. Hence, the total complexity of the algorithm is $O(NlogN)$. □

Example 3.1 (Overall sequential pattern mining) Based on Algorithm 3.1, the planbase of Table 1 is first transformed into the airport-location-sequence table, as shown in the first (*Plan#*) and the second (*Loc_Seq*) columns of Table 3. Then generalization is performed on the transformed sequence table as follows.

First, based on the data in the associated dimension table of the planbase, we can calculate the number of distinct values which may be generalized from the planbase, and determine whether it is useful to examine the generalized plan at a particular level of abstraction. For example, it is useless to examine the merge of plans at the airport code level, city level, precise departure/arrival time level, etc.

We may consider the generalization of airports to Large (L) and Small (S), which contains only two distinct symbols in the (airport) size-sequence column. Alternatively, we may consider to generalize the sequence of airports to state or region level, or air-line level. However, such cases will result in a much larger set of distinct symbols and are in general nonpromising. Suppose we do generalize the sequence planbase to these levels. We will derive the third to fifth columns of Table 3. Notice that an estimation of the number of distinct values at an abstraction level will save the generation of some nonpromising generalized attributes, such as the fourth to fifth columns of Table 3.

P#	Loc-Seq	Size-Seq	State-Seq	Region-Seq	···
1	ALB-JFK-ORD-LAX-SAN	**S-L-L-L-S**	N-N-I-C-C	E-E-M-P-P	···
2	SPI-ORD-JFK-SYR	**S-L-L-S**	I-I-N-N	M-M-E-E	···
⋮	⋮	⋮	⋮	⋮	⋮

Table 3: Muti-dimensional generalization of planbase

Plan#	Size-Seq	State-Seq	Region-Seq	···
1	**S**-L^+-**S**	N^+-I-C^+	E^+-M-P^+	···
2	**S**-L^+-**S**	I^+-N^+	M^+-E^+	···
⋮	⋮	⋮	⋮	⋮

Table 4: Merging consecutive, identical actions in plans

After generation of the generalized sequence planbase, a *merge* operator can be applied to the table to merge (and collapse) the consecutive identical symbols, which yields a sequence of actions with the + notation, as shown in Table 4. This table is then used to calculate the sequence in the generalized planbase whose support is no less than a predefined minimum support threshold, *overall_pattern_min_support*. Suppose the *overall_pattern_min_support* threshold is 8%. One can easily see that only airport-size-sequence generalization may survive the threshold test which may lead to the generalized sequence plan table as Table 5.

Size-Seq	support
S-L^+-**S**	20%
L^+	25%
L^+-**S**	9%
⋮	⋮

Table 5: Derivation of substantial plan sequences

Notice that the merge operation can be done simultaneously when constructing the abstraction sequence table or when counting the support in the derivation of substantial plan sequences. We show it separately here for explanation purposes.

Finally, the sequential patterns derived above can be further merged together by applying the option operator []. This may finally lead to only one sequential pattern, such as,

Size-Seq	support
[**S**]-L^+-[**S**]	98.5%

This pattern means although 98.5% of travel plans have the pattern [**S**]-L^+-[**S**], there are still 1.5% of travel plans which may have patterns like **S-S**, **S-S-L**, **L-S-L**, etc. □

3.2 Conquered Mining: Mining Characteristics in Partitioned Planbase

Once the overall plan sequence is generated, the sequence can be broken down into several subsequences, and the planbase can be partitioned accordingly. Then mining for common characteristics can be performed on each partitioned planbase and for each subsequence to find interesting common features in the partitioned planbase. This leads to the following algorithm.

Algorithm 3.2 (Mining partitioned planbase)

Input: The planbase and the generalized sequence pattern(s) output from Algorithm 3.1.

Output: Characteristic rules for each partitioned planbase.

Method:

1. For each mined sequence with sufficient support, partition the sequence into *transitional and repetitive subsequences*. As their names suggest, *transitional* are the subsequences whose adjacent symbols change, and *repetitive* consists of only repeated symbols.
2. Use each subpattern to perform a selection (partition) on the sequence planbase.
3. Mine characteristic properties for each subsequence pattern on each corresponding partitioned planbase.

This may involve the application of some special operators, which could be either built-in or user-defined, domain-specific operators. One useful operator is the equivalence testing operator, *equals*(), which tests whether the corresponding values are the same.

Collect count or other statistical information in the partitioned planbase, and the rules with high support are presented as subsequence characteristic rules.

Analysis. For each dimension, there may be a few high level patterns mined from Algorithm 3.1. Assume there are P high level patterns mined, and on average there may be r subsequences in each pattern. Since we perform a selection on each subsequence, the overall cost is $O(rPN)$. □

Example 3.2 (Mining partitioned planbase) We continue our running example of mining the travel planbase. According to Algorithm 3.2, the sequence **[S]-L^+-S** can be partitioned into three subsequences: two transitional subsequences, **S-L** and **L-S**, and one repetitive subsequence, L^+. These subsequences can be used to select the corresponding sequence planbase and then mine each corresponding partitioned sequence planbase.

The selection with the subsequence pattern **[S]-L** on the sequence planbase results in all the first hop flights from a small airport to a big one for those flight plans which follow the **[S]-L^+-S** model. Then generalization and test operators can be applied to such partitioned planbase to discover interesting characteristics. For example, the *equals*() operator can be applied to test the generalized attributes.

The equivalence testing operator is a powerful operator to find interesting common behavior in a large planbase. For example, in the mining subsequence pattern **S-L**, when generalizing airport to state, we may still have a large number of distinct states in the partitioned planbase. However, one may find most of the **S-L** subsequence share the same state in the departure and destination airports. That is, although the support for both departure and arrival airports are in Illinois is not high, the support for both departure and arrival airports are in the same state is rather high.

Similarly, the Region-Seq column also has high equality support for the pattern **S-L**. Since the Region-Seq is one level higher than the State-Seq in the Location concept hierarchy, there is no need to check every tuple in Region-Seq column, based on the *monotonicity* property. We just need to check the tuples which are not equal in State-Seq to find those **S-L** patterns which are not in the same State but now in the same Region. It is obvious the support will increase in this case.

For the S-L subsequence, two characteristic rules could be derived.

if Flight(x y) & AirportSize(x, S) & AirportSize(y, L) then State(x) = State(y) [60%]

if Flight(x, y) & AirportSize(x, S) & AirportSize(y, L) then Region(x) = Region(y) [78%]

It implies for traveling from a small airport x to a large airport y, there is 60% probability they are in the same state, and there is 78% probability they are in the same region.

Similarly, one can mine rules for the subsequence L^+. However, in this case, if we use the equivalence testing operator *equals*() to test the State-Seq column, there is a high probability to be false. That is, one may derive the following rule,

if Flight(x y) & AirportSize(x, L) & AirportSize(y, L) then State(x) != State(y) [80%]

It implies when flying from a large airport to another large airport, there is 80% probability that they are not in the same state. □

4 Discussion and Conclusion

We have introduced a generalization-based divide-and-conquer technique for mining plan database. We show that interesting algorithms can be developed for efficient and effective mining large planbases. The effectiveness is based on mining plan sequences at multiple levels of abstraction and at each corresponding subsequence. The efficiency is based on estimating the expected levels of abstraction before plan generalization and pattern search, and also based on the divide-and-conquer technique.

Although only the air flight planbase is used here as an example, the generalization-based divide-and-conquer plan mining method works in many other applications. For example, for Weblog mining, one may study the Web access patterns to identify popular web portals and common paths. This may lead to the derivation of valuable information for marketing and identifying user preferences in e-commerce.

Here we discuss several issues which may need further clarification or involve future work.

1. Determination of the levels of generalization: In our plan mining, the sequence generalization process is automated, and the levels of generalization is mainly determined by the number of distinct values of the sequence elements. If there are too many distinct values (above an expert-predefined, user-adjustable *attribute generalization threshold*), further generalization is needed in order to find promising sequence patterns with sufficient support. Alternatively, one

may use a *minimum support threshold* similar to association rule mining to confine that a pattern covers a sufficient number of cases. However, the effects and the principles of generalization control between the two approaches should be rather similar.

2. Applications of complex operators in plan mining: Currently, our subsequence characterization applies only a subset of fairly simple operators. More commonly used operators, such as *less_than()* instead of just *equals()*, can be used to generate more patterns. Moreover, complex relationships may be found without much added cost. For instance, associations may be drawn from subsequences.

3. In comparison with association-based sequential pattern mining: There have been studies on association-based sequential pattern mining [11]. The situation of plan mining is rather different from it: The former is to mine sequential patterns with sufficient but quite small support and discovers many patterns without generalization; where the latter (plan mining) is to seek for a small set but highly generalized patterns which cover substantial portions of the plans in the plan base. Moreover, the former filters out most of the elements in a sequence; whereas the latter considers most of the elements in the sequence since each action often corresponds to one step of a plan although there could be some nonessential actions. However, it is interesting to further study how to benefit from the existing sequential pattern mining work in plan mining.

4. Mining sequence patterns involving multidimensional attributes: In the example, we discussed mining sequence patterns involving *single dimension* only (such as size-sequence, region-sequence, and so on). Multi-dimensional sequence patterns can be mined using the same algorithm. For example, a sequence pattern may involve both airport-size and continent, such as **[S&NA]-**$(L\&NA)^+$**-(L&EU)** which indicates that a traveller flies within North America one or more times (starting possibly with a small airport) and then finally reaches a large airport in Europe. Such dimension combined mining also needs the generalization of each dimension to a high level before examining the combined sequence patterns.

5. Multi-stage divide-and-conquer: With more complex plan structures, a recursive or multi-stage divide-and-conquer technique can be developed to improve mining efficiency and discover hierarchical patterns.

In summary, our study shows that generalization-based, multi-dimensional mining of planbase based on a *divide and conquer strategy* may lead to the discovery of interesting, high-level concise sequence of plans, and each subsequence can be further partitioned based on mined patterns to discover their corresponding characteristics. Moreover, several interesting operators, such as $merge(), option()$, and $equals()$ are introduced to help the discovery of common patterns in the mining process. We are implementing our plan mining technique by constructing a PlanMiner. In the future, we seek to explore further optimization techniques that allow fast sequential mining and sophisticated characterizations in large planbases.

References

[1] J. F. Allen, J. Hendler, and A. Tate. *Readings in Planning*. Morgan Kaufmann Publishers, 1990.

[2] J. F. Allen, H. A. Kautz, R. N. Pelavin, and J. D. Tenenberg. *Reasoning about Plans*. Morgan Kaufmann Publishers, 1991.

[3] S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, 26:65–74, 1997.

[4] O. Etzioni. Acquiring search-control knowledge via static analysis. *Artificial Intelligence*, 62(2):225–302, 1993.

[5] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (eds.). *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.

[6] J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. *IEEE Trans. Knowledge and Data Engineering*, 5:29–40, 1993.

[7] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. in preparation, 1999.

[8] R. Korf. Planning as search: A quantitative approach. *Artificial Intelligence*, 33:65–88, 1985.

[9] H. Mannila, H Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences. In *Proc. 1st Int. Conf. Knowledge Discovery and Data Mining*, pp. 210–215, Montreal, Canada, Aug. 1995.

[10] S. Minton, J. G. Carbonell, C. A. Knoblock, D. R. Kuokka, O. Etzioni, and Y. Gil. Explanation-based learning: a problem solving perspective. *Artificial Intelligence*, 40:63–118, 1989.

[11] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proc. 5th Int. Conf. Extending Database Technology (EDBT)*, pp. 3–17, Avignon, France, March 1996.

[12] D. Weld. An introduction to least-commitment planning. *AI Magazine*, Winter 1994:27–61, 1994.

[13] S. Woods, A. Quilici, and Q. Yang. *Constraint-based Design Recovery for Software Reengineering: Theory and Experiments*. Kluwer, 1997.

[14] Q. Yang. *Intelligent Planning — A Decomposition and Abstraction Based Approach*. Springer-Verlag, 1997.

[15] M. J. Zaki, N. Lesh, and M. Ogihara. PLANMINE: Sequence mining for plan failures. In *Proc. 4th Int. Conf. Knowledge Discovery and Data Mining (KDD'98)*, New York, NY, August 1998.