Conquering the Search Space for the Calculation of the Maximal Frequent Set

Clara Nippl, Angelika Reiser, Bernhard Mitschang
Alle Rechte vorbehalten
Nachdruck auch auszugsweise verboten

©2000 SFB 342 Methoden und Werkzeuge für
die Nutzung paralleler Architekturen

Anforderungen an: Prof. Dr. A. Bode
Sprecher SFB 342
Institut für Informatik
Technische Universität München
D-80290 München, Germany

Druck: Fakultät für Informatik der
Technischen Universität München
Conquering the Search Space for the Calculation of the Maximal Frequent Set

Clara Nippl¹, Angelika Reiser¹, Bernhard Mitschang²

¹Computer Science Department, Technische Universität München
D - 80290 Munich, Germany
   e-mail: nippl@in.tum.de, Tel: +49 (89) 48095-172, Fax: +49 (89) 48095-198
²Institut für Parallele und Verteilte Höchstleistungsrechner, Universität Stuttgart
   D - 70565 Stuttgart, Germany

Abstract The most time consuming operation in data mining applications is the computation of frequent itemsets. Since the search space is exponential, efficient pruning is necessary. On the other hand, data mining on large data volumes makes the coupling of mining with database systems increasingly important. In this paper, we propose a new operator to efficiently calculate the support of candidate itemsets within the database engine. Based on this operator, we propose a novel approach to reduce search complexity by combining top-down with bottom up pruning in order to obtain an algorithmic complexity that is only proportional to the volume of the maximal frequent itemsets. In contrast to other approaches, this strategy avoids expensive database scans as well as intermediate result materialization.

1 Introduction

One of the central tasks in data mining is the discovery of frequent itemsets. The most popular example in this field is association rule mining. Given a set of transactions, where each transaction refers to a set of items, an association rule is an expression of the form $X \Rightarrow Y$ ($X$ determines $Y$), where $X$ and $Y$ are sets of items or itemsets. Each itemset is said to have a support $s$ if $s\%$ of the transactions in the database contain the itemset. The association rule is said to have confidence $c$ if $c\%$ of the transactions that contain $X$ also contain $Y$. Data mining of association rules from databases consists of finding the set of all such rules which meet the user-specified minimum confidence and support values. This task can be done in two steps. The first step consists of finding frequent itemsets, i.e. itemsets that occur in the database with certain user-specified frequency, called minimum support. Once the frequent itemsets and the corresponding supports are known, association rules can easily be generated [AY97]. Apart from association rule mining, frequent itemsets are also basic constituents of various other data mining fields as well, such as e.g. sequential pattern mining, classification problems or discovery of correlations [HGY98].

Several solutions have been devised to the problem of computing frequent itemsets. Many alternatives are variations of the Apriori algorithm [AM+95]. This method requires multiple passes over a database, which e.g. in the case of voluminous data warehouses cause prohibitive overhead in terms of I/O costs [AY97]. Furthermore, the algorithm employs a bottom-up search space exploration resulting in an explicit examination of all frequent itemsets. Since this results in exponential complexity w.r.t. the length of the longest itemset, the performance and applicability of the algorithm decreases for scenarios with long patterns and high data volumes. Another important aspect is that especially for complex applications the need for integrating data mining with the capabilities of database systems becomes imperative.
Frequent itemsets that have no superset that is frequent are called maximal frequent itemsets (MFI). The set of all maximal frequent itemsets, also called the maximal frequent set (MFS), implicitly defines the set of all frequent itemsets as well. Based on this observation, we propose a novel methodology to efficiently evaluate the maximal frequent set only. This strategy is based on a new database operator that efficiently calculates the support of candidate itemsets, as well as of all prefix subsets. Dynamic pruning combining both top-down as well as bottom-up techniques is used throughout search space exploration. As a result, the algorithmic complexity does not depend on the length of the longest MFI and scales roughly linearly with the MFS volume, being defined as the sum of the lengths of all MFIs. The most striking difference to other approaches lies in the drastically reduced I/O overhead. Thus, instead of performing multiple scans of the whole database, only selective disk accesses are necessary, depending on the current search space status. As a consequence, first experimental results [NRM99] show considerable performance improvements for the calculation of the MFS as compared to related work.

The strategy can efficiently be embedded into the database engine, resulting in a uniform processing scheme, without any additional intermediate result materializations or preparatory phases. In contrast to related work, we propose a non-blocking evaluation. Thus, first results (MFIs) can be delivered fast before the whole MFS is derived. Due to space restrictions, the complete description of the database integration as well as detailed performance evaluation can be found in [NRM99]. In this paper, we only focus on the search algorithm and its effective pruning techniques that guide the MFS processing. Please note that theorem proofs as well as detailed algorithms can be found in the appendix.

2 Related work

In [PCY97] an improvement of the Apriori algorithm was proposed, that reduces the database size (and implicitly complexity) in each pass. However, transaction trimming is an impracticable method for integrating this algorithm into an operating database engine. Other approaches [BM+97, SON95] propose methodologies to reduce the number of database scans. However, they still apply the same bottom-up methodology as the Apriori algorithm, thus also explicitly examining all frequent itemsets. As mentioned above, this yields to exponential complexity. In addition, experimental results also show that the proposed techniques cannot lower the number of database scans below 2, either. Other solutions are based on additional preprocessing of data [AY98].

In order to tackle complexity, some strategies employ randomized algorithms [GMS97]. However, these approaches are not complete, i.e. they do not guarantee that every frequent itemset will be returned.

Recently, solutions using also top-down search strategies were proposed, such as the MaxMiner algorithm [Ba98] or PincerSearch [LK98] or MaxClique and MaxEclat [ZP+97]. However, although the pruning strategies of these algorithms reduce the search complexity considerably as compared to Apriori, they still involve multiple database passes or even preparatory phases. Our measurements [NRM99] have shown that for these approaches the I/O costs alone exceed the overall costs of our algorithm.

3 Search Strategy and Support Evaluation

In our approach both generation of candidate itemsets as well as evaluation of corresponding supports
are mutually entwined algorithms. In order to understand the search space optimizations, in this paper it is sufficient to view the evaluation of given candidate itemsets as a primitive (database) operation, here called the ECS (Evaluation of Candidate Supports) operator, with the following functionality:

The input of the operator is a set of tuples of the form (itemset, item), defining a candidate itemset. The output is constituted of the same set of tuples, supplemented with an additional attribute. This attribute represents for each item of the itemset the support of the prefix that ends with that item. This strategy is exemplified in Fig. 1 for the sample itemset 100, constituted of the items \{10, 34, 55\}. An efficient and resource-effective implementation of this basic functionality is proposed and evaluated in [NRM99]. This basic operation is completed by a search space strategy, called MFSSearch, that decides on the set, sequence, and number of candidate itemsets that have to be processed in order to determine the MFS. Obviously this strategy directly influences the overall computation and space complexity. In the next section we introduce in a step-by-step manner our search strategy for itemset generation.

### 4 Generation of the Candidate Itemsets

In the following we describe the MFSSearch algorithm in detail. MFSSearch employs the functionality of the ECS operator for candidate and prefix itemset support calculation.

Given an infrequent itemset \(X = \{1, 2, \ldots, N-1, N\}\), in a top-down search it is necessary to test all of its subsets of level \(N-1\). This can be done by successively eliminating the items \(N-1, N-2, \ldots, 1\) from \(X\). It is not necessary to do this with item \(N\), since \(X - \{N\}\) is a prefix whose support is implicitly evaluated together with the support of \(X\) by the ECS operator. In the following, we will call the set of items needed to generate all unexplored subsets of level \(N-1\) for a given itemset \(X\) the ElimList of \(X\) or \(E_X\).

In this case \(E_X = \{1, 2, \ldots, N-1\}\). The subsets situated on the same level will be called siblings. However, if this procedure of generating subsets by eliminating the first \(N-1\) elements is applied recursively, duplicates are generated. This follows from the following observation:

**Observation:** If \(X_1\) and \(X_2\) are two subsets of itemset \(X\), obtained by eliminating two different items from \(X\), then \(X_1\) and \(X_2\) differ by only one item position.

Suppose \(X_1 = \{1, 2, \ldots, A, Z, B, \ldots, N-1\}\), \(X_2 = \{1, 2, \ldots, A, Y, B, \ldots, N-1\}\).

If this process is done recursively, one subset of \(X_1\) will be obtained by eliminating item \(Z\):
\(X_{1Z} = \{1, 2, \ldots, A, B, \ldots, N-1\}\). Similarly \(X_{2Y} = \{1, 2, \ldots, A, B, \ldots, N-1\}\) and \(X_{1Z} = X_{2Y}\).

In order to avoid duplicate generation, if \(X_2\) is expanded after \(X_1\), the ElimList of \(X_2\) must not contain \(Y\). More generally, if the siblings \(X_1, X_2, \ldots, X_{N-1}\) are expanded successively, the ElimList of a given itemset \(X_1\) must not contain any positions which differentiate this itemset from any of its siblings \(X_1, \ldots, X_{i-1}\). This can be done easily, if we decrease the original ElimList of \(X\) by one element in order to get the appropriate ElimList for each subset generated.
Thus for \( X = \{1, 2, \ldots, N-1, N\} \) and \( E_X = \{1, 2, \ldots, N-1\} \) we have the following sibling subsets and ElimLists:

\[
X_1 = \{1, 2, \ldots, N-2, N\}, \ E_1 = \{1, 2, \ldots, N-2\},
\]
\[
X_2 = \{1, 2, \ldots, N-3, N-1, N\}, \ E_2 = \{1, 2, \ldots, N-3\},
\]
\[
\vdots
\]
\[
X_{N-1} = \{2, \ldots, N-1, N\}, \ E_{N-1} = \emptyset.
\]

Fig. 2 shows a search space generated by the MFSSearch algorithm after employing only the ElimList technique described above, i.e. without any additional pruning as described later. We will call the subsets expanded by an itemset itself direct subsets, while subsets expanded by a sibling are called cross subsets. For instance \( \{36\} \) is a direct subset of \( \{136\} \), while \( \{16\} \) is a cross subset of \( \{136\} \), expanded by \( \{126\} \).

Based on the information given so far, we can now derive the following two basic properties.

**Theorem 1**: The ElimList method guarantees a full expansion of the search space without duplicate generation.

As a conclusion, given a superset \( X \) on level \( N \) with the ElimList \( E_x \) and a subset \( X_i \) generated by eliminating item \( N-i \) from \( X \), the ElimList of \( X_i \) is \( \{1, 2, \ldots, N-i-1\} \). This method of successively decreasing the ElimList for sibling subsets guarantees a complete and duplicate-free search space expansion.

Obviously, any subset of a frequent itemset is also frequent. Hence it is only necessary to expand the direct subsets of a given itemset if this itemset is infrequent. We call this form of pruning Direct Top-Down Pruning (DTDP).

**Theorem 2**: The upper bound for the number of itemsets considered by the MFSSearch algorithm for a finite item domain \( 1, 2, \ldots, N \) is \( 2^{N-1} \).

Thus, by employing this basic version of MFSSearch (ElimList technique and DTDP) the search complexity has already been reduced by a magnitude of 2.
Sibling subsets can be explored either backward or forward, as marked in Fig. 2 by arrows. The following section will detail on that and a summary of all possible and meaningful combinations of the various pruning and search strategies is given in Section 4.3 at the end of this chapter.

4.1 Backward Exploration (BE) of Itemsets

An important property for all backward-oriented strategies to be discussed below is given by the following theorem.

**Theorem 3:** Given two itemsets \( X_I \) and \( X_2 \), s.t. \( X_I \subseteq X_2 \). In the BE scenario \( X_I \) will be processed after \( X_2 \).

4.1.1 Cross Top-Down Pruning

In this section, we are interested in finding out how a given frequent itemset can prune cross subsets as well. In Fig. 2, if e.g. \( \{456\} \) is frequent, direct top-down pruning eliminates \( \{456\} \), but it is desirable to prune the cross subsets \( \{156\} \) and \( \{146\} \) as well. We call this Cross Top-Down Pruning (CTDP).

**Theorem 4:** Given a finite item domain \( 1, 2, \ldots, N \), In the backward exploration any itemset except itemset \( Z = \{N\} \) is a superset of at least one itemset that is not expanded yet.

Hence, once an itemset is found frequent, it can prune its (direct and cross) subsets from exploration. However, at a given moment, the search space consists of itemsets that are either a) expanded and explored, b) expanded or c) unexpanded. The fully explored search space is given in Fig. 2. Evidently, pruning can only affect category b) and c), i.e. not yet explored itemsets. Itemsets of category b) can be pruned together with their direct subsets as soon as a frequent superset is found. However, it is not clear how to prune itemsets of category c). In Fig. 2, if e.g. itemset \( X = \{456\} \) is found to be frequent, it can prune itemsets \( \{56\} \), \( \{46\} \) and \( \{6\} \). However, these are itemsets on Level 2 and 1, none of which have been expanded yet at the moment when \( X \) is explored. Thus, it is necessary to memorize \( X \) for itemsets that have not yet been explored, if these itemsets can produce subsets of \( X \). In this case, these are the itemsets \( \{12356\} \) and \( \{12346\} \). Such a set of relevant frequent itemsets, called FrequentSet, is logically assigned to each itemset element, similarly to its ElimList.

**Theorem 5:** The set of frequent itemsets \( F \) assigned to any candidate itemset contains only maximal frequent itemsets.

Given a frequent itemset \( X \) and an expanded but unexplored itemset \( Y \) (category b), with \( Y \) having a direct subset \( Z \) that is not expanded yet (i.e. of category c), s.t. \( Z \) is also a cross subset of \( X \). Now the question is how to prune the search space in order to avoid the evaluation of \( Z \).

**Theorem 6:** Given a frequent itemset \( X \) and an expanded but unexplored itemset \( Y \), \( X \not\subset Y \) and the ElimList of \( Y \) being \( E \). \( Y \) will expand a subset of \( X \) if

\[
\{y | y \in Y, y \notin X\} \subset E.
\]

This will be further on referred to as **Condition (1)**.

A relevant frequent itemset will be propagated to lower levels only if this condition is fulfilled.

**Example 1:** If the itemset \( X = \{456\} \) is frequent, it will be included in the FrequentSet of the expanded but unexplored itemsets \( \{12356\} \) and \( \{12346\} \) that also satisfy **Condition (1)**. When these itemsets are explored, they will further propagate \( X \) only to the subsets \( \{1256\} \), \( \{1246\} \) and \( \{1236\} \) on Level 4. This process continues and leads finally to the pruning of the itemsets \( \{56\} \), \( \{46\} \) and \( \{6\} \).

4.1.2 Bottom-Up Pruning

Bottom-up pruning (BUP) uses the property that if a subset of an itemset in the search space is found infrequent, it is no longer necessary to explore that itemset as it is infrequent anyway. According to The-
orem 3, in the BE scenario an itemset can never be a subset of an itemset that is expanded later. Thus, an entire itemset can never be used for BUP. However, by processing an itemset via the ECS operator, the supports of all prefixes are implicitly calculated as well. Thus, we can use infrequent prefixes for BUP.

**Definition 1:** Given an itemset \( X = \{1,2,\ldots,N-1,N\} \) with prefixes \( X_i = \{1,\ldots,i\}, X_{N-1} = \{1,2,\ldots,N-1,N\} \)

The maximal infrequent prefix (MIP) of \( X \) is

\[
\emptyset, \text{ if } X \text{ is frequent}
\]

\[
X_i, \text{ s.t. } X_i \text{ infrequent and } X_j \text{ frequent, } j < i.
\]

Thus, an early termination condition for the processing of an itemset \( X \) via the ECS operator is finding the maximal infrequent prefix of \( X \).

**Example 2:** Given a minimal support of 10 and \( X = \{2,3,4,5,6\} \). Assume that the following supports have been calculated: \( \text{sup}\{2\} = 88, \text{sup}\{2,3\} = 51, \text{sup}\{2,3,4\} = 9, \text{sup}\{2,3,4,5\} = 7, \text{sup}\{2,3,4,5,6\} = 1 \). Thus the MIP of \( X \) is \{2,3,4\} with the corresponding support 9. Once this prefix is found, it is not necessary to probe the remaining elements of \( X \), namely 5 and 6, as at this point it is known that \( X \) is infrequent. ❑

With top-down pruning as described in the previous section, once a superset of an itemset \( X \) has been found frequent, we could prune \( X \) together with all its direct subsets, as they are also all frequent. This is not always possible in BUP. More precisely, if a subset \( Y \) of an itemset \( X \) is found infrequent, we can prune \( X \), but not all direct subsets of \( X \), as they might not include \( Y \).

**Example 3:** Assuming that in the backward exploration from Fig. 2, the MIP of itemset \{23456\} is found to be \{23\}, also \{12356\} is infrequent and can be pruned. However, from its direct subsets only \{2356\} contains \{23\}, while the others still have to be explored. ❑

**Theorem 7:** In bottom-up pruning, a maximal infrequent prefix \( X \) can prune an itemset \( Y \) in the search space together with its direct subsets if \( X \subset Y \) and \( \text{ElimList}_Y \) doesn’t contain any items from \( X \). We will further refer to this as **Condition (2)**.

**Example 4:** Suppose that in the backward exploration from Fig. 2, the MIP of itemset \{456\} is found to be \{4\}. In this case, \{4\} can prune the whole branch rooted at \{1246\} since the ElimList of this item is \{12\} and thus every direct subset also includes \{4\}. ❑

Similar to top-down pruning, in order to incorporate also unexpanded itemsets in the BUP, it is necessary to keep a **set of infrequent itemsets** that are relevant for the direct subsets of a given itemset \( X \), called \( IF_X \).

A given prefix can be evaluated multiple times, within different itemsets.

**Theorem 8:** Given a finite item domain \( 1,2,\ldots,N \). In the BE scenario the last time a prefix \( P = \{P_0, P_1,\ldots, P_n\} \) is evaluated is within the itemset \( X = \{P_0, P_1,\ldots, P_n, N\} \).

We will further refer to the prefix \( X \setminus \{N\} \) of an itemset \( X \) as \( \text{PMax}_X \). From Theorem 8 results that given an infrequent itemset \( X \), its maximal infrequent prefix MIP can only be a subset of an itemset that is not explored yet if \( |\text{MIP}| < |X| - 1 \). We will further call this formula **Condition (3)**.

Indeed if \( |\text{MIP}| = |X| - 1 \), then \( \text{MIP} = \text{PMax}_X \) and according to Theorem 8 this prefix will not be expanded further on.

**Example 5:** If the MIP of itemset \( X = \{456\} \) is \{45\}, there is no sense to perform bottom-up pruning with this prefix, as it is not included in any itemset still to be explored. ❑

Another important result of Theorem 8 is that if \( |\text{MIP}| = |X| \), the prefix \( \text{PMax}_X \) is also a maximal frequent itemset. We will formulate this condition **Condition (4)** in the algorithm description given in the appendix.

Indeed, from \( X = \{1,2,\ldots,N\} \) and \( X \) is infrequent and \( |\text{MIP}| = |X| \), i.e. \( \text{MIP} = X \), results that \( \text{PMax}_X = \{1, 2, \ldots, N-1\} \) is frequent. According to Theorem 8, there is no other itemset in the search space still to be explored that includes \( \text{PMax}_X \). On the other hand, there is also no other superset of \( \text{PMax}_X \) explored earlier that has been found frequent, as in this case \( X \) would have been eliminated by top-down pruning.

From this results that \( \text{PMax}_X \) is a maximal frequent itemset.
4.2 Forward Exploration (FE) of Itemsets

From Theorem 3 results that in the FE scenario, an itemset $A$ can be a subset of an itemset $B$ that will be explored later. If both $A$ and $B$ are found frequent, $A$ cannot be a MFI. Thus, contrary to BE, in the FE scenario it is possible to generate also frequent itemsets that are not maximal. Hence it is necessary to have some filter mechanisms that return only MFI. This can be realized by e.g. explicitly maintaining a set of maximal frequent itemsets throughout the exploration. Once a frequent itemset $X$ is found, it is added to this list and eventual subsets of $X$ have to be eliminated, if existing. Thus, at the end of the algorithm the set contains the MFS only.

Similar techniques are used also in [Ba98, LK98]. The disadvantage of this approach is that in this way the maximal frequent itemsets can only be returned when the whole search space exploration is finished. More precisely, if FE is realized within the database engine, this would yield a blocking boundary, as all input has to be processed before the first output tuple, i.e. MFI, is delivered. In the BE scenario this is not necessary, since once an itemset is found to be frequent, it can immediately be returned, as Theorem 3 guarantees that it is also maximal.

We will further concentrate on pruning possibilities for the FE scenario. DTDP can be realized in the same way as described for BE. However, according to Theorem 3, in the FE scenario no itemset explored at a given time has cross subsets that are expanded later. Hence, CTDP is not applicable at all.

4.2.1 Bottom-Up Pruning

In the FE scenario, either entire itemsets or maximal infrequent prefixes can be used for BUP. However, contrary to BE, if we use entire itemsets for pruning, we cannot simply discard an itemset from exploration if one of its subsets is found infrequent. As shown in Section 4.1.2, each itemset $X$ in the search space stands in reality for two itemsets, namely $X$ and $PMAX_X = X \setminus \{N\}$. If we find an itemset $Y = \{Y_0, Y_1, ..., N\}$ to be infrequent and $Y \subseteq X$, we can discard $X$ from evaluation, but we still have to evaluate $PMAX_X$, as this itemset is not a superset of $Y$. Only if $N \notin Y$, $X$ can be totally discarded from evaluation.
Example 6: If in the forward exploration from Fig. 2 itemset \{246\} is infrequent, it can prune \{12456\} from evaluation, but it is still necessary to evaluate its prefix \{1245\}. However if the MIP of \{246\} is \{24\}, \{1245\} is infrequent and can be pruned as well.

In the backward evaluation scenario, we don’t have to consider this problem, because as shown in Section 4.1.2, only maximal infrequent prefixes can be used for BUP. Please note that the MFSSearch algorithm for both the backward and forward exploration scenarios is given in the appendix.

4.3 Summary

As detailed in the previous sections and shown in Fig. 3, the following pruning techniques have been developed for efficient generation of maximal frequent itemsets:

- **Direct Top-Down Pruning (DTDP)** prunes the direct subsets of an itemset \(X\). The technique ensures that these subsets will only be expanded and explored if \(X\) is infrequent. DTDP is applicable to both backward and forward exploration strategies.

- **Cross Top-Down Pruning (CTDP)** ensures that unexplored cross subsets of frequent itemsets are eliminated from exploration. CTDP makes use of a list of relevant frequent itemsets assigned to each expanded itemset, called *FrequentSet*, that is propagated selectively towards not yet explored subsets. It is only applicable to the BE scenario.

- **Bottom-Up Pruning (BUP)** eliminates supersets of infrequent itemsets from exploration. Analogously to CTDP, BUP makes use of a list of relevant infrequent itemsets. BUP is applicable to both exploration strategies. However, a tailoring to the associated strategy has to be provided.

A summarization of the pruning techniques and their application to BE and FE is given in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>Backward Exploration (BE)</th>
<th>Forward Exploration (FE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTDP ((X=\text{Frequent}))</td>
<td>• prunes direct subsets of (X)</td>
<td>• prunes direct subsets of (X)</td>
</tr>
</tbody>
</table>
| CTDP \((X=\text{Frequent})\) | • adds \(X\) to *FrequentSet*(\(Z\)), if \(Z\) expands a subset of \(X\) (Cond. 1)  
  • prunes \(Y\) and its direct subsets, if \(Y \subset X\)  
  | • not applicable                                                              |
| BUP \((X=\text{Infrequent})\) | • only possible if \(X\) is not an entire itemset (Cond. 3)  
  • if \(X \subset Y\), prunes \(Y\); direct subsets of \(Y\) are pruned only if Cond. 2 satisfied;  
  else adds \(X\) to *InfrequentSet*(\(Y\))  
  | • if \(X \subset Y\), but \(X \not\subset PMax_y\), prunes only \(Y\)  
  direct subsets of \(Y\) are pruned only if Cond. 2 satisfied;  
  else adds \(X\) to *InfrequentSet*(\(Y\))  
  • if \(X \subset PMax_y\), prunes also \(PMax_y\) |

5 Performance evaluation

For the performance evaluation of the MFSSearch algorithm we have used the MIDAS database prototype [BJ+96]. We have validated our approach using a 100 MB database, running on a SUN-ULTRA1
workstation with a 143 MHz Ultra Sparc processor. The item domain considered is from 1 to 7, contained in 67,806 transactions. In Fig. 4 we have shown the times that are necessary to derive the entire MFS for both the backward as well as forward exploration scenarios and varying supports.

As results from Fig. 4a, BE is more efficient for lower supports. The reason for this is that in this case we have a large number of long MFI s that can be used for top-down pruning. However, in the FE scenario, CTDP is not possible. In contrast, in the domain of higher supports and implicitly large number of infrequent itemsets BUP is most effective. In this case FE is slightly better than BE, resulting from the fact that in the BE scenario only prefixes can be used for BUP. In order to compare these performances with the Apriori algorithm that is the basis of most bottom-up approaches, we have also presented the time that is necessary to perform the multiple database scans specific to this algorithm. Please note that this curve doesn’t comprise any CPU costs that are also inherent to the Apriori algorithm. As can be seen in Fig. 4a, both variants of the MFSSearch processing scheme show a performance that is orders of magnitude better than the Apriori algorithm.

As shown in [NRM99] MFSSearch outperforms also solutions using top-down search strategies [Ba98, LK98, ZP+97] as well. This result is also influenced by the fact that in contrast to other approaches, MFSSearch reduces I/O costs by accessing the database only selectively, corresponding to the current search space status.

When integrated into the database engine, we expect that the BE scenario achieves generally the best performance. This results on one hand from the fact that it yields a non-blocking processing, hence rapid response times. On the other hand, this strategy combines both pruning strategies to achieve an efficient reduction of the search space for any support values, as shown in the following.

A detailed analysis on the effectiveness of the different pruning techniques for the BE scenario is given in Fig. 4b. Obviously, top-down pruning is most effective for lower supports, where large maximal frequent itemsets can prune several subsets, these being also frequent. Starting with a support of 20-25%, DTDP does not come to application at all. As for CTDP this point is reached with a support of ca. 50%. BUP is most effective with higher supports. The reason for this is that the higher the support, the more infrequent itemsets are found, that in turn can prune their supersets. The bottom curve in Fig. 4b shows that the best performance is achieved by the combination of all three pruning techniques. This leads to overall response times that are only proportional to the volume of maximal frequent itemsets. Additionally, in contrast to randomized algorithms, from Theorem 1 results that MFSSearch is also complete.

![Fig. 4: Effectiveness of pruning](image-url)
6 Conclusions

We have presented a processing scheme for the generation of the maximal frequent sets that employs a new operator, called ECS. This scheme avoids expensive database scans and thus drastically reduces the I/O costs as compared to conventional data mining algorithms. Only itemsets that are not supersets of any known infrequent itemsets or subsets of any known frequent itemsets are considered. As a result, the number of candidate itemsets considered is proportional only to the actual number of maximal frequent itemsets. Hence, the algorithm is also applicable for large item domains. Another possibility is to use MFSSearch for hybrid solutions as well, e.g. to restrict the considered item domain by means of sampling [FS+98]. MFSSearch is complete in the sense that it guarantees that all MFIs are derived. By using our candidate generation algorithm with a backward exploration of itemsets, any frequent itemset found is also a MFI. Thus it can immediately be returned to the user, yielding a non-blocking execution and thus short response times. The underlying theory and its applicability to MFSSearch was the focus of this paper. Another paper [NRM99] details on how the entire processing scheme, i.e. ECS operator combined with MFSSearch, can be efficiently integrated into a database engine, thus being able to make profit of all forms of query execution optimizations, including parallelization [NRM99].

Literature

HGY98 J. Han, W. Gong, Y. Yin: Mining Segment-Wise Periodic Patterns in Time Related Databases, In: Proc. Intl. Conf. on Knowledge Discovery and Data Mining, New York City, NY, August 1998.
Appendix

Proof to Theorem 1: Obviously, for an itemset on level \( N \) all subsets of level \( N-1 \) are generated if its ElimList contains all \( N \) elements. It is not necessary to expand its prefix of length \( N-1 \) since it is implicitly evaluated through ECS. Hence, it is sufficient to include the first \( N-1 \) elements into the ElimList. Consider a superset \( X \) on level \( N \) with the ElimList \( E_X = \{1,2,...,N-1\} \) and the sibling subsets (on level \( N-1 \)) \( X_1, X_2, ..., X_{N-1} \). In this case, subset \( X_i \) is generated by eliminating item \( N-i \) from \( X \), s.t. \( X_i = X - \{N-i\} \). In this case we will demonstrate that any item \( y \) in the ElimList of \( X_i \), s.t. \( y > N-i \), generates only a duplicate subset on level \( N-2 \).

This subset, named e.g. \( X_{i,N-y} \), is obtained by eliminating \( y \) from \( X_i \):

\[
X_{i,N-y} = X_i - \{y\} = X - \{N-i, y\}.
\]

On the other hand, there exists a sibling of \( X_i \) called \( X_{N,y} \) such that \( X_{N,y} = X - \{y\} \). Since \( y > N-i \), results that \( X_{N,y} \) has been expanded before \( X_i \) and that the ElimList of \( X_{N,y} \) also contains \( N-i \). Results that \( X_{N,y} \) has already expanded a subset \( X_{N,y,i} \) on level \( N-2 \) such that \( X_{N,y,i} = X_{N,y} - \{N-i\} = X - \{y, N-i\} \).

Thus, \( X_{i,N-y} = X_{i,N-y} \) and \( X_{i,N-y} \) is expanded after \( X_{i,N-y} \). Results that \( X_{i,N-y} \) is a duplicate.

Proof to Theorem 2: Assume that all itemsets are expanded. In the top-down search, this process is done from higher levels to lower ones. The total number of elements on level \( i \) is \( \binom{1}{N} \). However, not all of these elements need to be expanded, as some of them have already been processed as prefixes in level \( i+1 \). Thus, the elements that need to be expanded on level \( i \) is \( \binom{i}{N} - \binom{1}{N} \). Hence, for \( N \) items, the number of expanded itemsets is given by:

\[
\binom{N}{N} + \binom{N-1}{N} + \binom{N-2}{N} + \binom{N-3}{N} + ... \binom{1}{N} + \binom{-2}{N} + \binom{-1}{N} = \binom{N}{N} + \binom{N-2}{N} + ... + \binom{-1}{N} = 2^{N-1}.
\]

Proof to Theorem 3: By successively reducing the size of the ElimList as described in the Expand procedure, the direct subsets of an itemset \( X \) are only those that have not been expanded before by another sibling. From the nature of top-down processing, the direct subsets of an itemset \( X \) will be processed after \( X \). The cross subsets are related to siblings of \( X \) that are expanded before \( X \). As in the backward processing siblings are processed in the opposite order than they are expanded, results that also these cross subsets of \( X \) will be processed after \( X \).

Proof to Theorem 4: For the domain \( 1,2,...,N-1,N \), the last itemset to be expanded is itemset \( Z = \{N\} \). However, \( Z \) is a subset of any itemset in the search space.

Proof to Theorem 5: Assume \( X_1, X_2 \in F \) and \( X_1 \subset X_2 \). From Theorem 3 follows that \( X_1 \) will be processed later than \( X_2 \). But \( X_2 \) is already frequent, from which results that it prunes \( X_1 \), so that \( X_1 \) cannot be also included in \( F \). Contradiction.

Proof to Theorem 6: As presented at the beginning of this section, the subsets of \( Y \) are obtained by eliminating elements of \( E \) from \( Y \). If there is one element \( y \in Y, y \notin X \), so that \( y \notin E \), results that \( y \) will be present in all direct subsets of \( Y \). Follows that all subsets of \( Y \) contain one element that is not included in \( X \). Thus they cannot be subsets of \( X \).

Proof to Theorem 7: The direct subsets of \( Y \) are obtained by eliminating elements of \( \text{ElimList}_Y \) from \( Y \). From \( X \) is included in \( Y \), and \( \text{ElimList}_Y \) doesn’t contain any elements from \( X \), results that all direct subsets of \( Y \) also contain \( X \). Since \( X \) is infrequent, these direct subsets can also be pruned.

Proof to Theorem 8: Assume that there exists an itemset \( Y = \{P_0, P_1, ..., P_p, P_{n+1}, ..., N\} \) that is expanded after \( X \). But \( X \subset Y \), s.t. according to Theorem 3, \( X \) must be expanded after \( Y \). Contradiction.
The MFSSearch algorithm for the backward exploration scenario:

**MFSSearch** algorithm for a finite item domain 1,2,...,N
1. $X := \{1,2,...,N\}$; $E_X := \{1,2,...,N-1\}$; $F_X := \emptyset$; $IF_X := \emptyset$.
2. get MIP, sup from ECS(X)
3. if ($\text{sup} < \text{minsup}$) // X infrequent, MIP = PMax
   4. if ($|\text{MIP}| < |X| - 1$) // Condition (3)
      5. $IF_X := IF_X \cup \text{MIP}$; // Propagate Relevant Infrequent Itemset
   6. else if ($|\text{MIP}| = |X|$) // Condition (4)
      7. return $X \setminus \{N\}$; // $X \setminus \{N\}$ Maximal Frequent Itemset
8. Expand($X, E_X, F_X, IF_X$);

**Expand**(Itemset $X$, ElimList $E$, FrequencySet $F$, InfrequentSet $IF$)
1. for each $i = 1, n-1$ ($n =$ size of ElimList)
2. $X_i := X \setminus \{e_{n-i}\}$;
3. if ($\exists F_{X_i} \in F, X_i \subset X$) // Cross Top-Down Pruning
4. $X_i := \emptyset$;
5. else
6. $E_i := E \setminus \{e_{n-i},...,e_{n-1}\}$;
7. $F_i := \emptyset$; $IF_i := \emptyset$;
8. for each $F_{X_i} \in F$
   9. if condition (1) // Condition (1)
      10. $F_i := F_i \cup F_{X_i}$; // Propagate Relevant Frequent Itemsets
11. for each $IF_{X_i} \in I$, $IF_{X_i} \subset X_i$
12. if condition (2) // Condition (2)
13. $X_i := \emptyset$; // Bottom-Up Pruning affecting also direct subsets
14. else
15. Mark $X_i$ as infrequent; // Bottom-Up Pruning affecting only the itemset
16. $IF_i := IF_i \cup IF_{X_i}$; // Propagate Relevant Infrequent Itemsets
17. for each $X_i \neq \emptyset$, $i := n-1, 1$ // Backward Exploration
18. if $\neg\text{Infrequent}(X_i)$$
19. get MIP, sup from ECS($X_i$)
20. if ($\text{sup} < \text{minsup}$) // $X_i$ infrequent, MIP
21. if ($|\text{MIP}| < |X| - 1$) // Condition (3)
22. BottomUp (MIP);
23. else if ($|\text{MIP}| = |X_i|$) // Condition (4)
24. return $X_i \setminus \{N\}$; // $X_i \setminus \{N\}$ Maximal Frequent Itemset
25. Expand($X_i, E_i, F_i, IF_i$);
26. else
27. CrossTopDown($X_i$);
28. return $X_i$; // Maximal Frequent Itemset
29. else Expand($X_i, E_i, F_i, IF_i$);

**CrossTopDown**(Frequent_Itemset $X$)
1. for each candidate $Y$, $Y$ expanded but unexplored
2. if condition (1)
3. $F_Y := F_Y \cup X$;

**BottomUp**(Infrequent_Itemset $I$)
1. for each candidate $Y$, $Y$ expanded but unexplored
2. if $I \subset Y$
3. if condition (2)
4. $Y := \emptyset$; // prune $Y$ together with its direct subsets
5. else
6. Mark $Y$ as Infrequent; // prune only $Y$
7. $IF_Y := IF_Y \cup I$; // propagate $I$ to be taken into account for subsets of $Y$

As can be seen from MFSSearch, the procedure Expand is used to address both pruning and search strat-
egies within the given search space.

The **Expand procedure adapted to the forward exploration scenario:**

**Expand**(Itemset $X$, ElimList $E$, InfrequentSet $IF$)
1. for each $i = 1, n-1$ ($n = size$ of ElimList)
2. $X_i := X \setminus \{e_{n-i}\}$
3. $E_i := E \setminus \{e_{n-i}, ..., e_{n-1}\}$
4. $IF_i := \emptyset$
5. for each $IF_X \in I$, $IF_X \subset X$
6. if condition (2)
7.   $X_i := \emptyset$; // Bottom-Up Pruning affecting also direct subsets
8. else
9.   Mark $X_i$ as infrequent; // Bottom-Up Pruning affecting only the itemset
10. if $I \subset PMAX_X$
    // prune also PMax of $X$
11.   Mark $PMAX_X$ as infrequent;
12.   $IF_i := IF_i \cup IF_X$; // Propagate Relevant Infrequent Itemsets
13. for each $X_i \neq \emptyset$, $i = 1, n-1$
14. if $\neg$Infrequent($X_i$)
    // probe $X$
15.   get MIP, sup from ECS($X_i$);
16. if (sup < minsup)
    // $X_i$ infrequent, MIP = PMax
17.   BottomUp (MIP);
18. if $([MIP] = |X_i|)$
    // condition (4)
19.   UpdateMFS($X_i \setminus \{N\}$); // $X_i \setminus \{N\}$ Frequent Itemset
20. Expand($X_i, E_i, IF_i$);
21. else
22.   UpdateMFS($X_i$); // $X_i$ Frequent Itemset
23. else
24. if $\neg$Infrequent($X_i \setminus \{N\}$)
    // probe PMax of $X$
25.   get MIP, sup from ECS($X_i \setminus \{N\}$);
26. if (sup > minsup)
    $\neg$Infrequent($X_i \setminus \{N\}$);
27.   UpdateMFS($X_i \setminus \{N\}$); // $X_i \setminus \{N\}$ Frequent Itemset
28. Expand($X_i, E_i, IF_i$);

**UpdateMFS**(Frequent Itemset $X$)
1. for each $Y \in MFS$
2. if $Y$ is a subset of $X$
3.   eliminate $Y$;
4. $MFS := MFS \cup X$;

**BottomUp**(Infrequent Itemset $I$)
1. for each candidate $Y$, $Y$ expanded but unexplored
2. if $I \subset Y$
3. if condition (2)
4. $Y := \emptyset$; // prune $Y$ together with its direct subsets
5. else
6.   Mark $Y$ as infrequent; // prune only $Y$
7. if $I \subset PMAX_Y$
8.   Mark $PMAX_Y$ as infrequent; // prune also $PMAX_Y$
9. $IF_Y := IF_Y \cup I$; // propagate $I$ to be taken into account for subsets of $Y$
SFB 342: Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

bisher erschienen:

Reihe A

Liste aller erschienenen Berichte von 1990-1994 auf besondere Anforderung

342/01/95 A Hans-Joachim Bungartz: Higher Order Finite Elements on Sparse Grids
342/02/95 A Tao Zhang, Seonglim Kang, Lester R. Lipsky: The Performance of Parallel Computers: Order Statistics and Amdahl's Law
342/03/95 A Lester R. Lipsky, Appie van de Liefvoort: Transformation of the Kronecker Product of Identical Servers to a Reduced Product Space
342/04/95 A Pierre Fiorini, Lester R. Lipsky, Wen-Jung Hsin, Appie van de Liefvoort: Auto-Correlation of Lag-k For Customers Departing From Semi-Markov Processes
342/05/95 A Sascha Hilgenfeldt, Robert Balder, Christoph Zenger: Sparse Grids: Applications to Multi-dimensional Schrödinger Problems
342/06/95 A Maximilian Fuchs: Formal Design of a Model-N Counter
342/07/95 A Hans-Joachim Bungartz, Stefan Schulte: Coupled Problems in Microsystem Technology
342/08/95 A Alexander Pfaffinger: Parallel Communication on Workstation Networks with Complex Topologies
342/09/95 A Ketil Stølen: Assumption/Commitment Rules for Data-flow Networks - with an Emphasis on Completeness
342/10/95 A Ketil Stølen, Max Fuchs: A Formal Method for Hardware/Software Co-Design
342/11/95 A Thomas Schmekenburger: The ALDY Load Distribution System
342/12/95 A Javier Esparza, Stefan Römer, Walter Vogler: An Improvement of McMillan's Unfolding Algorithm
342/13/95 A Stephan Melzer, Javier Esparza: Checking System Properties via Integer Programming
342/14/95 A Radu Grosu, Ketil Stølen: A Denotational Model for Mobile Point-to-Point Dataflow Networks
342/15/95 A Andrei Kovalyov, Javier Esparza: A Polynomial Algorithm to Compute the Concurrency Relation of Free-Choice Signal Transition Graphs
342/16/95 A Bernhard Schätz, Katharina Spies: Formale Syntax zur logischen Kernsprache der Focus-Entwicklungsmethodik
342/17/95 A Georg Stellner: Using CoCheck on a Network of Workstations
342/18/95 A Arndt Bode, Thomas Ludwig, Vaidy Sunderam, Roland Wismüller: Workshop on PVM, MPI, Tools and Applications
342/19/95 A Thomas Schmekenburger: Integration of Load Distribution into ParMod-C
342/20/95 A Ketil Stølen: Refinement Principles Supporting the Transition from Asynchronous to Synchronous Communication
Reihe A

342/21/95 A Andreas Listl, Giannis Bozias: Performance Gains Using Subpages for Cache Coherency Control
342/22/95 A Volker Heun, Ernst W. Mayr: Embedding Graphs with Bounded Treewidth into Optimal Hypercubes
342/23/95 A Petr Jančář, Javier Esparza: Deciding Finiteness of Petri Nets up to Bisimulation
342/24/95 A M. Jung, U. Rüde: Implicit Extrapolation Methods for Variable Coefficient Problems
342/01/96 A Michael Griebel, Tilman Neumhoeffer, Hans Regler: Algebraic Multigrid Methods for the Solution of the Navier-Stokes Equations in Complicated Geometries
342/02/96 A Thomas Grauschopf, Michael Griebel, Hans Regler: Additive Multilevel Preconditioners based on Bilinear Interpolation, Matrix Dependent Geometric Coarsening and Algebraic-Multigrid Coarsening for Second Order Elliptic PDEs
342/03/96 A Volker Heun, Ernst W. Mayr: Optimal Dynamic Edge-Disjoint Embeddings of Complete Binary Trees into Hypercubes
342/04/96 A Thomas Huckle: Efficient Computation of Sparse Approximate Inverses
342/05/96 A Thomas Ludwig, Roland Wismüller, Vaidy Sunderam, Arndt Bode: OMIS — On-Line Monitoring Interface Specification
342/06/96 A Eckhart Kindler: A Compositional Partial Order Semantics for Petri Net Components
342/07/96 A Richard Mayr: Some Results on Basic Parallel Processes
342/08/96 A Ralph Radermacher, Frank Weimer: INSEL Syntax-Bericht
342/10/96 A Stefan Lamberts, Thomas Ludwig, Christian Röder, Arndt Bode: PFS-Lib — A File System for Parallel Programming Environments
342/11/96 A Manfred Broy, Gheorghe Ștefănescu: The Algebra of Stream Processing Functions
342/12/96 A Javier Esparza: Reachability in Live and Safe Free-Choice Petri Nets is NP-complete
342/13/96 A Radu Grosu, Ketil Stølen: A Denotational Model for Mobile Many-to-Many Data-flow Networks
342/14/96 A Giannis Bozias, Michael Jaedicke, Andreas Listl, Bernhard Mitschang, Angelika Reiser, Stephan Zimmermann: On Transforming a Sequential SQL-DBMS into a Parallel One: First Results and Experiences of the MIDAS Project
342/15/96 A Richard Mayr: A Tableau System for Model Checking Petri Nets with a Fragment of the Linear Time μ-Calculus
342/16/96 A Ursula Hinkel, Katharina Spies: Anleitung zur Spezifikation von mobilen, dynamischen Focus-Netzen
342/17/96 A Richard Mayr: Model Checking PA-Processes
342/18/96 A Michaela Huhn, Peter Niebert, Frank Wallner: Put your Model Checker on Diet: Verification on Local States
342/01/97 A Tobias Müller, Stefan Lamberts, Ursula Maier, Georg Stellner: Evaluierung der Leistungsfähigkeit eines ATM-Netzes mit parallelen Programmierbibliotheken
Reihe A

342/02/97 A  Hans-Joachim Bungartz and Thomas Dornseifer: Sparse Grids: Recent Developments for Elliptic Partial Differential Equations
342/03/97 A  Bernhard Mitschang: Technologie für Parallele Datenbanken - Bericht zum Workshop
342/04/97 A  nicht erschienen
342/05/97 A  Hans-Joachim Bungartz, Ralf Ebner, Stefan Schulte: Hierarchische Basen zur effizienten Kopplung substrukturierter Probleme der Strukturmechanik
342/06/97 A  Hans-Joachim Bungartz, Anton Frank, Florian Meier, Tilman Neunhoef er, Stefan Schulte: Fluid Structure Interaction: 3D Numerical Simulation and Visualization of a Micropump
342/07/97 A  Javier Esparza, Stephan Melzer: Model Checking LTL using Constraint Programming
342/08/97 A  Niels Reimann: Untersuchung von Strategien für verteiltes Last- und Ressourcenmanagement
342/09/97 A  Markus Pizka: Design and Implementation of the GNU INSEL-Compiler
342/10/97 A  Manfred Broy, Franz Regensburger, Bernhard Schätz, Katharina Spies: The Steamboller Specification - A Case Study in Focus
342/11/97 A  Christine Röckl: How to Make Substitution Preserve Strong Bisimilarity
342/12/97 A  Christian B. Czech: Architektur und Konzept des Dycos-Kerns
342/13/97 A  Jan Philipps, Alexander Schmidt: Traffic Flow by Data Flow
342/14/97 A  Norbert Fröhlich, Rolf Schlegelbach, Josef Fleischmann: Partitioning VLSI-Circuits for Parallel Simulation on Transistor Level
342/15/97 A  Frank Weimer: DaViT: Ein System zur interaktiven Ausführung und zur Visualisierung von INSEL-Programmen
342/16/97 A  Niels Reimer, Jürgen Rudolph, Katharina Spies: Von FOCUS nach INSEL - Eine Aufzugssteuerung
342/17/97 A  Radu Gresu, Ketil Stolen, Manfred Broy: A Denotational Model for Mobile Point-to-Point Data-flow Networks with Channel Sharing
342/18/97 A  Christian Röder, Georg Stellner: Design of Load Management for Parallel Applications in Networks of Heterogenous Workstations
342/19/97 A  Frank Wallner: Model Checking LTL Using Net Unfoldings
342/20/97 A  Andreas Wolf, Andreas Knoch: Einsatz eines automatischen Theorem-beweisers in einer taktikgesteuerten Beweisumgebung zur Lösung eines Beispiels aus der Hardware-Verifikation - Fallstudie -
342/21/97 A  Andreas Wolf, Marc Fuchs: Cooperative Parallel Automated Theorem Proving
342/22/97 A  T. Ludwig, R. Wismüller, V. Sunderam, A. Bode: OMIS - On-line Monitoring Interface Specification (Version 2.0)
342/23/97 A  Stephan Merkle: Verification of Fault Tolerant Algorithms Using PEP
342/24/97 A  Manfred Broy, Max Breitling, Bernhard Schätz, Katharina Spies: Summary of Case Studies in Focus - Part II
342/25/97 A  Michael Jaeckle, Bernhard Mitschang: A Framework for Parallel Processing of Aggregate and Scalar Functions in Object-Relational DBMS
342/26/97 A  Marc Fuchs: Similarity-Based Lemma Generation with Lemma-Delaying Tableau Enumeration
<table>
<thead>
<tr>
<th>Date</th>
<th>Authors/Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>342/27/97</td>
<td>Max Breitling: Formalizing and Verifying TimeWarp with FOCUS</td>
</tr>
<tr>
<td>342/29/97</td>
<td>Radu Grosu, Ketil Stolen: Compositional Specification of Mobile Systems</td>
</tr>
<tr>
<td>342/02/98</td>
<td>Ursula Hinkel: Home Shopping - Die Spezifikation einer Kommunikationsanwendung in FOCUS</td>
</tr>
<tr>
<td>342/03/98</td>
<td>Katharina Spies: Eine Methode zur formalen Modellierung von Betriebssystemkonzepten</td>
</tr>
<tr>
<td>342/04/98</td>
<td>Stefan Bischof, Ernst W. Mayr: On-Line Scheduling of Parallel Jobs with Runtime Restrictions</td>
</tr>
<tr>
<td>342/05/98</td>
<td>St. Bischof, R. Ebner, Th. Erlebach: Load Balancing for Problems with Good Bisectors and Applications in Finite Element Simulations: Worst-case Analysis and Practical Results</td>
</tr>
<tr>
<td>342/06/98</td>
<td>Giannis Bozas, Susanne Kober: Logging and Crash Recovery in Shared-Disk Database Systems</td>
</tr>
<tr>
<td>342/07/98</td>
<td>Markus Pizka: Distributed Virtual Address Space Management in the MoDiS-OS</td>
</tr>
<tr>
<td>342/08/98</td>
<td>Nieß Reimer: Strategien für ein verteiltes Last- und Ressourcenmanagement</td>
</tr>
<tr>
<td>342/10/98</td>
<td>Richard Mayr: Lossy Counter Machines</td>
</tr>
<tr>
<td>342/11/98</td>
<td>Thomas Huckle: Matrix Multilevel Methods and Preconditioning</td>
</tr>
<tr>
<td>342/12/98</td>
<td>Thomas Huckle: Approximate Sparsity Patterns for the Inverse of a Matrix and Preconditioning</td>
</tr>
<tr>
<td>342/13/98</td>
<td>Antonin Kucera, Richard Mayr: Weak Bisimilarity with Infinite-State Systems can be Decided in Polynomial Time</td>
</tr>
<tr>
<td>342/01/99</td>
<td>Antonin Kucera, Richard Mayr: Simulation Preorder on Simple Process Algebras</td>
</tr>
<tr>
<td>342/02/99</td>
<td>Johann Schumann, Max Breitling: Formalisierung und Beweis einer Verfeinerung aus FOCUS mit automatischen Theorembeweisern – Fallstudie –</td>
</tr>
<tr>
<td>342/03/99</td>
<td>M. Bader, M. Schimper, Chr. Zenger: Hierarchical Bases for the Infinite Helmholtz Equation</td>
</tr>
<tr>
<td>342/04/99</td>
<td>Frank Strobl, Alexander Wisspeintner: Specification of an Elevator Control System</td>
</tr>
<tr>
<td>342/05/99</td>
<td>Ralf Ebner, Thomas Erlebach, Andreas Ganz, Claudia Gold, Clemens Harflinger, Roland Wismüller: A Framework for Recording and Visualizing Event Traces in Parallel Systems with Load Balancing</td>
</tr>
</tbody>
</table>
Reihe A

342/07/99 A Max Breitling, Jan Philipp: Black Box Views of State Machines
342/08/99 A Clara Nippl, Stephan Zimmermann, Bernhard Mitschang: Design, Implementation and Evaluation of Data Rivers for Efficient Intra-Query Parallelism
342/09/99 A Robert Sandner, Michael Maurer: Integrierte Beschreibung automatisierter Produktionsanlagen - eine Evaluierung praxisnaher Beschreibungstechniken
342/10/99 A Alexander Sabbah, Robert Sandner: Evaluation of Petri Net and Automata Based Description Techniques: An Industrial Case Study
342/01/00 A Javier Esparza, David Hansel, Peter Rossmanith, Stefan Schwoon: Efficient Algorithm for Model Checking Pushdown Systems
342/02/00 A Barbara König: Hypergraph Construction and Its Application to the Compositional Modelling of Concurrency
342/03/00 A Max Breitling and Jan Philipp: Verification Diagrams for Dataflow Properties
342/04/00 A Günther Rackl: Monitoring Globus Components with MIMO
342/05/00 A Barbara König: Analysing Input/Output Capabilities of Mobile Processes with a Generic Type System
342/06/00 A Michael Bader, Christoph Zenger: A Parallel Solver for Convection Diffusion Equations based on Nested Dissection with Incomplete Elimination
342/07/00 A Clara Nippl, Angelika Reiser, Bernhard Mitschang: Extending Database Functionality to Support Frequent Itemset Processing
342/08/00 A Clara Nippl, Angelika Reiser, Bernhard Mitschang: Conquering the Search Space for the Calculation of the Maximal Frequent Set
SFB 342: Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

Reihe B

342/1/90 B Wolfgang Reisig: Petri Nets and Algebraic Specifications
342/2/90 B Jörg Desel: On Abstraction of Nets
342/3/90 B Jörg Desel: Reduction and Design of Well-behaved Free-choice Systems
342/4/90 B Franz Abstreiter, Michael Friedrich, Hans-Jürgen Plewan: Das Werkzeug runtime zur Beobachtung verteilter und paralleler Programme
342/1/91 B Barbara Paech: Concurrency as a Modality
342/2/91 B Birgit Kandler, Markus Pawlowski: SAM: Eine Sortier-Toolbox – Anwenderbeschreibung
342/3/91 B Erwin Loibl, Hans Obermaier, Markus Pawlowski: 2. Workshop über Parallelisierung von Datenbanksystemen
342/4/91 B Werner Pohlmann: A Limitation of Distributed Simulation Methods
342/5/91 B Dominik Gomm, Ekkart Kindler: A Weakly Coherent Virtually Shared Memory Scheme: Formal Specification and Analysis
342/6/91 B Dominik Gomm, Ekkart Kindler: Causality Based Specification and Correctness Proof of a Virtually Shared Memory Scheme
342/7/91 B W. Reisig: Concurrent Temporal Logic
342/1/92 B Malte Grosse, Christian B. Suttner: A Parallel Algorithm for Set-of-Support
            Christian B. Suttner: Parallel Computation of Multiple Sets-of-Support
342/2/92 B Arndt Bode, Hartmut Wedekind: Parallelrechner: Theorie, Hardware, Software, Anwendungen
342/1/93 B Max Fuchs: Funktionale Spezifikation einer Geschwindigkeitsregelung
342/2/93 B Ekkart Kindler: Sicherheits- und Lebensdauereigenschaften: Ein Literaturüberblick
342/1/94 B Andreas Listl, Thomas Schnekenburger, Michael Friedrich: Zum Entwurf eines Prototypen für MIDAS