Mining Itemset-Based Distinguishing Sequential Patterns with Gap Constraint

Hao Yang[1], Lei Duan[1,2(✉)], Guozhu Dong[3], Jyrki Nummenmaa[4], Changjie Tang[1], and Xiaosong Li[2]

[1] School of Computer Science, Sichuan University, Chengdu, China
hyang.cn@outlook.com, {leiduan,cjtang}@scu.edu.cn
[2] West China School of Public Health, Sichuan University, Chengdu, China
lixiaosong1101@126.com
[3] Department of Computer Science and Engineering, Wright State University, Dayton, USA
guozhu.dong@wright.edu
[4] School of Information Sciences, University of Tampere, Tampere, Finland
jyrki.nummenmaa@uta.fi

Abstract. Mining contrast sequential patterns, which are sequential patterns that characterize a given sequence class and distinguish that class from another given sequence class, has a wide range of applications including medical informatics, computational finance and consumer behavior analysis. In previous studies on contrast sequential pattern mining, each element in a sequence is a single item or symbol. This paper considers a more general case where each element in a sequence is a set of items. The associated contrast sequential patterns will be called itemset-based distinguishing sequential patterns (itemset-DSP). After discussing the challenges on mining itemset-DSP, we present iDSP-Miner, a mining method with various pruning techniques, for mining itemset-DSPs that satisfy given support and gap constraint. In this study, we also propose a concise border-like representation (with exclusive bounds) for sets of similar itemset-DSPs and use that representation to improve efficiency of our proposed algorithm. Our empirical study using both real data and synthetic data demonstrates that iDSP-Miner is effective and efficient.

Keywords: Itemset · Sequential pattern · Contrast mining

1 Introduction

Imagine you are a supermarket manager facing a collection of customers' shopping records, each of which is a sequence of all purchases by a customer over a fixed time period. (See Table 1 for illustration.) To provide specialized service

This work was supported in part by NSFC 61103042, SKLSE2012-09-32, and China Postdoctoral Science Foundation 2014M552371. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

M. Renz et al. (Eds.): DASFAA 2015, Part I, LNCS 9049, pp. 39–54, 2015.
DOI: 10.1007/978-3-319-18120-2_3

Table 1. A toy dataset of shopping records of married and unmarried customers

ID	Shopping records	Married
S1	<{*bread, milk*} {*milk, towel*} {*coffee, beef, cola*} {*lipstick*}>	Yes
S2	<{*bread, perfume*} {*book*} {*coffee, beef, cola*} {*lipstick*} {*milk*}>	
S3	<{*towel, bread, perfume, beef, book*} {*coffee, beef, cola*} {*book*} {*milk*}>	
S4	<{*bread, perfume*} {*coffee, beef, cola*} {*lipstick, shaver*} {*milk*}>	
S5	<{*towel, bread*} {*bread*} {*cola, shaver*} {*coffee, beef, cola*} {*milk*}>	No
S6	<{*bread*} {*book*} {*milk, shaver*} {*cola*} {*towel, book*}>	
S7	<{*milk*} {*book, bread*} {*milk, shaver*} {*coffee, beef, cola*}>	
S8	<{*bread, cola*} {*coffee*} {*cola*} {*lipstick, cola*} {*milk, cola*}>	

to married customers, you may want to find and utilize informative differences between the married and unmarried customers on their shopping preferences.

The above motivation scenario cannot be addressed well using existing sequential pattern mining [1] or contrast data mining [2] methods, and thus suggests a novel data mining problem. In a sequential dataset of two classes for this scenario, each sequence is an ordered list of itemsets; given a target class, we want to find the sequential patterns that are frequent in the target class but infrequent in the other class. We call such a pattern an *itemset-based distinguishing sequential pattern (itemset-DSP)* since each of its elements is an itemset instead of a single item. Itemset-DSP mining is an interesting problem with many useful applications. As another example in addition to the shopping application given above, when an analyst in a pharmaceutical company is investigating the effect of a new drug, she may record the symptoms of patients once every 12 hours after taking the drug over one week, then compare the observed data with similarly observed data of patients not taking the drug.

While there are many existing studies on distinguishing sequential pattern mining, they focus on distinguishing sequential patterns whose elements are single items. The itemset-DSP mining problem addressed here is different. It focuses on mining patterns from sequences whose elements are itemsets. Moreover, there is a serious need to represent the patterns concisely, to avoid combinative explosion. Due to these key differences, the potential application and the mining methods of this mining problem differ significantly from those for the case of single item based sequences. We will review the related work and explain the differences systematically in Section 3.

To tackle the problem of mining itemset-DSPs, we need to address several technical challenges. First, a brute-force method, which enumerates every non-empty itemset to generate candidate elements for sequence patterns is very costly on sequence sets with a large number of distinct items and a large maximum number of items in an element. We need an efficient method to avoid generating useless candidates.

Second, we need to have a concise yet complete way to represent sequential patterns satisfying the support thresholds, so that the number of discovered patterns can be reduced, the mining results are more comprehensible, and the algorithm can be efficient.

Third, we also need to find effective techniques to efficiently discover the itemset-DSPs with concise representations. This issue is also complicated because we also consider gap constraint in the discovery of itemset-DSPs.

This paper makes the following main contributions on mining minimal itemset-DSPs with gap constraint: (1) introducing a novel data mining problem of itemset-DSP mining; (2) presenting an efficient algorithm for discovering itemset-DSPs and presenting a concise representation for the mining results; (3) conducting extensive experiments on both real data and synthetic data, to evaluate our itemset-DSP mining algorithm, and to demonstrate that the proposed algorithm can find interesting patterns, and it is effective and efficient.

The rest of the paper is organized as follows. We formulate the problem of itemset-DSP mining in Section 2, and review related work in Section 3. In Section 4, we discuss the critical techniques of our method (called iDSP-Miner). We report a systematic empirical study in Section 5, and conclude the paper in Section 6.

2 Problem Formulation

We start with some preliminaries. Let Σ be the *alphabet*, which is a finite set of distinct items. An *element* (of sequences, defined below) is a subset e of Σ. The *size* of e is the number of items in e, denoted by $|e|$. Given two elements e and e' such that e' is not empty, if $e' \subseteq e$, then we say e' is a *sub-element* of e.

An *itemset-based sequence* S over Σ is an ordered list of elements with the form $S = < e_1 e_2 ... e_n >$, where e_i $(1 \leq i \leq n)$ is an element. The *length* of S is the number of elements in S, denoted by $||S||$. For brevity, below we will often call itemset-based sequences as sequences when it is clear what is meant.

We use $S_{[i]}$ to denote the i-th element in S $(1 \leq i \leq ||S||)$. For two elements $S_{[i]}$ and $S_{[j]}$ in S, satisfying $1 \leq i < j \leq ||S||$, the *gap* between $S_{[i]}$ and $S_{[j]}$, denoted by $Gap(S, i, j)$, is the number of elements between $S_{[i]}$ and $S_{[j]}$ in S. Thus, $Gap(S, i, j) = j - i - 1$.

Example 1. For $S =<$\{*bread, milk*\} \{*milk, towel*\} \{*coffee, beef, cola*\} \{*lipstick*\}$>$, we have $||S|| = 4$, $S_{[1]} =$ \{*bread, milk*\}, and $|S_{[1]}| = 2$. The sub-elements of $S_{[1]}$ include \{*bread*\}, \{*milk*\} and \{*bread, milk*\}. The gap between $S_{[1]}$ and $S_{[4]}$, i.e. $Gap(S, 1, 4)$, is 2.

For two sequences S and S' satisfying $||S|| \geq ||S'||$, if there exist integers $1 \leq k_1 < k_2 < ... < k_{||S'||} \leq ||S||$, such that $S'_{[i]} \subseteq S_{[k_i]}$ for all $1 \leq i \leq ||S'||$, then we say $< k_1, k_2, ..., k_{||S'||} >$ is an *occurrence* of S' in S (and we also say S contains S'). Observe that S' may have more than one occurrences in S.

The gap constraint γ is defined as an interval which consists of two nonnegative integers $[\gamma.min, \gamma.max]$. Given two sequences S and S', let $< k_1, k_2, ..., k_{||S'||} >$ be an occurrence of S' in S. We say S' is a *subsequence* of S (and S is a *super-sequence* of S'). Furthermore, if for all $1 \leq i < ||S'||$, $\gamma.min \leq Gap(S, k_i, k_{i+1}) \leq \gamma.max$, we say that S' is a *subsequence* of S with gap constraint γ, denoted by $S' \sqsubseteq_\gamma S$.

Example 2. Let S =<{*book, milk*} {*bread*} {*book, milk, coffee*} {*coffee*}> and S' =<{*book, milk*} {*coffee*}>. Since $S'_{[1]}$ ={*book, milk*} $\subseteq S_{[1]}$ = {*book, milk*} and $S'_{[2]}$ = {*coffee*} $\subseteq S_{[3]}$ = {*book, milk, coffee*}, $< 1, 3 >$ is an occurrence of S' in S. Similarly, $S'_{[1]} \subseteq S_{[1]}$, $S'_{[2]} \subseteq S_{[4]}$, so $< 1, 4 >$ is also an occurrence of S' in S. Let gap constraint $\gamma = [2, 3]$. Because $2 \leq Gap(S, 1, 4) = 2 \leq 3$, $S' \sqsubseteq_{[2,3]} S$.

The *support* of a sequence P with gap constraint γ in sequence set D, denoted by $Sup(D, P, \gamma)$, is defined by Equation 1.

$$Sup(D, P, \gamma) = \frac{|\{S \in D \mid P \sqsubseteq_\gamma S\}|}{|D|} \tag{1}$$

Please note that given an element e (i.e. a length-1 sequence), the support of e, which is the ratio of the number of sequences containing e to $|D|$, is independent of γ. Thus, we denote $Sup(D, e)$ the support of e in D for brevity.

Definition 1 (Minimal Itemset-Based Distinguishing Sequential Patterns with Gap Constraint). *Given two sets of itemset-based sequences, D_+ and D_-, two thresholds, α and β, and gap constraint γ, a sequence $P = < e_1 e_2 ... e_{||P||} >$ is a Minimal Itemset-based Distinguishing Sequential Pattern with Gap Constraint (itemset-DSP), if the following conditions are true:*

1. *(support contrast) $Sup(D_+, P, \gamma) \geq \alpha$ and $Sup(D_-, P, \gamma) \leq \beta$;*
2. *(minimality) There does not exist a sequence $P' =< e_k e_{k+1} ... e_{k+m} >$ satisfying Condition 1 such that $k \geq 1$ and $k + m \leq ||P||$.*

Given α, β, and γ, the minimal itemset-based distinguishing sequential pattern mining problem is to find all the itemset-DSPs from D_+ and D_-.

Table 2. List of itemset-DSPs discovered in Table 1 ($\alpha = 0.6, \beta = 0.3, \gamma = [0, 2]$)

ID	itemset-DSP P	$Sup(D_+, P, \gamma)$	$Sup(D_-, P, \gamma)$
P1	<{*coffee, beef, cola*} {*milk*}>	0.75	0.25
P2	<{*coffee, beef*} {*milk*}>	0.75	0.25
P3	<{*coffee, cola*} {*milk*}>	0.75	0.25
P4	<{*beef, cola*} {*milk*}>	0.75	0.25
P5	<{*beef*} {*milk*}>	0.75	0.25
P6	<{*lipstick*}>	0.75	0.25
P7	<{*beef, perfume*}>	0.75	0.0
P8	<{*perfume*}>	0.75	0.0

Example 3. Consider the sequences in Table 1. Let support thresholds $\alpha = 0.6$ and $\beta = 0.3$, and gap constraint $\gamma = [0, 2]$. There are 8 itemset-DSPs (Table 2) discovered from the married customers (D_+) and the unmarried customers (D_-). Taking <{*beef, cola*} {*milk*}> for instance, we can see that 75% married customers (compared against 25% unmarried customers) buy beef and cola together, and will buy milk within the next three shopping purchases.

Table 3 lists the frequently used notations of this paper.

Table 3. Summary of notations

Notation	Description
Σ	alphabet (the set of items)
$\|e\|$	the size of e (the number of items in e)
$e' \subseteq e$	e' is a sub-element of e
$\|\|S\|\|$	length of S (the number of elements in S)
$S_{[i]}$	the i-th element in S ($1 \leq i \leq \|\|S\|\|$)
γ	gap constraint
$S' \sqsubseteq_\gamma S$	S' is a subsequence of S with gap constraint γ
$Sup(D, P, \gamma)$	support of sequence P with gap constraint γ in sequence set D
D_+, D_-	the positive, negative sequence sets resp.
α, β	the positive, negative support thresholds resp.

3 Related Work

Sequential pattern mining [3] is a significant task in data mining and has attracted extensive attention from both research and industry. Several types of sequential patterns, such as frequent sequential pattern [4], distinguishing sequential pattern [5], closed sequential pattern [6], (partial) periodic sequential pattern [7,8] and partial order pattern [9], have been proposed. There are quite a few successful applications of sequential pattern mining, such as protein and nucleotide sequence analysis [10,11], software bug feature discovery [12] and musical data analysis [13].

There are several studies on mining sequential patterns from sequences in which each element is an itemset. For example, Rabatel *et al.* [14] considered mining sequences with contextual information. Han *et al.* [7] considered mining sequences where each position contains an itemset. Feng *et al.* [15] proposed a language-independent keyword extraction algorithm based on mining sequential patterns without semantic dictionary. Chang *et al.* [16] used the length of time interval to measure the importance of patterns, and presented a framework to mine time-interval weighted sequential patterns. Recently, Low-Kam *et al.* [17] proposed a method to mine statistically significant, unexpected patterns, so that the number of discovered patterns is reduced. However, these studies are significantly different from our study, since we consider the support contrast measure instead of just the support measure.

There are several studies considering gap constraint in sequential pattern mining. For example, Antunes *et al.* [18] proposed an algorithm to handle the sequence pattern mining problem with gap constraint based on *PrefixSpan* [19]. Xie *et al.* [20] studied the discovery of frequent patterns satisfying one-off condition and gap constraint from a single sequence. Zhang *et al.* [21] solved the problem of mining frequent periodic patterns with gap constraint from a single sequence.

Distinguishing sequential patterns has many interesting applications, as it can describe contrast information between different classes of sequences. Ji *et al.* [5] proposed *ConsGapMiner* for mining minimal distinguishing subsequence patterns

with gap constraints. Shah *et al.* [22] mined contrast patterns with gap constraint from peptide datasets, and applied patterns to build a supervised classifier for folding prediction. Deng *et al.* [23] built a classifier for sequence data based on contrast sequences. Wang *et al.* [24] introduced the concept of density into distinguishing sequential pattern mining problem, and designed a method to mine this kind of contrast patterns.

To the best our knowledge, there are no previous existing methods tackling exactly the same problem as itemset-DSP mining. The most related previous work is that of Ji *et al.* [5], which finds the minimal distinguishing subsequences. However, it is considerably different from our work since they focused on the sequences in which each element is a single item rather than an itemset. Moreover, Ji *et al.* [5] didn't consider the concise representation of patterns.

4 Design of iDSP-Miner

In this section, we present our method, iDSP-Miner, for mining itemset-DSPs from D_+ and D_-. In general, the framework of iDSP-Miner includes: candidate element generation, candidate pattern enumeration, support contrast checking and minimality test. Technically, the key issues of iDSP-Miner are the generation and effective representation of candidate elements.

4.1 Candidate Element Generation and Representation

Recall that an itemset-DSP is an ordered list of elements. For candidate itemset-DSP generation, the first step is enumerating the elements that can be used to compose a candidate itemset-DSP.

Naturally, we want to know "how to represent patterns in a concise way and how to generate candidate elements efficiently?" To answer this question, we first make some observations and then introduce some necessary definitions.

Observation 1. *Every element of an itemset-DSP must be a subset of a sequence element in the dataset.*

By Observation 1, a brute-force way is enumerating the subsets of all sequence elements that occur in the given data as candidate elements. Clearly this method is time-consuming, and the number of candidate elements can be massive.

Observation 2. *Some itemset-DSP may be a subsequence of some other itemset-DSP. For example, in Table 2, $P2, P3, P4$ and $P5$ are subsequences of $P1$.*

Definition 2 (Element Instance). *Given a set of sequences D, if there are sequences S, $S' \in D$ and integers i and j such that $1 \leq i \leq ||S||$ and $1 \leq j \leq ||S'||$, and $e = S_{[i]} \cap S'_{[j]}$ is non-empty, we call e an element instance in D.*

Example 4. Let S =<{*bread, milk*} {*milk, towel*}> and S' =<{*coffee, cola*} {*bread, coffee, book*}>. Then, there are 7 element instances: {*bread, milk*} ($S_{[1]} \cap S_{[1]}$), {*milk, towel*} ($S_{[2]} \cap S_{[2]}$), {*coffee, cola*} ($S'_{[1]} \cap S'_{[1]}$), {*bread, coffee, book*} ($S'_{[2]} \cap S'_{[2]}$), {*milk*} ($S_{[1]} \cap S_{[2]}$), {*bread*} ($S_{[1]} \cap S'_{[2]}$), and {*coffee*} ($S'_{[1]} \cap S'_{[2]}$).

Given an element instance e, the *position* of e in sequence S, denoted by $pos(e, S)$, is $\{i \mid e \subseteq S_{[i]}\}$. For a sequence set D, the *position list* of e, denoted by $posList(e, D)$, is the set of positions of e associated with all sequences of D. That is, $posList(e, D) = \{< S.id, pos(e, S) > \mid S \in D\}$, where $S.id$ is the index of S in D. We will ignore $pos(e, S)$ in $posList(e, D)$ if $pos(e, S) = \emptyset$.

Example 5. Consider Table 1 and element $\{book\}$. Then, $posList(\{book\}, D_+) = \{< S2, \{2\} >, < S3, \{1, 3\} >\}$.

iDSP-Miner starts with computing all element instances in D_+ (denoted by E_+) and all element instances in D_- (denoted by E_-), respectively.

Theorem 1. *Given sequence set D, sequence P and gap constraint γ, we have $Sup(D, P', \gamma) \geq Sup(D, P, \gamma)$ for all $P' =< P_{[k]}P_{[k+1]}...P_{[k+m]} >$ such that $k \geq 1$ and $k + m \leq ||P||$.*

Proof (Outline). For all $S \in D$, if $P \sqsubseteq_\gamma S$, then we have $P' \sqsubseteq_\gamma S$. □

Corollary 1. *Let P be an itemset-DSP satisfying conditions in Definition 1. For each element e of P, we have $Sup(D_+, e) \geq \alpha$.*

Proof (Outline). By Theorem 1, we have $Sup(D_+, e) \geq Sup(D_+, P, \gamma) \geq \alpha$. □

It follows that every element of an itemset-DSP must be a subset of an element instance in E_+. Moreover, we have following pruning rule for E_+:

Pruning Rule 1. *Element instances $e \in E_+$ satisfying $Sup(D_+, e) < \alpha$ can be pruned from E_+.*

A practical observation is that otherwise similar sequences have sets and some of their subsets in equal positions in the result set. Thus, we need an efficient method to represent the set-subset structures and an efficient way to maintain those structures in our mining algorithm, at the same time avoiding repeated computations on the sequence elements. We will introduce the concept of equivalence element for the representation of the set-subset structures, and a split operation to maintain these structures in our algorithm.

Definition 3 (Element Closure). *Given an element e, the closure of e, denoted by $\mathcal{C}(e)$, is the set of all non-empty sub-elements of e.*

To concisely represent iDSPs and also to make iDSP-Miner efficient, we introduce a concept that is somehow similar to "borders" [25] and "equivalence class" [26] (which were used previously in data mining). Traditionally, both "border" and *equivalence class* were used to represent collections of itemsets that share some properties such as "always occur together in sequences of D". We are interested in a particular kind of borders each containing one longest itemset (similar to closed pattern) and several shorter itemsets; traditionally, such a border represents all itemsets that are subsets of the longest pattern and are supersets of some of the shorter patterns. In this paper, we use such borders in a new and different way, by having the shorter itemsets as excluders.

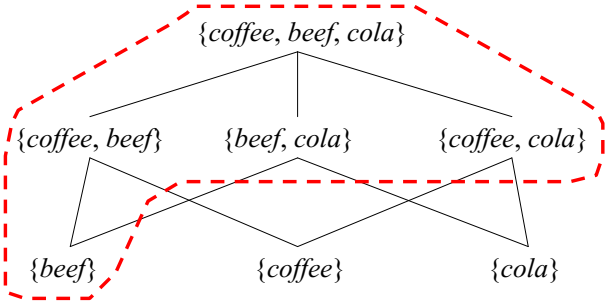


Fig. 1. Illustration of $[\{\{coffee\}, \{cola\}\}, \{coffee,\ beef,\ cola\}]$ (within the red dash line)

We define *equivalence element* to uniquely and concisely represent a set of elements, using (1) an element c (which we will call a *closed element*) and (2) a set $\mathcal{X}$ of elements (which we will call *excluders*), in the form of $[\mathcal{X}, c]$; here, $[\mathcal{X}, c]$ represents $\{e \mid e \subseteq c \wedge e \not\subseteq x \text{ for every } x \in \mathcal{X}\}$.

Observe that we have the following relationships between the closure and the equivalence element: $\mathcal{C}(c) = [\emptyset, c]$ and $[\mathcal{X}, c] = \mathcal{C}(c) \setminus \bigcup_{x \in \mathcal{X}} \mathcal{C}(x)$.

For example, as shown in Figure 1, $[\{\{coffee\}, \{cola\}\}, \{coffee,\ beef,\ cola\}] = \{\{coffee,\ beef,\ cola\}, \{coffee,\ beef\}, \{coffee,\ cola\}, \{beef,\ cola\}, \{beef\}\}$.

For element instances in E_+, we construct their closures, denoted by $EC^r_+ = \{[\emptyset, e] \mid e \in E_+\}$. Please note that there may be some redundancy in EC^r_+. For example, $[\emptyset, e'] \subset [\emptyset, e]$ if e and e' are two elements in E_+ satisfying $e' \subset e$.

To handle those subsets with different support, we define the *split operation* to divide an equivalence element into two disjoint parts. For equivalence element $\hat{e} = [\mathcal{X}, e] \in EC^r_+$, if there is an element $e' \in E_+$ such that $e' \in \hat{e}$ and $e' \neq e$, we split $\hat{e}$ by e', denoted by $\hat{e}|e'$, into two disjoint equivalence elements $[\mathcal{X} \widetilde{\cup} \{e'\}, e]$ and $[\{x \in \mathcal{X} \mid x \subset e'\}, e']$, where $\mathcal{X} \widetilde{\cup} \{e'\}$ denote $\mathcal{X} \cup \{e'\} \setminus \bigcup_{x \in \mathcal{X}} \{x \mid x \subset e'\}$.

Example 6. Let $\hat{e} = [\{\{coffee\}, \{beef\}\}, \{coffee,\ beef,\ cola\}]$, and $e' = \{coffee,\ cola\}$. The results of splitting $\hat{e}$ by e' are: $[\{\{coffee,\ cola\}, \{beef\}\}, \{coffee,\ beef,\ cola\}]$ and $[\{\{coffee\}\}, \{coffee,\ cola\}]$.

Obviously, $[\mathcal{X}, e] = [\mathcal{X} \widetilde{\cup} \{e'\}, e] \cup [\{x \in \mathcal{X} \mid x \subset e'\}, e']$. We use EC_+ to denote the set of all equivalence elements after removing the redundancy.

Corollary 2. *Given sequence set D, gap constraint γ, support threshold β, and element e in sequence P, if $Sup(D, e) \leq \beta$, then $Sup(D, P, \gamma) \leq \beta$.*

Proof (Outline). By Theorem 1, we have $Sup(D, P, \gamma) \leq Sup(D, e) \leq \beta$. □

It follows that an element e that satisfies $Sup(D_-, e) > \beta$ may occur in an itemset-DSP. Thus, for equivalence element $\hat{e} = [\mathcal{X}, e] \in EC_+$, if there is an element $e' \in E_-$ such that $Sup(D_-, e') > \beta$, $e' \in \hat{e}$ and $e' \neq e$, we split $\hat{e}$ by e' into two equivalence elements $[\mathcal{X} \widetilde{\cup} \{e'\}, e]$ and $[\{x \in \mathcal{X} \mid x \subset e'\}, e']$. We denote EC the set of equivalence elements after this splitting process.

Given equivalence element $\hat{e} = [\mathcal{X}, e] \in EC$, for any element $e' \in \hat{e}$, we have $posList(e, D_+) = posList(e', D_+)$. iDSP-Miner takes each equivalence element in EC as a set of candidate elements to generate candidate patterns. This leads generally to a significant reduction in the number of candidate elements. The details will be discussed in Section 4.2.

Example 7. Consider Table 1. Given $\alpha = 0.6, \beta = 0.3, \gamma = [0, 2]$, we see that $EC = \{[\emptyset, \{milk\}], [\emptyset, \{lipstick\}], [\emptyset, \{beef\}], [\emptyset, \{cola\}], [\emptyset, \{bread\}], [\{\{bread\}\}, \{bread, perfume\}], [\{\{cola\}, \{beef\}, \{coffee\}\}, \{coffee, beef, cola\}], [\emptyset, \{coffee\}]\}$.

4.2 Pattern Mining

To ensure the completeness of candidate pattern enumeration, iDSP-Miner traverses the set enumeration tree [27] of equivalence elements in EC in a depth-first manner.

Given a node $\mathbb{N}$ in the set enumeration tree, let $\hat{P}$ be the list of equivalence elements that occur on the path from the root node to $\mathbb{N}$. Thus, $\hat{P}$ is a concise representation of $\{P \mid P_{[i]} \in \hat{P}_{[i]} \text{ for } 1 \leq i \leq ||P||, ||P|| = ||\hat{P}||\}$. As the elements represented by an equivalence element occur together, given sequence set D and gap constraint γ, for $P, P' \in \hat{P}$ $(P \neq P')$, we have $Sup(D, P, \gamma) = Sup(D, P', \gamma)$.

We define $Sup(D, \hat{P}, \gamma) = Sup(D, P, \gamma)$, where $P \in \hat{P}$ and $P_{[i]}$ is the closed element of $\hat{P}_{[i]}$ for $1 \leq i \leq ||P||$.

Pruning Rule 2. *If $Sup(D_+, \hat{P}, \gamma) \geq \alpha$ and $Sup(D_-, \hat{P}, \gamma) \leq \beta$, according to the minimality condition in the problem definition (Definition 1), all descendants of $\mathbb{N}$ can be pruned.*

Pruning Rule 3. *If $Sup(D_+, \hat{P}, \gamma) < \alpha$, according to Theorem 1, all descendants of $\mathbb{N}$ can be pruned.*

If $Sup(D_+, \hat{P}, \gamma) \geq \alpha$ and $Sup(D_-, \hat{P}, \gamma) > \beta$, then, to search for super-patterns with lower support in D_-, we extend the set enumeration tree by appending another equivalence element (from EC) as a child of $\mathbb{N}$, to $\hat{P}$ to generate a new candidate.

For any pattern satisfying the support contrast condition, iDSP-Miner performs the minimality test. That is, it compares the pattern with the other discovered patterns to remove the non-minimal ones.

To further remove the redundant representation of itemset-DSPs, for the patterns satisfying the support contrast and minimality conditions, iDSP-Miner simplifies the representation of patterns as follows. Given two patterns $\hat{P}$ and $\hat{P}'$ with the same length, let $\hat{P}_{[k]} = [\mathcal{X}, e]$ and $\hat{P}'_{[k]} = [\mathcal{X}', e']$ $(k \in [1, ||\hat{P}||])$. If $e' \in \mathcal{X}$ and $\hat{P}_{[i]} = \hat{P}'_{[i]}$ for $1 \leq i \leq ||\hat{P}||$, $i \neq k$, then we say $\hat{P}$ and $\hat{P}'$ are *mergeable*. iDSP-Miner merges $\hat{P}$ with $\hat{P}'$ into a new pattern $\hat{P}''$, such that $\hat{P}''_{[i]} = \hat{P}_{[i]}$ and $\hat{P}''_{[k]} = [\mathcal{X} \cup \mathcal{X}' \setminus \{e'\}, e]$ $(1 \leq k, i \leq ||\hat{P}||, i \neq k)$.

Algorithm 1. iDSP-Miner($D_+, D_-, \alpha, \beta, \gamma$)

Input: D_+: a class of sequences, D_-: another class of sequences, α: minimal support for D_+, β: maximal support for D_-, γ: gap constraint
Output: Ans: the set of itemset-DSPs of D_+ against D_- with concise representation

initialize $Ans \leftarrow \emptyset$;
$E_+ \leftarrow$ element instances in D_+; $E_- \leftarrow$ element instances in D_-;
$EC \leftarrow \{[\emptyset, e] \mid e \in E_+\}$;
while $\exists \hat{e} = [\mathcal{X}, e] \in EC, e' \in E_+$ such that $e' \in \hat{e}$ and $e' \neq e$ **do**
  $EC \leftarrow EC \setminus \hat{e} \cup (\hat{e}|e')$;
end while
while $\exists \hat{e} = [\mathcal{X}, e] \in EC, e' \in E_-$ satisfying $Sup(e', D_-) > \beta$ such that $e' \in \hat{e}$ and $e' \neq e$ **do**
  $EC \leftarrow EC \setminus \hat{e} \cup (\hat{e}|e')$;
end while
for each candidate pattern $\hat{P}$ searched by traversing the set enumeration tree of EC in a depth-first manner **do**
  if $Sup(D_+, \hat{P}, \gamma) < \alpha$ **then**
    prune all super-sequences of $\hat{P}$;
  end if
  if $Sup(D_-, \hat{P}, \gamma) \leq \beta$ **then**
    prune all super-sequences of $\hat{P}$;
    if $\hat{P}$ is minimal **then**
      $Ans \leftarrow Ans \cup \{\hat{P}\}$;
    end if;
  end if
end for
for every mergeable pair of $\hat{P}, \hat{P}' \in Ans$ **do**
  merge $\hat{P}$ with $\hat{P}'$;
end for
return Ans;

Example 8. Let $\hat{P}$ = <[{{*coffee, cola*}, {*beef*}}, {*coffee, beef, cola*}] [∅, {*milk*}]>, $\hat{P}'$ = <[{{*cola*}}, {*coffee, cola*}] [∅, {*milk*}]>. Then, we can get <[{{*cola*}, {*beef*}}, {*coffee, beef, cola*}] [∅, {*milk*}]> by merging $\hat{P}$ with $\hat{P}'$.

Algorithm 1 gives the pseudo-code of iDSP-Miner. Again, taking Table 1 as an example, the results of iDSP-Miner include: <[{{*bread*}},{*bread, perfume*}]>, <[∅,{*lipstick*}]>, and <[{{*cola*},{*coffee*}},{*coffee, beef, cola*}] [∅,{*milk*}]>. Compared with patterns listed in Table 2, we can see that our method can represent patterns more concisely.

5 Empirical Evaluation

In this section, we report a systematic empirical study using both real and synthetic sequence sets to test the effectiveness and efficiency of iDSP-Miner. All experiments were conducted on a PC computer with an Intel Core i7-3770

Table 4. Sequence set characteristics

Sequence set	DB	DM	IR
Num. of sequences	100	100	100
Num. of items	1921	1966	1477
Avg. element size	4.30	6.67	4.77
Min. element size	1	1	1
Max. element size	49	60	54
Avg. sequence length	30.54	20.12	20.91
Min. sequence length	5	10	7
Max. sequence length	44	37	37

3.40 GHz CPU, and 8 GB main memory, running Windows 7 operating system. All algorithms were implemented in Java and compiled using JDK 8.

5.1 Effectiveness

Arnetminer[1] groups computer science researchers by different research topics and computes the H-index score for each researcher. We apply iDSP-Miner to analyzing the differences of publication preferences among researchers in database (DB), data mining (DM) and information retrieval (IR). We fetch top 100 scholars in each topic sorted by the H-index score. For each researcher, we construct a sequence, in which an item is the title of a conference or a journal where the researcher published a paper, and an element is the set of items that are in the same year. We collect the publication information of each researcher until 2013 from DBLP[2]. Table 4 shows the characteristics of the sets of sequences. We use "$D_+ vs D_-$" to denote the two sequence sets that we selected to analyze. For example, "DB*vs*IR" implies that we find itemset-DSPs from DB against IR.

Table 5 summarizes the characteristics of discovered patterns. We can see that with the increase of α, the number of itemset-DSPs, the average/maximum pattern length, and the average/maximum element size are typically decreased.

Table 6 lists the discovered itemset-DSPs with concise representation in IR*vs*DB when α (min support for IR) $= 0.4$, β (max support for DB) $= 0.2$, and γ (gap constraint) $= [0, 5]$. We can observe that, as shown by patterns <[{{CIKM}}, {SIGIR, CIKM}]> and <[∅, CIKM] [∅, {CIKM}] [∅, {CIKM}]>, researchers in IR prefer publishing in SIGIR and CIKM conferences.

Figure 2 shows the number of itemset-DSPs and the number of itemset-DSPs with concise representation. We can see that the number of discovered patterns is reduced by our equivalence element based representation. Especially, when the number of itemset-DSPs is large, using a small number of itemset-DSPs where equivalence items are used, one can represent many more detailed itemset-DSPs in a highly structured manner. Thus, the results of iDSP-Miner are easier to manage and easier to digest for user. Notably, if the average element size is close

[1] http://arnetminer.org/
[2] http://dblp.uni-trier.de/

Table 5. Characteristics of discovered patterns by iDSP-Miner ($\beta = 0.2$, $\gamma = [0, 5]$) (# iDSP(CR): the number of itemset-DSPs with concise representation, $||P||$: pattern length, $|e|$: element size)

Sequence sets	α	# iDSP(CR)	Avg. $\|\|P\|\|$	Max. $\|\|P\|\|$	Avg. $\|e\|$	Max. $\|e\|$
DB*vs*DM	0.25	208	2.31	5	1.05	3
	0.3	153	2.29	5	1.05	2
	0.35	109	2.26	5	1.05	2
	0.4	82	2.15	4	1.07	2
	0.45	52	2.31	4	1.03	2
DB*vs*IR	0.25	47	1.04	2	1.30	3
	0.3	30	1.03	2	1.25	2
	0.35	21	1	1	1.26	2
	0.4	17	1	1	1.26	2
	0.45	12	1	1	1.2	2
DM*vs*IR	0.25	96	1.16	3	1.40	3
	0.3	63	1.16	3	1.38	3
	0.35	46	1.20	3	1.33	3
	0.4	35	1.27	2	1.24	2
	0.45	27	1.28	2	1.17	2
DM*vs*DB	0.25	85	1.17	4	1.39	3
	0.3	54	1.17	4	1.37	3
	0.35	34	1.08	3	1.38	3
	0.4	25	1.09	3	1.29	2
	0.45	18	1.06	2	1.24	2
IR*vs*DB	0.25	38	1.06	3	1.56	3
	0.3	27	1.07	3	1.51	3
	0.35	22	1.10	3	1.32	3
	0.4	15	1.10	3	1.27	2
	0.45	8	1.11	3	1.29	2
IR*vs*DM	0.25	54	1.42	6	1.37	3
	0.3	37	1.41	5	1.35	3
	0.35	31	1.57	5	1.18	3
	0.4	23	1.66	4	1.13	2
	0.45	18	1.86	4	1.09	2

to 1.0, then the difference in the output size between the concise representation and listing all patterns explicitly is likely to be small, like in Figure 2 (a).

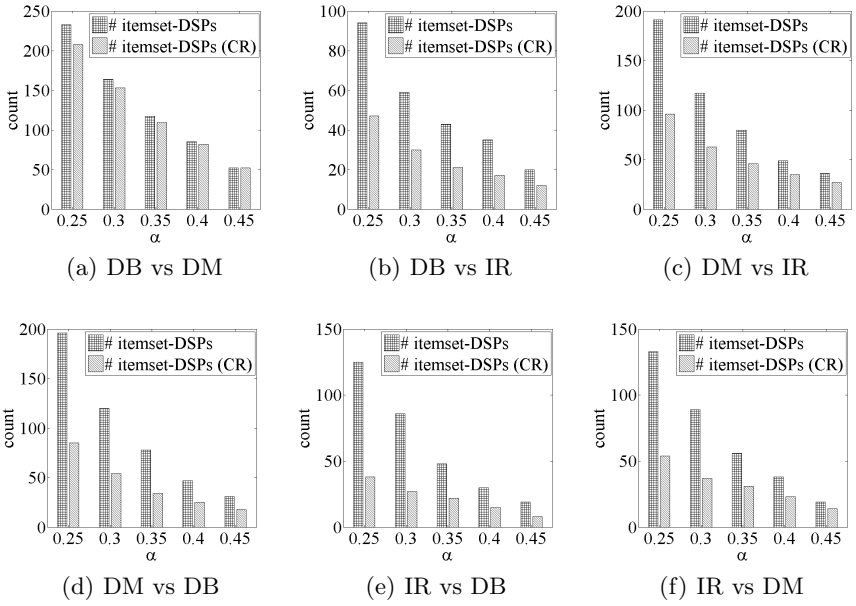
We note that iDSP-Miner is efficient. For example, the average runtime is 2.51 seconds when $\alpha = 0.4$, $\beta = 0.2$ and $\gamma = [0, 5]$. We will present more analysis on the efficiency of iDSP-Miner in the next section.

5.2 Efficiency

To the best of our knowledge, there were no previous methods tackling exactly the same mining problem as the one studied in this paper. Therefore, we evaluate the efficiency of only iDSP-Miner and the baseline method, which takes each subset of an element instance as a candidate element. Please note that the mining

Table 6. item-DSPs with concise representation for IR*vs*DB ($\alpha = 0.4$, $\beta = 0.2$, $\gamma = [0, 5]$)

<[∅, {TREC, SIGIR}]>	<[{{CIKM}}, {SIGIR, CIKM}]>
<[∅, {JASIST}]>	<[∅, {SIGIR, SIGIR Forum}]>
<[∅, {SIGIR, Inf. Retr.}]>	<[∅, {SIGIR, Inf. Process. Manage.}]>
<[∅, WWW]]>	<[∅, {ACM Trans. Inf. Syst.}] [∅, {CIKM}]>
<[{{CIKM}}, {TREC, CIKM}]>	<[∅, {TREC, Inf. Process. Manage.}]>
<[∅, {RIAO}]>	<[{{CIKM}}, {CIKM, SIGIR Forum}]>
<[∅, {JASIS}]>	<[∅, CIKM] [∅, {CIKM}] [∅, {CIKM}]>
<[∅, {SIGIR, ECIR}]>	



(a) DB vs DM (b) DB vs IR (c) DM vs IR

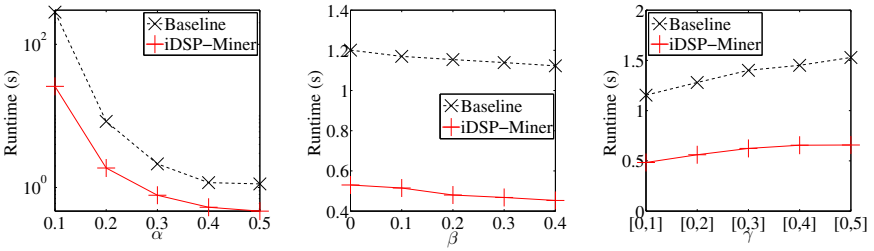
(d) DM vs DB (e) IR vs DB (f) IR vs DM

Fig. 2. Comparison of the number of itemset-DSPs and the number of itemset-DSPs with concise representation

result of iDSP-Miner is the set of itemset-DSPs with concise representation, while the mining result of the baseline method is the set of itemset-DSPs.

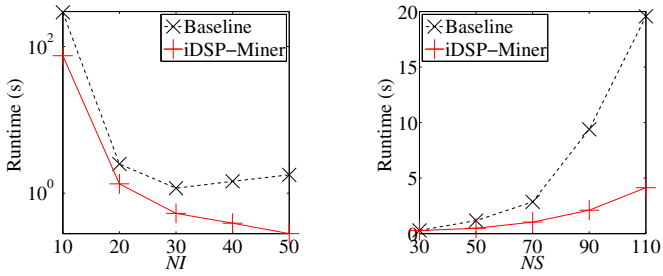
We generate synthetic sequence sets for efficiency test. There are four parameters for synthetic sequence set generation: the number of items (denoted by NI), the number of sequences (denoted by NS), the average sequence length (denoted by SL), and the average element size (denoted by ES).

Figure 3 shows the efficiency test of iDSP-Miner with respect to α, β and γ when $NI = 30$, $NS = 50$, $SL = 10$ and $ES = 4$. We can see that the runtime of both iDSP-Miner and the baseline method decrease with the increase of α and β, while the runtime of both iDSP-Miner and the baseline method increase with larger gap constraint. iDSP-Miner runs faster than the baseline method, since

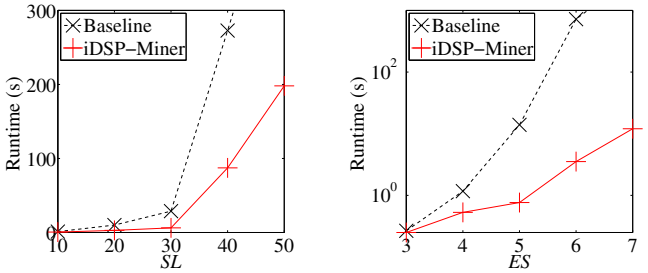


(a) β=0.0, γ=[1,3] (b) α=0.4, γ=[1,3] (c) α=0.4, β=0.0

Fig. 3. Efficiency evaluation



(a) NS=50, SL=10, ES=4 (b) NI=30, SL=10, ES=4



(c) NI=30, NS=50, ES=4 (d) NI=30, NS=50, SL=10

Fig. 4. Scalability evaluation

iDSP-Miner employs a concise representation for candidate patterns to avoid repeated computation.

Figure 4 illustrates the scalability of iDSP-Miner with respect to NI, NS, SL and ES when $\alpha = 0.2$, $\beta = 0.0$ and $\gamma = [1,3]$. When NI becomes larger, more elements can be generated. However, the number of elements/patterns satisfying the positive support threshold condition decreases. Thus, the runtime is reduced by Pruning Rules 1 and 3. On the other hand, more candidate patterns will be generated by increasing NS, SL and ES. Correspondingly, the runtime of iDSP-Miner will increase. Again, we can see that iDSP-Miner runs faster than the baseline method for all parameter settings.

Please note that in some cases in Figure 3 and Figure 4, logarithmic scale has been used for the runtime to better demonstrate the difference in the behavior between iDSP-Miner and the baseline. This should be clear from the figures.

6 Conclusions

In this paper, we propose and study a new problem of mining itemset-based distinguishing sequential patterns with a gap constraint. To mine these patterns and to present the result sets concisely, we propose an algorithm called iDSP-Miner. Our experiments verify the effectiveness and efficiency of iDSP-Miner.

In our work we apply the straightforward assumption that the user preference for output is a minimum distinguishing sequential pattern. It is interesting to explore strategies for incorporating domain constraints into itemset-DSP mining, and design a tuning mechanism for positive and negative support thresholds. Instead of the traditional gap constraint used in this paper, we are also considering the use of temporal gap constraints for sequences of timestamped events.

References

1. Dong, G., Pei, J.: Sequence Data Mining. Springer-Verlag, Berlin, Heidelberg (2007)
2. Dong, G., Bailey, J., eds.: Contrast Data Mining: Concepts, Algorithms, and Applications. CRC Press (2012)
3. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering, pp. 3–14. IEEE Computer Society, Washington, DC (1995)
4. Zaki, M.J.: Spade: an efficient algorithm for mining frequent sequences. Mach. Learn. **42**(1–2), 31–60 (2001)
5. Ji, X., Bailey, J., Dong, G.: Mining minimal distinguishing subsequence patterns with gap constraints. Knowl. Inf. Syst. **11**(3), 259–286 (2007)
6. Yan, X., Han, J., Afshar, R.: Clospan: mining closed sequential patterns in large databases. In: SDM (2003)
7. Han, J., Dong, G., Yin, Y.: Efficient mining of partial periodic patterns in time series database. In: Proceedings of the 15th International Conference on Data Engineering, pp. 106–115. IEEE Computer Society, Washington, DC (1999)
8. Zhang, M., Kao, B., Cheung, D.W., Yip, K.Y.: Mining periodic patterns with gap requirement from sequences. ACM Trans. Knowl. Discov. Data **1**(2), August 2007
9. Pei, J., Wang, H., Liu, J., Wang, K., Wang, J., Yu, P.S.: Discovering frequent closed partial orders from strings. IEEE Trans. on Knowl. and Data Eng. **18**(11), 1467–1481 (2006)
10. Ferreira, P.G., Azevedo, P.J.: Protein sequence pattern mining with constraints. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 96–107. Springer, Heidelberg (2005)
11. She, R., Chen, F., Wang, K., Ester, M., Gardy, J.L., Brinkman, F.S.L.: Frequent-subsequence-based prediction of outer membrane proteins. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 436–445. ACM, New York, NY (2003)

12. Zeng, Q., Chen, Y., Han, G., Ren, J.: Sequential pattern mining with gap constraints for discovery of the software bug features. Journal of Computational Information Systems **10**(2), 673–680 (2014)
13. Conklin, D., Anagnostopoulou, C.: Comparative pattern analysis of cretan folk songs. Journal of New Music Research **40**(2), 119–125 (2011)
14. Rabatel, J., Bringay, S., Poncelet, P.: Contextual sequential pattern mining. In: Proceedings of the 2010 IEEE International Conference on Data Mining Workshops. ICDMW 2010, pp. 981–988. IEEE Computer Society, Washington, DC (2010)
15. Feng, J., Xie, F., Hu, X., Li, P., Cao, J., Wu, X.: Keyword extraction based on sequential pattern mining. In: Proceedings of the Third International Conference on Internet Multimedia Computing and Service. ICIMCS 2011, pp. 34–38. ACM, New York, NY (2011)
16. Chang, J.H.: Mining weighted sequential patterns in a sequence database with a time-interval weight. Know.-Based Syst. **24**(1), 1–9 (2011)
17. Cécile, L.K., Chedy, R., Mehdi, K., Jian, P.: Mining statistically significant sequential patterns. In: Proceedings of the 13th IEEE International Conference on Data Mining (ICDM2013). ICDM2013, pp. 488–497. IEEE Computer Society, Dallas, TX (2013)
18. Antunes, C., Oliveira, A.L.: Generalization of pattern-growth methods for sequential pattern mining with gap constraints. In: Perner, P., Rosenfeld, A. (eds.) MLDM 2003. LNAI 2734, vol. 2734, pp. 239–251. Springer, Heidelberg (2003)
19. Pei, J., Han, J., Mortazavi-asl, B., Pinto, H., Chen, Q., Dayal, U., Chun Hsu, M.: Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth. In: Proceedings of the 17th International Conference on Data Engineering, pp. 215–224. IEEE Computer Society, Washington, DC (2001)
20. Xie, F., Wu, X., Hu, X., Gao, J., Guo, D., Fei, Y., Hua, E.: MAIL: mining sequential patterns with wildcards. Int. J. Data Min. Bioinformatics **8**(1), 1–23 (2013)
21. Zhang, M., Kao, B., Cheung, D.W., Yip, K.Y.: Mining periodic patterns with gap requirement from sequences. ACM Transactions on Knowledge Discovery from Data (TKDD) **1**(2), 7 (2007)
22. Shah, C.C., Zhu, X., Khoshgoftaar, T.M., Beyer, J.: Contrast pattern mining with gap constraints for peptide folding prediction. In: FLAIRS Conference, pp. 95–100 (2008)
23. Deng, K., Zaïane, O.R.: Contrasting sequence groups by emerging sequences. In: Gama, J., Costa, V.S., Jorge, A.M., Brazdil, P.B. (eds.) DS 2009. LNCS, vol. 5808, pp. 377–384. Springer, Heidelberg (2009)
24. Wang, X., Duan, L., Dong, G., Yu, Z., Tang, C.: Efficient mining of density-aware distinguishing sequential patterns with gap constraints. In: Bhowmick, S.S., Dyreson, C.E., Jensen, C.S., Lee, M.L., Muliantara, A., Thalheim, B. (eds.) DASFAA 2014, Part I. LNCS 8421, vol. 8421, pp. 372–387. Springer, Switzerland (2014)
25. Dong, G., Li, J.: Efficient mining of emerging patterns: discovering trends and differences. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 43–52 (1999)
26. Li, J., Liu, G., Wong, L.: Mining statistically important equivalence classes and delta-discriminative emerging patterns. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD 2007, pp. 430–439 (2007)
27. Rymon, R.: Search through systematic set enumeration. In: Proc. of the 3rd Int'l Conf. on Principle of Knowledge Representation and Reasoning. KR 1992, pp. 539–550 (1992)