COLARM: Cost-based Optimization for Localized Association Rule Mining*

Abhishek Mukherji*
Samsung Research America - Silicon Valley
1732 North 1st Street
San Jose, CA 95112.
a.mukherji@samsung.com

Elke A. Rundensteiner
Dept. of Computer Science
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609.
rundenst@cs.wpi.edu

Matthew O. Ward
Dept. of Computer Science
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609.
matt@cs.wpi.edu

ABSTRACT

Association rule mining typically focuses on discovering *global* rules valid across the entire dataset. Yet *local* rules valid for subsets of the dataset, while significantly different from global rules, are often also of tremendous importance to analysts. In this work, we tackle this overlooked problem of *online mining of localized association rules*. We provide support for analysts to interactively mine rules that are hidden in a global context yet are locally significant.

To tackle this problem we design a compact *multidimensional itemset-based data partitioning* (MIP-index). MIP-index offers efficient mining performance by utilizing precomputed results, while still allowing the user the flexibility of selecting any data subset of interest at run-time. We design a suite of alternative execution strategies for processing such localized mining requests. Optimization principles such as *selection push-up*, *supported R-tree filter* and *differential treatment of contained and partially overlapped MIPs* are proposed. We analytically and experimentally demonstrate that different execution strategies are effective for different query scenarios. Given a localized mining query, our COLARM *query optimizer* takes a cost-based approach to identify the best strategy for execution. Through extensive experiments using benchmark data sets we demonstrate that the COLARM optimizer is highly accurate in online plan selection and discovering localized rules (otherwise hidden in the global context) in a diversity of localized mining requests.

1. INTRODUCTION

1.1 Motivation

Mining of association rules and correlations from large datasets has been recognized as an increasingly critical technology for data-intensive decision-making activities [2,10,16,17]. Different aspects of rule mining have received substantial attention. In particular, (a) improving the efficiency of mining algorithms [4, 6]; (b) generating rules at different data granularities [10, 19]; (c) mining rules over heterogeneous data types [20]; (d) defining rule interestingness measures [23]; and, (e) parameter space based interactive rule exploration [13, 15]. All these efforts, including our prior work PARAS [13, 15], focus on discovering *global* rules valid across the entire dataset. Yet local rules valid for subsets of the dataset, while significantly different from global rules, are often also of tremendous importance to analysts. The problem of *online discovery of localized rules* from subsets of data has received little attention.

Company	Title	Location	Gender	Age	Salary
IBM	QA Lead	Boston	M	30-40	60K-90K
IBM	Sw Engg	Boston	F	20-30	90K-120K
IBM	Engg Mgr	SFO	M	20-30	90K-120K
Google	Sw Engg	SFO	F	20-30	90K-120K
Google	Sw Engg	Boston	F	20-30	90K-120K
Google	Sw Engg	Boston	M	20-30	90K-120K
Google	Tech Arch	Boston	M	40-50	120K-150K
Microsoft	Engg Mgr	Seattle	F	30-40	90K-120K
Microsoft	Sw Engg	Seattle	F	30-40	90K-120K
Facebook	QA Mgr	Seattle	F	30-40	90K-120K
Facebook	QA Engg	Seattle	F	20-30	30K-60K

Table 1: The example salary dataset.

Importance of local patterns. *Simpson's paradox* [18] establishes that *local* patterns can be very different from *global* patterns. We illustrate this phenomenon in the context of rule mining. The *salary* dataset (Table 1) shows salary information of anonymized IT employees in different regions of the United States. Based on the complete dataset, a global salary trend given by rule R^G = (*A0* $\rightarrow$ *S2)* [1] can be mined. Rule R^G means that the employees with *age* between 20 and 30 usually have *salaries* ranging between \$90K and \$120K. R^G has 45% support, i.e., it holds true for five out of the total of eleven records. It also has a confidence of 83% (5/6). Next, if an analyst wants to learn the local trends for the *female employees in Seattle* (here, the last four records), a *localized* rule, namely, R^L = (A1 $\rightarrow$ S2) will be discovered with a 75% support and a 100% confidence. Interestingly, the *global* rule R^G does not hold true in this subset.

The above example highlights that *localized rule* R^L will be hidden in the *global* context unless the analyst lowers the minsupport to $<$ 27% (possibly outputting overwhelmingly large set of rules).

*Supported by National Science Foundation under grants IIS-0812027, CCF-0811510 and IIS-1117139.

*This work was done when Abhishek was a graduate student at the Computer Science Department of Worcester Polytechnic Institute.

[1]For rule mining over quantitative data [20], the attribute-value pairs are discretized into disjoint intervals (e.g., Age 20-30,30-40, and so on.). We denote the intervals for each dimension as Age = {A0,A1,A2,...}, Salary = {S0,S1,S2,...}, and so on. Here, A0= (Age in 20$-$30 years) and S2 = (Salary in 90K$-$120K).

Even if, in the *global* context, R^L is discovered as a low support rule, the analyst may not discover that this low ranked rule R^L is prominent in a local context. In general, as trends may vary greatly from region to region, from job to job and across different age groups, *global* patterns may not represent the dataset adequately.

One of the best-known real-life examples of Simpson's paradox occurred when the University of California, Berkeley was sued for bias against women who had applied for admission to graduate schools there. The admission figures for the fall of 1973 showed that men applying were more likely than women to be admitted, and the difference was so large that it was unlikely to be due to chance. But when examining the individual departments, it appeared that no department was significantly biased against women. In fact, most departments had a "small but statistically significant bias in favor of women". Moreover, this problem has also been motivated in some of the prior works [8, 22]. In our experimental evaluation (Section 5.3), we observe strong evidence of such Simpson's paradox in our tested datasets in the context of localized rule mining queries.

Need for a POQM solution. Employing a mining algorithm at query-time on even a moderately sized dataset to generate rules may be prohibitively costly [2, 16]. This holds true even in the context of localized rule mining; as confirmed by our experiments. To maintain user interest and focus, real-time query turnaround times are essential. Clearly, innovations are needed beyond making the rule mining algorithms faster. Towards this end, online mining solutions [2, 13, 15] employ the *preprocess once - query many* (*POQM*) paradigm. The expensive frequent itemset generation step is performed offline using a primary support threshold[2] to prestore frequent itemsets in an itemset-based index. At query-time, rules satisfying user thresholds are generated using the index. In this work we propose to extend the POQM paradigm towards online mining of localized rules. Further, for the localized rule mining scenario we discover that due to uncertain nature of subset selection we cannot guarantee that a POQM solution always outperforms a naive solution of running the rule mining algorithm from scratch on the chosen subset. Thus, there is scope for comparison of alternative execution plans and plan selection using cost-based optimization.

1.2 The State-of-the-Art

Pioneering works in *constraint-based* mining are done by Boulicaut et al. [7, 12]. We leverage from the survey of condensed representations [7] as we propose to prestore closed frequent itemsets for our localized rule mining problem. While [12] presents an algorithm to efficiently manage the constraint-based mining task when the sequential databases contain consecutive repetitions of their items, they do not focus on online mining of rules. Overall, we extend the problem of local pattern detection [14] towards discovering local association rules. Unlike existing works, we explore the problem of online localized rule mining and employ a POQM solution to solve this problem.

Recent works on rule mining over multidimensional windows [8, 22] are closely related. However, their scope is restricted to transactional datasets such as Market Basket. They make the restricting assumption that the partitioning attributes (such as location and time) are distinct from the transactions that compose the itemsets and rules. We, instead, focus on a relational data model where both the subset (equivalent of partitioning attributes) and the itemsets to form rules come from a common pool of attribute-value pairs. Moreover, both are defined at query-time. Hence, our target problem is fundamentally different from prior works [8, 22]. For detailed analysis of related works see Section 6.

[2]Assuming that analysts are not interested in rules with support lower than the *primary support* [2].

1.3 Research Challenges

Our goal of designing a POQM solution for online *localized* rule mining imposes the following research challenges.

Offline data preprocessing. Precomputing *locally* frequent itemsets is complex. For a dataset containing m records, the total number of possible subsets that the analyst can select at query-time is given by the Bell Number [9] as $\sum_{i=0}^{m}\{^m_i\}$, where $\{^m_i\}$ is the Stirling number. Clearly, prestoring all possible partitions and their corresponding frequent itemsets is infeasible. An effective indexing strategy for compactly prestoring itemsets (such as suggested in [7]) is needed to render POQM feasible in a local context.

Online focal subset selection. We denote the subset of interest to the analyst as the *focal subset*. The focal subset is specified as a query-time parameter and may range from as few as a single record to as many as all records of the dataset. Clearly, a versatile online query processing strategy is required that can utilize the precomputed index structure to efficiently identify the focal subset and the candidate frequent itemsets.

Online localized rule verification. Most existing online mining solutions [2, 8] only verify the *minsupport* (as it is relatively inexpensive to compute), while avoiding the costlier *minconfidence* checks. Due to the importance of null-invariant measures [23], we propose to verify both *minsupport* and *minconfidence*. However, as these measures can only be verified at query-time, efficient mechanisms are needed for the same.

1.4 The COLARM Approach

The key contributions of this work are summarized as follows:

• We formulate the *online localized rule mining query*. In addition to the traditional rule interestingness thresholds (*minsupport* and *minconfidence*), a localized rule mining query introduces two new parameters, namely, *range* and *item* to enable subset selection and item attribute specification, respectively (Section 2).

• We introduce the ***M**ultidimensional **I**temset **P**artitioning* index (MIP-index). MIP-index offers efficient mining performance by utilizing precomputed results, while still allowing the flexible query-time selection of data subsets. This is achieved by compactly prestoring two features of the itemsets, namely, (a) the multidimensional bounding box of the itemsets; and, (b) the items composing the itemsets (Section 3).

• We isolate different online mining steps and optimize each of these steps by employing novel optimization principles in the context of online localized rule mining. We construct several alternative *mining plans* by pipelining the different optimized steps. We develop a cost model to estimate benefits and overheads of these plans. We further develop a **C**ost-based **O**ptimizer for **L**ocalized **A**ssociation **R**ule **M**ining, in short **COLARM**, for online selection of the most cost-effective plan (Section 4).

• Our extensive experimental study using benchmarks such as chess, mushroom and PUMSB datasets from UCI ML repository [5] establishes that our COLARM optimizer is highly accurate ($\sim$93%) in selecting the most efficient mining plan for a rich diversity of online mining requests. We also observe strong evidence of Simpson's paradox in the context of localized rule mining (Section 5).

2. PRELIMINARIES

We first describe how we represent itemsets in a multidimensional space and then formulate the *online localized rule mining problem* using the terms defined in Table 2.

2.1 Itemsets in Multidimensional Space

Consider a relational database $\mathcal{D}$ with *n* attributes $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$. Let there be *m* data records in $\mathcal{D}$. Each data record r consists of

Symbol	Definition
n	Total number of attributes in a relation.
minsupp	User-specified min. support.
minconf	User-specified min. confidence.
A^{range}	Range attribute-value pairs in the WHERE clause of Q.
A^{item}	Item attributes specified in the WHERE clause of Q.
$\mathcal{D}^Q$	User-chosen focal subset.
$Supp_I^G$	Global support of an itemset I in the complete dataset $\mathcal{D}$.
$Supp_I^Q$	Local support of an itemset I w.r.t. the subset $\mathcal{D}^Q$.
$Conf_R^G$	Global confidence of Rule R in the complete dataset $\mathcal{D}$.
$Conf_R^Q$	Local confidence of Rule R w.r.t. the subset $\mathcal{D}^Q$.
$\{I_S^Q\}$	Set of candidate itemsets w.r.t. $\mathcal{D}^Q$ output by SEARCH.
$\{I_E^Q\}$	Set of candidate itemsets in $\mathcal{D}^Q$ output by ELIMINATE.
$\{R^Q\}$	Set of local rules in $\mathcal{D}^Q$ output by VERIFY.

Table 2: List of notation.

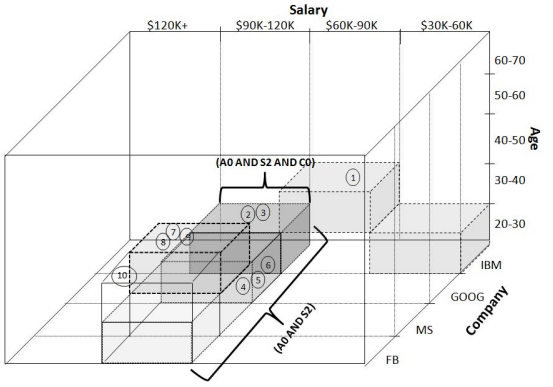


Figure 1: Itemsets in n-Dimensional Space.

value $<v_1,\ldots,v_i,\ldots,v_n>$, where v_i corresponds to attribute $\mathcal{A}_i$ of the *n* attributes. Rule mining [3] works with only nominal attributes, while quantitative attributes are first discretized[3]. For $\mathcal{D}$, a single attribute-value pair ($\mathcal{A}_i = v_i$) forms an *item* and a collection of such items is called an *itemset*. For example, in the salary dataset (Table 1), A0=(Age=20-30) is an *item*. (Age=20-30, Salary=90K-120K), also denoted as (A0,S2), is a *2-itemset* as it consists of two items. Further, (Age=20-30, Salary=90K-120K, Company=IBM), denoted as (A0,S2,C0), is a *3-itemset*.

For ease of depiction, in Figure 1 we assume that the salary dataset consists of three dimensions, namely, age, salary and company. Here, the records {2,3} belong to the cell (A0, S2, C0). The cells (3-itemsets), namely, (A0, S2, C0), (A0, S2, C1), (A0, S2, C2) and (A0, S2, C3) combine to form the 2-itemset (A0,S2). The bounding box for (A0,S2) is shown in Figure 1. This conforms to the *downward closure property*[4] of itemsets that we utilize in our work. While cells (A0, S2, C0) and (A0, S2, C1) contain records {2, 3} and {4, 5, 6} respectively, cells (A0, S2, C2) and (A0, S2, C3) are empty. Thus, in general we divide an n-dimensional space into (n-itemset) *cells* at the lowest granularity. Several i-itemsets can be combined to form each (i-1)-itemset. This bottom up process can be repeated until 2-itemsets are composed.

2.2 Localized Association Mining Problem

An online mining query for *localized* rules can be specified by query Q. Given a dataset $\mathcal{D}$, localized rules valid in the *focal subset* $\mathcal{D}^Q$ are requested. The *range parameter* $\mathcal{A}^{range}$ in the WHERE clause specifies $\mathcal{D}^Q$. For each range attribute $\mathcal{A}_i^{range}$, the user selects values, where $v_{i,p}^{range}$ denotes the p^{th} value.

```
Q: REPORT LOCALIZED ASSOCIATION RULES
FROM Dataset D
WHERE RANGE A^range = { A_1^range = (v_{1,1}^range, ...), ...,
A_k^range = (v_{k,1}^range, ...) }
AND
[ ITEM ATTRIBUTES A^item = {A_1^item, ..., A_f^item} ]
HAVING minsupport = minsupp and
minconfidence = minconf;
```

For the salary dataset (Table 1), (Age={20-30, 30-40} and Company = {IBM}) forms an example focal subset $\mathcal{D}^Q$ denoting the IBM employees between ages 20 and 40. By default $\mathcal{A}_i^{range}$ spans over the domain of $\mathcal{A}_i^{range}$. The users can optionally use the *item* clause ($\mathcal{A}^{item}$) to specify the attributes for generating rules. Therefore, ($\mathcal{A}^{range} \subseteq \mathcal{A}$) and ($\mathcal{A}^{item} \subseteq \mathcal{A}$).

Let the user-defined focal subset be $\mathcal{D}^Q$. Size of the focal subset ($|\mathcal{D}^Q|$) is the number of tuples in the focal subset. It is computed by counting the tuples overlapping with the *range parameter* $\mathcal{A}^{range}$. For an itemset I, if $\mathcal{D}_I^Q$ denotes the records of $\mathcal{D}^Q$ that support I, the *local support* for I ($Supp_I^Q$) = $\frac{|\mathcal{D}_I^Q|}{|\mathcal{D}^Q|}$. Itemset I is *frequent* in $\mathcal{D}^Q$ if $Supp_I^Q \geq$ *minsupp*. Further, for a rule R = (X $\Rightarrow$ Y), where I = (X $\cup$ Y), the *local confidence* $Conf_R^Q$ is given by the fraction $\frac{Supp_I^Q}{Supp_X^Q}$. Only if $Conf_R^Q \geq$ *minconf*, Rule R is included in output $\{R^Q\}$. As *range* and *item* attributes are known only at query-time, offline precomputation of local support and confidence values is not possible. The overall problem is formulated below.

DEFINITION 1. ***Localized Association Mining Problem:*** *Given the range attributes $\mathcal{A}^{range}$, the item attributes $\mathcal{A}^{item}$ and the thresholds, namely, minsupp and minconf, find the set of association rules $\{R^Q\} = \{R_1^Q, R_2^Q, \ldots, R_m^Q\}$ valid for the focal subset $\mathcal{D}^Q$ (defined by $\mathcal{A}^{range}$). such that for every rule $R_k^Q = X_k \Rightarrow Y_k$, $X_k \cup Y_k \subseteq \mathcal{A}^{item}$, the local support $Supp_k^Q \geq$ minsupp and the local confidence $Conf_k^Q \geq$ minconf.*

3. THE COLARM DESIGN

To process a localized rule mining query Q, our approach adapts a *preprocess-once-query-many* (POQM) paradigm-based solution composed of two phases, namely, *offline preprocessing* and *online query processing* as described below.

3.1 The COLARM Framework

We now give an overview of our proposed approach that we call the COLARM framework (Figure 2). The framework consists of an *offline preprocessing phase* and an *online query processing phase*. The *offline preprocessing phase* computes and stores the itemset information using efficient index structures called the MIP-index (explained in Section 3.3). The index statistics are also pre-computed and stored for analysis of online mining strategies. In the *online query processing phase*, a user submits a mining request that consists of *minsupp*, *minconf* and focal subset $\mathcal{D}^Q$.

In this work, we propose a suite of six alternate mining plans for executing the localized mining request submitted by the user. The cost for each mining plan is derived and discussed in detail in Section 3.4. Based on the index statistics provided by the *offline*

[3]Discretization of quantitative data (e.g., discretization of attribute Age as {20-30, 30-40, ...} versus {20-40,40-60,...}) is an orthogonal problem and existing works [10, 20] can be applied offline to achieve the best discretization for a dataset.

[4]For a frequent itemset, all its subsets are also frequent and thus for infrequent itemset, all its supersets must also be infrequent.

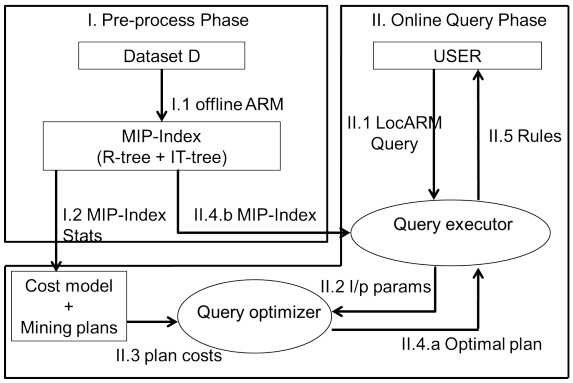


Figure 2: The COLARM framework.

preprocessing phase and the online query parameters, namely, *minsupp*, *minconf* and focal subset $\mathcal{D}^Q$, our proposed COLARM query optimizer estimates the costs of the alternate plans. The estimates are a constant time computation of six formulae, each corresponding to one mining plan. For a given mining request, the COLARM optimizer suggests the plan with the lowest estimated cost which is then used by the executor to process the online mining request.

3.2 Offline Preprocessing Phase

The preprocessing phase is a one-time offline step. For answering localized mining queries using POQM, we extend ideas from two works. First, as summarized by Boulicaut [7], existing works prestore frequent itemsets in a compact hierarchical itemset-based index such as a closed IT-tree [24]. At query-time, the index can be utilized for rule generation and to verify if the generated rules qualify the minsupp and *minconf* thresholds. Next, Das et al. [8] answer windowed frequent itemset queries by prestoring the bounding boxes of the window attributes. We thus propose to utilize a multidimensional index to store bounding boxes of itemsets. But in our relational model as *range* and *item* are specified at query-time, the multidimensional index must be more flexible than required by their transactional model. In contrast to these two works, we find that our target problem requires both features of an itemset to be prestored. Thus, for each itemset we prestore (a.) the bounding box of the itemset within the multidimensional space and (b.) the items composing the itemset in a compact hierarchical index. While feature (a.) enables us to search overlaps between the focal subset and the pre-stored itemsets, (b.) helps in efficient online rule generation and threshold verification.

Similar to prior online rule mining works [2, 8], we prestore all itemsets that satisfy a domain-specific *primary support threshold* by employing a rule mining algorithm (Charm [24]) to collect all the closed frequent itemsets at an offline step. However, the support and confidence of the rules composed from the prestored itemsets are determined at query-time based on the focal subset. Extending our knowledge from Section 2.1, in a multidimensional space of n (=3 in Figure 1) attributes, we construct a hierarchy of itemsets upto 2-itemsets, starting with the *n-itemset* cells at the finest granularity. This hierarchical collection of itemset partitions the multidimensional space into bounding boxes denoted by $\{\mathcal{D}^P\}$. We call these partitions *Multidimensional Itemset Partitions* (MIPs). Each MIP, denoted by I_k^P, represents both the bounding box $\mathcal{D}_k^P$ of the itemset in a multidimensional space as well as the actual items (attribute-value pairs) composing the itemset I. In other words, the symbols $\mathcal{D}_k^P$ and I_k^P are henceforth used interchangeably in the rest of this document to denote a MIP for an itemset I. As shown in Figure 1, each *i-itemset* MIP at level i is composed of one or more (i+1)-itemset MIPs. This MIP-index enables *downward closure property* to be implicitly applied during online mining such that if an MIP I at level i does not qualify the *minsupp*, all (i+1)-itemset MIPs that contain I can also be eliminated.

3.3 The Two Level MIP-index

We adopt a two-layered structure to store the two features of the MIPs, namely, (a.) the bounding boxes and (b.) the items composing the itemsets, as follows.

The multidimensional index for MIPs. The bounding boxes of MIPs can be indexed using any multidimensional index. Here, we use an R-tree, as it is proven to be efficient for searches over multi-dimensional boxes [11]. An R-tree index can be used to perform a *range* search to retrieve the MIPs that overlap with the focal subset $\mathcal{D}^Q$. An R-tree supports partitions of different granularities as well as overlaps and containments among partitions. An example *R-tree* is shown in Figure 3.(a).

The closed IT-tree for itemsets. We employ the *itemset-tidset search tree* [24], or in short the *IT-tree*[5]. The *IT-tree* is compact as it stores only the *closed* frequent itemsets. As shown by Zaki et al. [24], there are significant gains both in storage and computation time by utilizing the *closed IT-tree* for rule generation. An example *closed IT-tree* is shown in Figure 3.(b).

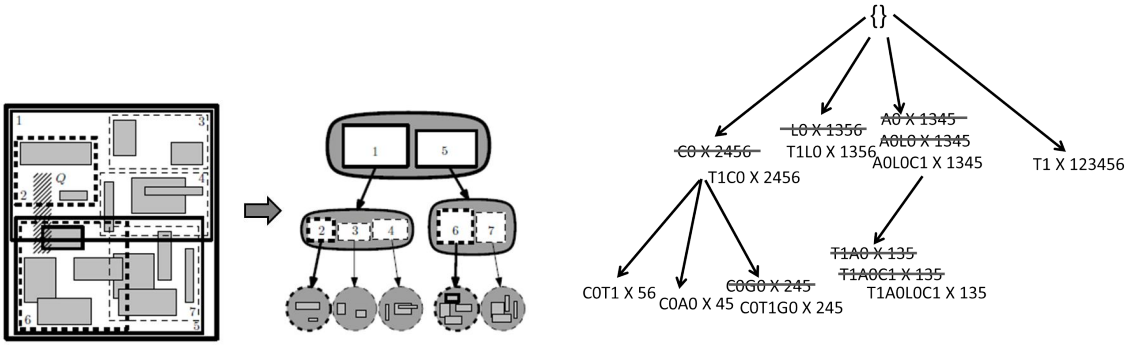
Offline MIP-index construction. Construction of MIP-index at the offline step consists of first generating all closed frequent itemsets (CFIs) using the CHARM algorithm [24] using the primary support threshold. The generated CFIs are stored in an IT-tree. The details of time and space complexity of generating CFIs and constructing IT-tree can be found in [24]. Further, we construct the R-tree using bounding boxes of the closed frequent itemsets. As this is a one time offline construction, we employ the R-tree packing scheme proposed by [11]. They proposed a method to build a packed R-tree that achieves (almost) 100% space utilization. A detailed time and space analysis for R-tree construction can be found in [11]. In this work we focus on online query processing costs that are more important than costs of one-time offline MIP-index construction costs for the problem of online localized rule mining.

3.4 Online Query Processing

In the *online query processing phase*, the user submits a localized mining query Q with four parameters $\mathcal{A}^{range}$, $\mathcal{A}^{item}$, *minsupp* and *minconf*. $\mathcal{A}^{range}$ defines the focal subset $\mathcal{D}^Q$. The candidate frequent itemsets for $\mathcal{D}^Q$ are identified by performing a range search using the focal subset $\mathcal{D}^Q$ over the MIPs $\{\mathcal{D}^P\}$ pre-stored in a R-tree. The *candidate itemsets*, denoted by $\{I_S^Q\}$, only contain the itemsets within the *item attributes* $\mathcal{A}^{item}$. Next, for each *candidate itemset* I ϵ $\{I_S^Q\}$, rules are generated using the IT-tree and the *minsupp* and *minconf* thresholds with respect to the focal subset $\mathcal{D}^Q$ are verified. Here, we make a simplifying assumption that the users are allowed to specify $\mathcal{A}^{range}$ with the prestored n-itemset cells at the lowest granularity to avoid sub-cell computations. For example, if age attribute is specified in the MIP-index with increments of 10 as {20-30, 30-40, ...}, the user selection must align with them and ranges such as Age=25-30 or Age=35-40 cannot be specified. This is a valid assumption as optimal discretization decisions using key ideas from [10, 19, 20] can be made for a given dataset during preprocessing based on the domain and the data characteristics.

Containment vs. partial overlap. Based on the expanse of the focal subset $\mathcal{D}^Q$, the precomputed MIPs can be categorized into three mutually exclusive groups, namely, *contained within* $\mathcal{D}^Q$ ($\{\mathcal{D}^P\}_c$), *partially overlapping with* $\mathcal{D}^Q$ ($\{\mathcal{D}^P\}_p$) and *disjoint with*

[5]Detailed description of the IT-tree can be obtained from [24].



(a) R-tree (b) Closed IT-tree

Figure 3: The two-level MIP-index.

$\mathcal{D}^Q$ ($\{\mathcal{D}^P\}_d$). Figure 4 illustrates five MIPs (marked with different patterns) of a dataset $\mathcal{D}$, $\{\mathcal{D}_1^P, \ldots, \mathcal{D}_5^P\}$. For an example *focal subset* $\mathcal{D}_1^Q$, the MIPs $\mathcal{D}_1^P$ and $\mathcal{D}_2^P$ are fully *contained* in it, i.e., $\{\mathcal{D}^P\}_c$ = $\{\mathcal{D}_1^P, \mathcal{D}_2^P\}$. There are no *partially overlapped* MIPs, i.e., $\{\mathcal{D}^P\}_p$ = $\emptyset$. The rest are *disjoint* MIPs, i.e., $\{\mathcal{D}^P\}_d$ = $\{\mathcal{D}_3^P, \mathcal{D}_4^P, \mathcal{D}_5^P\}$. Example focal subset $\mathcal{D}_1^Q$ is straightforward to process as only *contained* MIPs exist eliminating the need for record-level processing.

The focal subset $\mathcal{D}_2^Q$ (Figure 4) represents a distinct scenario with *partially overlapped* MIPs $\{\mathcal{D}^P\}_p$ = $\{\mathcal{D}_3^P, \mathcal{D}_4^P, \mathcal{D}_5^P\}$ and *disjoint* MIPs $\{\mathcal{D}^P\}_d$ = $\{\mathcal{D}_1^P, \mathcal{D}_2^P\}$. There are no *contained* MIPs, i.e., $\{\mathcal{D}^P\}_c$ = $\emptyset$. In such cases, for each *partially overlapped* MIP $\mathcal{D}_k^P$ (ϵ $\{\mathcal{D}^P\}_p$), the records lying at the intersection of $\mathcal{D}^Q$ and $\mathcal{D}_k^P$ must be collected using a costly database scan. The collected records must then be used to verify the thresholds for the *candidate itemset* $\{I_k^P\}$. Hence, processing partially overlapped MIPs is much costlier than processing fully contained MIPs. In a query scenario, different $\mathcal{D}^Q$ may vary in their sizes and locations within the multidimensional space. Overall, a variety of query scenarios are possible ranging from all *contained* MIPs, to mix of *contained* and *partially overlapped* MIPs, to all *partially overlapped* MIPs. A single solution may not be suitable to process all query scenarios in the most efficient manner. Therefore, we develop a suite of alternate *mining plans* and an online optimizer for selection of the most efficient plan to execute online localized mining requests.

4. STRATEGIES FOR ONLINE MINING

As opposed to treating the rule mining process as a black box, we now isolate each step in the process as an operator. Each operator has precise inputs, outputs and functionality. The goal is to find scope for optimizing each isolated operator without affecting the other operators. We first present the overall plan for processing

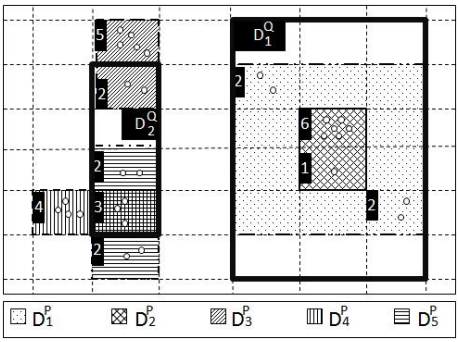
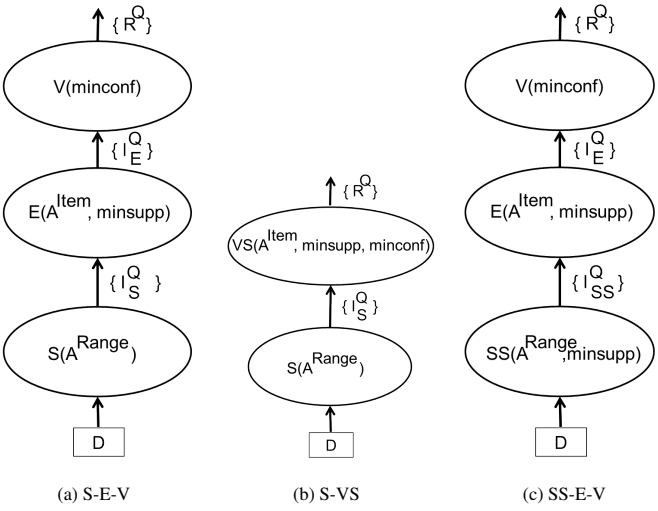


Figure 4: Itemsets covering cells.



(a) S-E-V (b) S-VS (c) SS-E-V

Figure 5: The POQM mining plans.

Symbol	Definition
C^I	Number of singleton items in an Itemset I.
TR_{const}	Constant cost of reading records from an R-tree node.
h	Height of the R-tree.
N_j	Number of nodes at level j of the R-tree.
$\mathcal{D}^Q_{i_{avg}}$	Avg. extent of the i^{th} range attribute in the focal subset.
$\mathcal{D}^P_{j,i_{avg}}$	Avg. extent of the MIPs in R-tree at level j and attribute i.
{R}	Candidate set of rules for confidence check.

Table 3: Notation used in cost estimates.

online localized rule mining queries using the MIP-index, followed by our proposed optimizations. The plan consists of a pipeline of three basic operators as shown in Figure 5.(a).

1. **SEARCH**: Given the *range* attributes $\mathcal{A}^{range}$ in the WHERE clause of Query Q, the R-tree is searched to output overlapping *candidate itemsets* denoted by $\{I_S^Q\}$. The *SEARCH* operator is defined as $\mathbf{S}[\mathcal{A}^{range}, R-tree] \longrightarrow \{I_S^Q\}$. Details of the R-tree search can be found in [21].

2. **ELIMINATE**: Given the list of *candidate itemsets* $\{I_S^Q\}$ output by the *SEARCH* operator and the *item* attributes $\mathcal{A}^{item}$, the support of the candidate itemsets must satisfy the *minsupp* and the itemsets must be composed of only the *item* attributes $\mathcal{A}^{item}$. Thus, a reduced list of candidate itemsets, denoted as $\{I_E^Q\}$, is produced. The *ELIMINATE* operator is defined as $\mathbf{E}[\{I_S^Q\}, \mathcal{A}^{item}, \textbf{\textit{minsupp}}] \longrightarrow \{I_E^Q\}$.

3. **VERIFY**: Given the reduced list of candidate itemsets $\{I_E^Q\}$, the *closed IT-tree* [24] and the records in $\mathcal{D}^Q$ are used to first *generate* rules and then *verify* whether the confidence values of the rules satisfy the *minconf* threshold. The rules $\{R^Q\}$ are returned to the user. The *VERIFY* operator is defined as $\mathbf{V}[\{I_E^Q\}, \textbf{\textit{minconf}}] \longrightarrow \{R^Q\}$. Refer to [24] for details on the traditional algorithms for rule generation and minconf verification using the *IT-tree*.

Below, we apply several novel optimization principles in the context of online localized rule mining. Each resulting optimized plan comes, not only with the benefits achieved, but also with some overhead costs. We design a model for estimating the costs of each plan. For each online request, a single most efficient plan is used for execution. We conclude this section with a summary of the six proposed alternative mining plans in Table 4.

4.1 The Basic S-E-V Plan

The basic mining plan can be composed by pipelining the three operators, namely, *SEARCH*, *ELIMINATE* and *VERIFY*. The resultant *S-E-V plan* is depicted in Figure 5.(a). The *SEARCH* operator, performing an R-tree search to output the overlapping MIPs $\{I_S^Q\}$, works as an inexpensive *coarse* granularity filter. For each candidate itemset I ϵ $\{I_S^Q\}$, the *ELIMINATE* operator filters items using $\mathcal{A}^{item}$ and verifies if local support Supp_I^Q exceeds the user-defined *minsupp*. The qualified candidate itemsets $\{I_E^Q\}$ (where $\{I_E^Q\} \subseteq \{I_S^Q\}$) are used in the *VERIFY* operator to generate rules. For each rule r, only if its local confidence (Conf_r^Q) satisfies the user-defined *minconf* will r get included in the final output $\{R^Q\}$. The detailed algorithm for threshold verification and rule generation using the *IT-tree* can be found in [24]. Both the *ELIMINATE* and the *VERIFY* operators require *finer* granularity checks at the *record-level*, i.e., the records that lie at the intersection of I and $\mathcal{D}^Q$. The candidate itemsets $\{I_S^Q\}$ and $\{I_E^Q\}$ produced by *SEARCH* and *ELIMINATE* respectively may contain *false positives* but are guaranteed to not generate any *false negatives* as even the partially overlapping itemsets are pushed up as candidates, but itemsets that must form the answers are never eliminated.

Execution costs and analysis. The cost of the *SEARCH* operator (i.e., an R-tree lookup), depends on the expected number of disk accesses [21]. Lemma 4.1[6] gives an estimated count of the candidate itemsets ($\{I_S^Q\}$) output by *SEARCH*.

LEMMA 4.1. *An R-tree storing a set of N MIPs $\{\mathcal{D}_1^P, \ldots, \mathcal{D}_N^P\}$ with average extent for the i^{th} attribute as $\mathcal{D}_{i_{avg}}^P$ and the focal subset $\mathcal{D}^Q$ with average extent of the i^{th} attribute as $\mathcal{D}_{i_{avg}}^Q$, for n attributes; the average number of candidate itemsets $\{I_S^Q\}$ intersected by $\mathcal{D}^Q$ is approximately computed as:*

$$|\{I_S^Q\}| = N \times \prod_{i=1}^{n} (\mathcal{D}_{i_{avg}}^P + \mathcal{D}_{i_{avg}}^Q)$$

While the cost for *ELIMINATE* is given in Equation 1, Lemma 4.2[7] gives the estimated count of candidate itemsets ($\{I_E^Q\}$) output by *ELIMINATE*.

[6]Derived from the R-tree search algorithm [21].

[7]Derived from the IT-tree minsupp check algorithm [24].

LEMMA 4.2. *Given a set of candidate itemsets $\{I_S^Q\}$ with itemset I having local support Supp_I^Q and the focal subset $\mathcal{D}^Q$ with threshold requirement of minsupp, the average number of candidate itemsets $\{I_E^Q\}$ output by ELIMINATE is:*

$$|\{I_E^Q\}| = \textstyle\sum_{i \epsilon \{I_S^Q\}} (Supp_i^Q + minsupp)$$

In the *VERIFY* operator, the output ruleset $\{R^Q\}$ can be generated using the IT-tree. For an itemset I, one needs to traverse the IT-tree up to the level of I in the IT-tree (Lemma 4.3[8]). The overall cost of the *S-E-V plan* as an addition of the individual costs is shown in Equation 1.

LEMMA 4.3. *The level of the IT-tree at which an itemset I exists equals the number of singleton items composing I denoted by C^I.*

$$\begin{aligned}
&COST(S-E-V) = COST(S) + COST(E) + COST(V), where,\\
&COST(S) = [\sum_{j=1}^{h-1} \{N_j \times \prod_{k=1}^{n} (\mathcal{D}_{j,i_{avg}}^P + \mathcal{D}_{k_{avg}}^Q)\}],\\
&COST(E) = [|\{I_S^Q\}| \times |D^Q|],\\
&COST(V) = [\sum_{i \epsilon \{I_E^Q\}} (C^i \times |D^Q|) + \sum_{r \epsilon \{R\}} (Conf_r^Q + minconf)].
\end{aligned} \tag{1}$$

The SEARCH is inexpensive as it is a coarse granularity check. However, if a larger range is chosen by the user it may require multiple disk accesses and may impact the execution costs. In summary, the S-E-V plan would perform well for small $\mathcal{D}^Q$ expanses when most of the MIPs are filtered out in the R-tree search. The VERIFY operator requires record-level operations and multiple iterations over the IT-tree. Thus it tends to be the bottleneck of this plan. The S-E-V plan is effective if the ELIMINATE achieves high reduction of candidate itemsets from $\{I_S^Q\}$ to $\{I_E^Q\}$, with possibly low overhead. The smaller the set $\{I_E^Q\}$ reaching the costly VERIFY operator, the better the plan's performance. However, a small overhead for performing the record-level ELIMINATE operator arises.

4.2 Pushing Selection Up The Plan

The reduction of the candidate itemsets by ELIMINATE is factor affecting the effectiveness of the S-E-V plan. However, if the expensive record-level ELIMINATE filters no or very few candidate itemsets, i.e., $\{I_E^Q\} \simeq \{I_S^Q\}$, then ELIMINATE no longer remains a beneficial filter. We now introduce the *S-VS strategy* that rewrites the S-E-V plan by merging the ELIMINATE and the VERIFY operations into a single operator called the *SUPPORTED-VERIFY* operator (defined below). The *S-VS* plan is depicted in Figure 5.(b).

SUPPORTED-VERIFY operator. This operator is represented as $VS[\{I_S^Q\}, \mathcal{A}^{item}, minsupp, minconf] \longrightarrow \{R^Q\}$. It takes as input the candidate itemsets from the SEARCH ($\{I_S^Q\}$), the item attributes $\mathcal{A}^{item}$ and the thresholds *minsupp* and *minconf* from the query. It first filters itemsets not in $\mathcal{A}^{item}$. With the remaining qualified itemsets, a set of rules $\{R\}$ is generated but only the rules satisfying both *minsupp* and *minconf* are output as $\{R^Q\}$.

Execution costs and analysis. SEARCH cost for S-VS remains the same as S-E-V. The *minsupp*-based elimination of itemsets is interleaved with the *minconf* verification using the IT-tree. For

[8]Derived from the IT-tree rule generation algorithm [24].

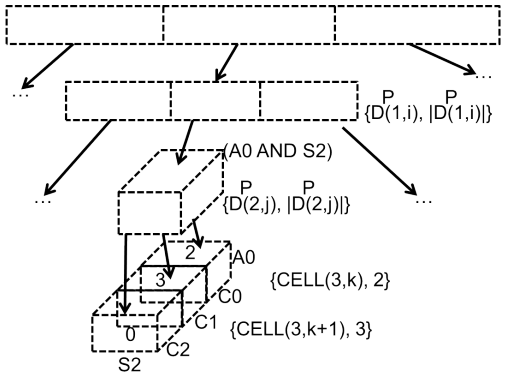
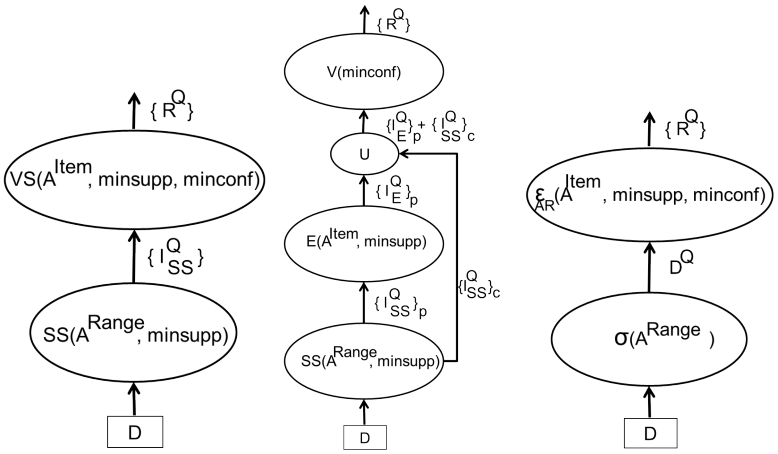


Figure 6: Supported R-tree.



(a) SS-VS (b) SS-E-U-V (c) ARM

Figure 7: More mining plans.

cases $\{I_E^Q\} \simeq \{I_S^Q\}$, S-VS is bound to outperform S-E-V. This plan is depicted in Fig. 5.(b) and Eq. 2 shows the overall costs.

$$COST(S-VS) = COST(S)+COST(VS), where,$$
$$COST(VS) = [\sum_{i\epsilon\{I_S^Q\}} (C^i \times |\mathcal{D}^Q|)+ \sum_{r\epsilon\{R\}} (Conf_r^Q + minconf)]. \quad (2)$$

4.3 The Supported R-tree Filter

In the above two plans, SEARCH selects all MIPs overlapping with with $\mathcal{D}^Q$ for the expensive record-level support check irrespective of the extent of overlap, i.e., MIPs overlapping by as few as a single record are also chosen. In cases where $\mathcal{D}^Q$ partially overlaps with a large number of MIPs, an excessive number of record-level support checks are needed. With the goal of further reducing $\{I_S^Q\}$, we now design the *SS-E-V* plan (depicted in Fig. 5.(c)). The SEARCH operator is modified using the insights from Lemma 4.4.

> LEMMA 4.4. *Given a focal subset $\mathcal{D}^Q$ ($\subseteq \mathcal{D}$) and any itemset I whose bounding box overlaps $\mathcal{D}^Q$, the number of records in $\mathcal{D}^Q$ that contain I, denoted by $|\mathcal{D}_I^Q|$, has an upper bound $|\mathcal{D}_I^G|$, where $|\mathcal{D}_I^G|$ denotes the total number of records supporting I in the full dataset $\mathcal{D}$. In short, $|\mathcal{D}_I^Q| \leq |\mathcal{D}_I^G|$.*

Let $|\mathcal{D}_I^G|$ (e.g., 2000) denote the global count of prestored itemset I with respect to the entire dataset $\mathcal{D}$ (e.g., 100000). Given a focal subset $\mathcal{D}^Q$ (e.g., 10000) and *minsupp* = 25%, we define $\frac{|\mathcal{D}_I^G|}{|\mathcal{D}^Q|}$ (here, 2000/10000 = 20%) as the upper bound of the local support of I, i.e., $\text{Supp}_I^Q \leq \frac{|\mathcal{D}_I^G|}{|\mathcal{D}^Q|}$ (using Lemma 4.4). $\text{Supp}_I^Q = \frac{|\mathcal{D}_I^G|}{|\mathcal{D}^Q|}$ only if all records of $\mathcal{D}$ that contain I are included in $\mathcal{D}^Q$. By prestoring global support counts ($|\mathcal{D}_I^G|$) of an itemset I with its MIP bounding box inside the R-tree, we can eliminate I if *minsupp* $> \frac{|\mathcal{D}_I^G|}{|\mathcal{D}^Q|}$. This customized R-tree structure is henceforth called the **Supported R-tree** (Figure 6). This *supported R-tree filter* may significantly reduce the list of candidate itemsets without having to perform a record-level support checks. The modified *SEARCH* operator exploiting the supported R-tree is defined below.

SUPPORTED-SEARCH operator. The SS$[\mathcal{A}^{range}$, *minsupp*$] \longrightarrow \{I_{SS}^Q\}$ operator takes in the range attributes $\mathcal{A}^{range}$ and the *minsupp* threshold. It outputs the candidate itemsets $\{I_{SS}^Q\}$ that overlap with the $\mathcal{A}^{range}$ only if their global support counts exceed the *minsupp*. Thus the SS operator works as a *coarse granularity supported R-tree filter* together with performing the range search.

Execution costs and analysis. The costs incurred by the SS-E-V are shown in Equation 3. Unlike the regular SEARCH operator, the selectivity of the SS operator also takes the support of the stored MIPs and the *minsupp* into account. This coarse granularity *minsupp* check can be cheaply interleaved with the range search. The costs of ELIMINATE and VERIFY are identical to Equation 1. The key contribution of the SS-E-V plan is that the SUPPORTED-SEARCH potentially generates fewer candidate itemsets ($\{I_{SS}^Q\}$) than produced by the original SEARCH operator, i.e., $\{I_{SS}^Q\} \ll \{I_S^Q\}$. Thus, potentially reducing the inputs to ELIMINATE and VERIFY operators.

$$COST(SS-E-V) = COST(SS)+COST(E)+COST(V), where,$$
$$COST(SS) = [\sum_{j=1}^{h-1}\{N_j \times \prod_{k=1}^{n}(\mathcal{D}_{j,k_{avg}}^P+\mathcal{D}_{k_{avg}}^Q)\times$$
$$(Supp_j + minsupp)\}]. \quad (3)$$

4.4 Pipelining both Supported Operators

The SS-E-V plan can run into the same issues with the ELIMINATE as the S-E-V plan, i.e., if $\{I_E^Q\} \simeq \{I_{SS}^Q\}$, then ELIMINATE may not be an effective filter. We may instead opt to pipeline SUPPORTED-SEARCH and SUPPORTED-VERIFY operators resulting in the *SS-VS* plan (Figure 7.(a)).

Execution costs and analysis. The cost incurred by the SS-VS (Equation 4) is the sum of the costs of SS and VS operators. The VS operator processes candidate itemsets $\{I_{SS}^Q\}$ output by SS. SS-VS is guaranteed to outperform SS-E-V when $\{I_E^Q\} \simeq \{I_{SS}^Q\}$.

$$COST(SS-VS) = COST(SS)+COST(VS). \quad (4)$$

4.5 Treating Contained and Overlapped MIPs Differently

In the plans described thus far, irrespective of whether a MIP is fully contained within $\mathcal{D}^Q$ or it only overlaps by as few as a single record, it is sent for the record-level threshold check. We now introduce a key property in Lemma 4.5 that opens a new opportunity for optimization.

Mining Plan	Optimization	Query Cost
S-E-V	Basic SEARCH+ELIMINATE+VERIFY plan	COST(S) + COST(E) + COST(V)
S-VS	Selection push-up	COST(S) + COST(VS)
SS-E-V	Supported R-tree filter	COST(SS) + COST(E) + COST(V)
SS-VS	Supported R-tree filter + selection push-up	COST(SS) + COST(VS)
SS-E-U-V	Supported R-tree filter + differential treatment of containment and overlap	COST(SS) + COST(E) + COST(U) + COST(V)
ARM	Traditional rule mining over focal subset	COST(σ) + COST(ε_{AR})

Table 4: Summary of the six mining plans.

LEMMA 4.5. *If the MIP bounding box D_I^P of an itemset I is completely **contained** within the focal subset $\mathcal{D}^Q$, the **local support** of I equals its **global support**. In other words, if ($D_I^P \subseteq \mathcal{D}^Q$), then ($Supp_I^Q = Supp_I^G$). Proof is trivial.*

By Lemma 4.5, once a fully contained Itemset I is included in the candidate list $\{I_{SS}^Q\}$ by the SS operator, any record-level check of I's support is redundant. Therefore, we now propose a mining plan that treats the fully *contained* MIPs differently from the *partially overlapped* MIPs. The MIPs contained within $\mathcal{D}^Q$, denoted by $\{I_{SS}^Q\}_c$, can be safely sent to VERIFY for rule generation and *minconf* verification. Only the MIPs that *partially overlap* with $\mathcal{D}^Q$, denoted by $\{I_{SS}^Q\}_p$, are required to undergo the expensive record-level support check. Here, subscripts c and p stand for *contained* and *partially overlapped*, respectively. ELIMINATE outputs the candidate itemsets $\{I_E^Q\}_p$ ($\subseteq \{I_{SS}^Q\}_p$). Figure 7.(b) depicts the resultant SS-E-U-V plan. Lastly, we introduce an itemset-level UNION operator to merge the two lists of itemsets to form a single list of itemsets for rule generation in VERIFY.

UNION operator. This operator, denoted by U[$\{I_{I1}^Q\}$, $\{I_{I2}^Q\}$, ...] $\longrightarrow \{I^Q\}_{p,c}$, takes as input two or more lists of itemsets, (here, $\{I_E^Q\}_p$ and $\{I_{SS}^Q\}_c$) and produces a single list which is the union of the two inputs.

Execution costs and analysis. The SS-E-U-V plan now incorporates all the optimizations designed above. It is expected to be highly effective in processing a large spectrum of query requests. The SS operator's cost remains the same as SS-VS, but SS now splits the candidate itemsets into two mutually exclusive sets, namely, the *contained* MIPs $\{I_{SS}^Q\}_c$ and *partially overlapped* $\{I_{SS}^Q\}_p$ MIPs. The partially overlapped MIPs ($\{I_{SS}^Q\}_p$) must first pass through ELIMINATE to produce a reduced list $\{I_E^Q\}_p$ ($\subseteq \{I_{SS}^Q\}_p$). As the two lists $\{I_{SS}^Q\}_c$ and $\{I_E^Q\}_p$ merged by UNION are mutually exclusive (requiring no duplicate removal), UNION incurs a constant cost. VERIFY processes a total of ($|\{I_E^Q\}_p|$ + $|\{I_{SS}^Q\}_c|$) candidate itemsets for rule generation (Equation 5).

$$
\begin{aligned}
&COST(SS-E-U-V) = COST(SS)+COST(E)+COST(U)+\\
&COST(V), where,\\
&COST(U) = [U_{const}],\\
&COST(V) = [\sum_{i\epsilon(\{I_E^Q\}_p+\{I_{SS}^Q\}_c)} (C^i \times |\mathcal{D}^Q|)+\\
&\sum_{r\epsilon\{R\}} (Conf_r^Q+minconf)].
\end{aligned}
\tag{5}
$$

4.6 Employing the Traditional Mining Plan

For the extreme case when huge numbers of partially overlapped MIPs might require expensive record-level checks, a traditional two-step rule mining algorithm may be more practical than employing the sophisticated MIP-index-based mining plans. This traditional plan may employ a mining algorithm (Charm [24]) directly over the extracted focal subset $\mathcal{D}^Q$. We refer to this baseline plan as the ARM plan (Figure 7.(c)) that is composed of the following two operators:

SELECT operator. Denoted as $\sigma[\mathcal{A}^{range}, \mathcal{D}] \longrightarrow \mathcal{D}^Q$, SELECT takes as input the range attributes $\mathcal{A}^{range}$ and the dataset $\mathcal{D}$ to output the records of the focal subset $\mathcal{D}^Q$.

ARM operator. This operator, denoted by $\varepsilon_{AR}[\mathcal{D}^Q, \mathcal{A}^{item}$, *minsupp*, *minconf* $] \longrightarrow \{R^Q\}$, performs the traditional ***A**ssociation **R**ule **M**ining* (ARM) from scratch. It takes the focal subset $\mathcal{D}^Q$, the item attributes $\mathcal{A}^{item}$ and the thresholds *minsupp* and *minconf* as inputs. It outputs the desired ruleset $\{R^Q\}$.

Execution costs and analysis. SELECTION using an R-tree requires extracting the records from the overlapped bounding boxes. The costs for the rule mining step depend on the input data size, the length of the maximum frequent itemset and the number of dimensions. The costs of this baseline plan is computed by adding the SELECTION and ARM costs (Equation 6).

$$
\begin{aligned}
&COST(ARM-Plan) = COST(\sigma)+COST(\varepsilon_{AR}), where,\\
&COST(\sigma) = [(\sum_{j=1}^{h-1}\{N_j \times \prod_{k=1}^{n}(\mathcal{D}_{j,k_{avg}}^P+\mathcal{D}_{k_{avg}}^Q)\}) \times TR_{const}],\\
&COST(\varepsilon_{AR}) = [|\mathcal{D}^Q| \times (\max_{I\epsilon\{I^Q\}}(C^I)) \times n].
\end{aligned}
\tag{6}
$$

5. EXPERIMENTAL VALIDATION OF THE COLARM FRAMEWORK

In this section we provide experimental results to explore the validity of our COLARM framework. We focus on three questions:

- Is the COLARM framework capable of providing correct decisions regarding the best choice among the set of six plans?
- Do the proposed optimizations in the five MIP-index based mining plans achieve the expected execution cost benefits for the tested datasets?
- What is the extent of Simpson's Paradox observed in the context of localized rule mining?

We conducted experiments on a Windows 7 machine with Intel(R) Xeon(R) CPU X3440@2.53 GHz processor and 8 GB of RAM. The mining plans and the COLARM optimizer are coded using C++.

Experimental datasets. We use three real benchmark datasets from the UC Irvine Machine Learning Repository [5], namely, chess, mushroom and PUMSB. Chess contains 3196 records with 76 distinct items. Mushroom contains 8124 records with 120 distinct items. PUMSB contains 49046 records with 7117 distinct items. Figure 8 depicts the number of closed frequent itemsets stored in the MIP-index structure for chess, mushroom and PUMSB datasets, respectively as we vary the primary threshold values[9]. For both Chess and PUMSB datasets the number of closed itemsets drastically increases with a decrease in the primary threshold. The change in Mushroom is rather gradual. A detailed analysis of the three chosen datasets including a discussion of the distribution of

[9]The primary support ranges are as suggested in [24].

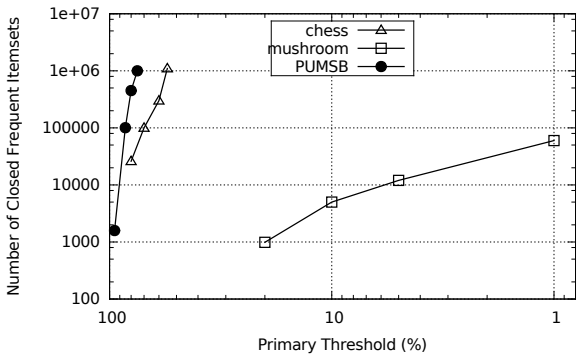
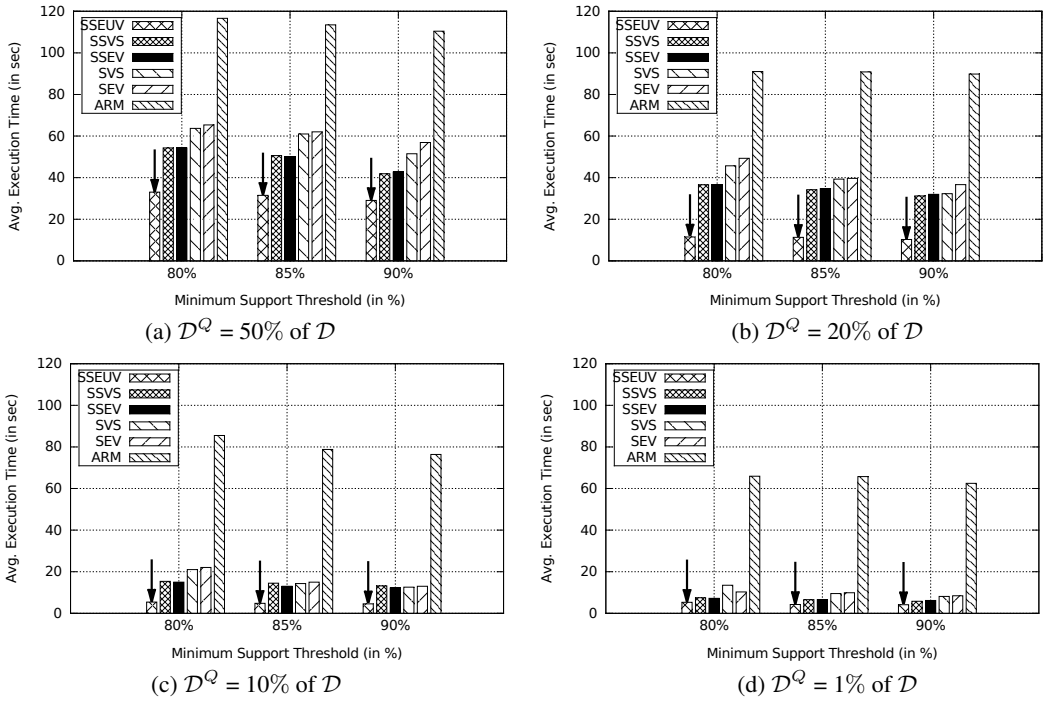


Figure 8: # Frequent itemsets by primary thresholds.



(a) $\mathcal{D}^Q$ = 50% of $\mathcal{D}$ (b) $\mathcal{D}^Q$ = 20% of $\mathcal{D}$

(c) $\mathcal{D}^Q$ = 10% of $\mathcal{D}$ (d) $\mathcal{D}^Q$ = 1% of $\mathcal{D}$

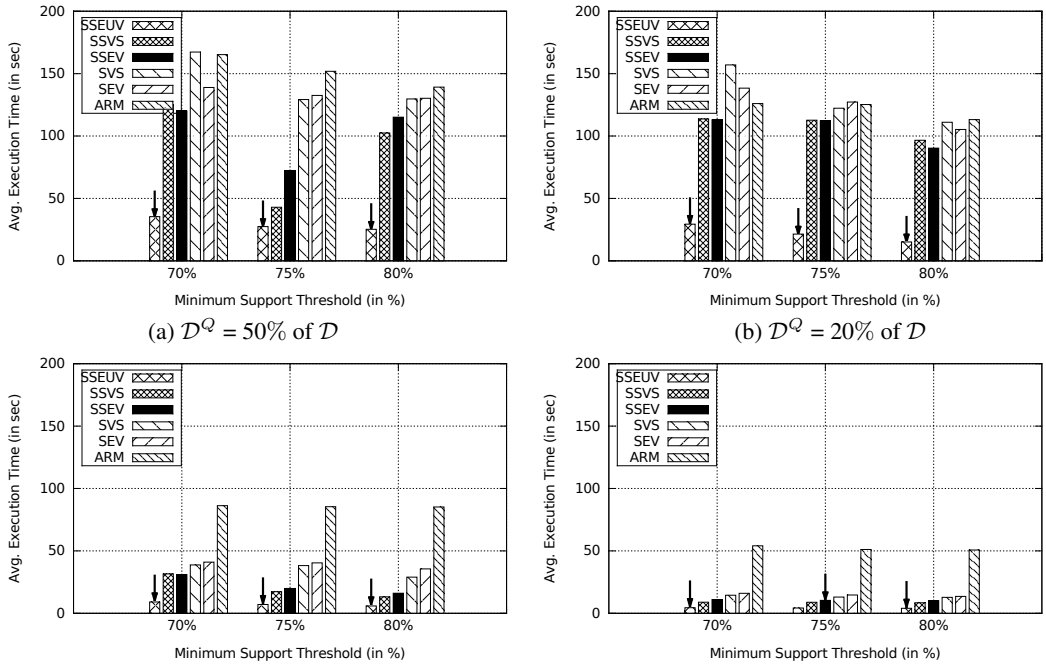
Figure 9: Avg. CPU costs of mining plans for chess dataset.

closed frequent itemsets (CFIs) by their length can be found in [24]. The length of an itemset I corresponds to the *number of singleton items in I*, also denoted by C^I (see Table 3). In other words, the length also denotes the number of attributes (dimensions) in the R-tree. Overall, chess and PUMSB have a symmetric distribution of CFIs whereas mushroom has a bi-modal distribution of closed frequent itemsets (CFIs). Thus, these three datasets represent diverse characteristics for testing the effectiveness of the COLARM framework in terms of local itemsets discovered and selecting mining plans with quick response times.

Preparing the MIP-index is a one-time offline step irrespective of the number of online requests processed thereafter. Thus, the index construction costs are of less importance. Infact, prior works [13, 24] have studied, in particular, the impact of the number of prestored itemsets on the subsequent online execution. For the online local mining experiments, we focus on a different set of measurements as described below. For our tested scenarios of online mining queries, we use the MIP-index structures created using primary support 60% for chess, 5% for mushroom and 80% for PUMSB datasets, respectively. The corresponding MIP-index structure stores approximately 300000, 10000 and 450000 closed frequent itemsets for chess, mushroom and PUMSB, respectively.

Experimental methodology. Our experiments test two metrics, namely, (a.) the query response time benefits of using the cost-based optimizer for plan selection, and (b.) the extent of Simpson's paradox in localized mining. The *range* and *item* attributes are part of the user request (see Q in Sec. 2.2). This query-time selection from a common pool of attributes means that existing solutions [8] are not applicable and thus can not be compared against. We study the following measures:

1. **Execution time metric**: We compare the average execution times of the six alternate mining plans for different query parameter settings i.e., by changing the *focal subset size*, *minsupp* and *minconf*, respectively. We compare plan selection by the COLARM optimizer based on the estimated costs with the actual execution costs of the plans.



(a) $\mathcal{D}^Q$ = 50% of $\mathcal{D}$ (b) $\mathcal{D}^Q$ = 20% of $\mathcal{D}$

(c) $\mathcal{D}^Q$ = 10% of $\mathcal{D}$ (d) $\mathcal{D}^Q$ = 1% of $\mathcal{D}$

Figure 10: Avg. CPU costs of mining plans for mushroom dataset.

2. **Optimization benefits**: In addition to the above factors derived from the cost model, we also evaluate the effectiveness of our optimized MIP-index-based mining plans measured by their respective gains in reducing the execution costs compared with the basic S-E-V plan.

3. **Local rules vs. global rules**: For the tested local mining sccenarios, we compare the counts of fresh local rules versus the global rules to determine the extent of Simpson's paradox observed.

5.1 Accuracy of the COLARM Optimizer

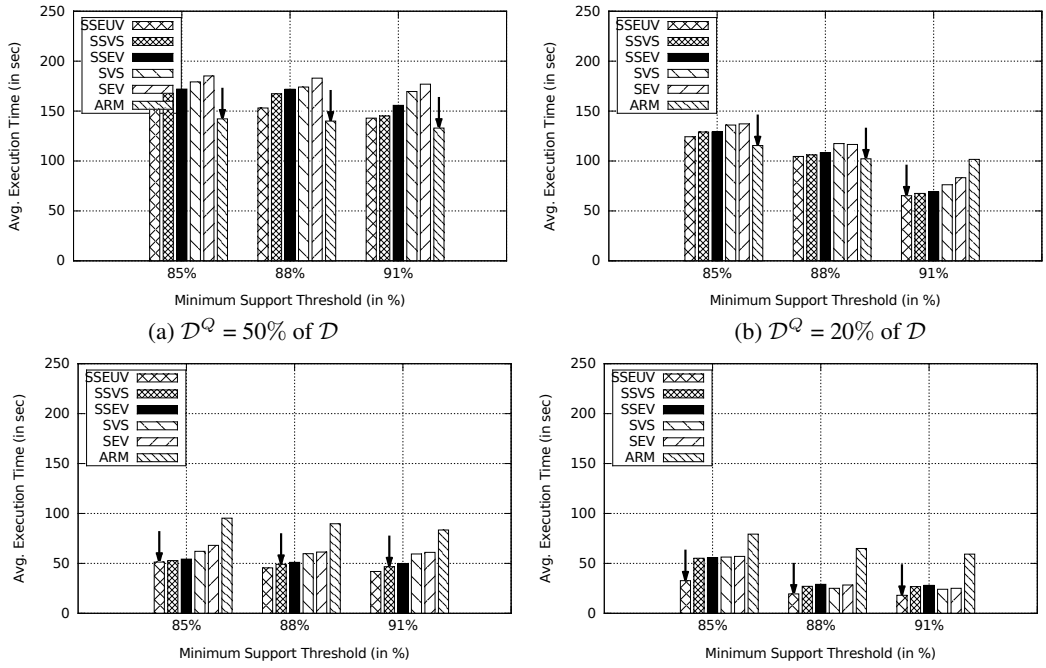
Here we present our experiments to validate the COLARM optimizer's capability to identify the most cost effective plan for a diversity of tested mining requests. We vary the mining parameters such as $\mathcal{D}^Q$ *size* (4 distinct values), *minsupp* (3 distinct values) and *minconf* (3 distinct values) to create a total of 36 distinct localized mining requests for each of the three datasets. For each tested mining request, the COLARM optimizer computes the estimated costs of the six plans using the cost formulae derived in Section 4 and suggests the plan with the minimum estimated cost. To demonstrate the effectiveness of our COLARM optimizer, we plot the average execution times of the six plans for the tested scenarios and indicate with an arrow the plan chosen by COLARM in majority of the cases.

In our experiments we vary the $\mathcal{D}^Q$ sizes as % of the complete dataset $\mathcal{D}$, namely, 50%, 20%, 10% and 1%. Similarly, different *minsupp* (as % of subset $\mathcal{D}^Q$) values are used, namely, {80, 85, 90} for the chess dataset, {70, 75, 80} for the mushroom dataset and {85, 88, 91} for the PUMSB dataset, respectively. Further, we used high minconf values, namely, {85, 90, 95}, for all datasets. These chosen *minsupp* and *minconf* values are based on the primary support values chosen for creating the MIP-index and the analysis of the distribution of their closed frequent itemsets by their lengths provided in [24].

Impact of varying focal subset sizes. Figures 9, 10 and 11 depict the average execution times of the six mining plans for chess, mushroom and PUMSB, respectively. For each $|\mathcal{D}^Q|$ (say, 50% of $|\mathcal{D}|$), the execution time is averaged over several runs by submitting queries with fixed sized $\mathcal{D}^Q$ over different regions of the dataset. The plan chosen by the COLARM optimizer on majority of runs based on the estimated costs is marked with an arrow. We fix the *minconf* to 85%. For each dataset, $|\mathcal{D}^Q|$ equals (a) 50%, (b) 20%, (c) 10% and (d) 1% of $|\mathcal{D}|$ of the tuples. For all datasets (Figures 9, 10 and 11), with a decrease in the focal subset size from 50% to 1% (charts (a) to (d)), the execution costs of the plans decrease drastically. This conforms to predicted trends of the cost model. The smaller the focal subset $|\mathcal{D}^Q|$, the fewer the overlapping MIPs to be verified. This leads to a faster response times.

Impact of varying minimum support. For the chess dataset (Figures 9 (a)-(d)), our proposed mining plans consistently outperform the traditional *ARM* plan. As expected, the SS-E-U-V plan is optimized to deliver the best execution time. This trend is attributed to the sparse population of itemsets in chess and a symmetric distribution of itemsets by length. In Figures 10 (a)-(d) a similar trend is observed for the mushroom dataset. We also observe that among the MIP-index based plans the average execution cost decreases from S-E-V to SS-E-U-V in most cases as each of the optimizations are employed. In Figures 11 (a)-(d) for the PUMSB dataset, the MIP-index-based mining plans continue to perform distinctly better at the lower $|\mathcal{D}^Q|$ sizes. However, for the higher $|\mathcal{D}^Q|$ (50% and 20%), we observed no clear winner. In some cases, the baseline ARM plan outperforms the other plans. This distinct trend in PUMSB dataset is attributed to due the symmetric distribution of itemsets by length and high density of the dataset. The *impact of varying* ***minconf*** is similar to, yet less drastic than, that of varying *minsupp*. Thus, those experimental results are omitted due to space constraints.

Plan selection accuracy of COLARM optimizer. In Figures 9, 10 and 11, we depict with an *arrow* the mining plans chosen by the COLARM optimizer using our cost model. As the results are



(a) $\mathcal{D}^Q$ = 50% of $\mathcal{D}$ (b) $\mathcal{D}^Q$ = 20% of $\mathcal{D}$

(c) $\mathcal{D}^Q$ = 10% of $\mathcal{D}$ (d) $\mathcal{D}^Q$ = 1% of $\mathcal{D}$

Figure 11: Avg. CPU costs of mining plans for PUMSB dataset.

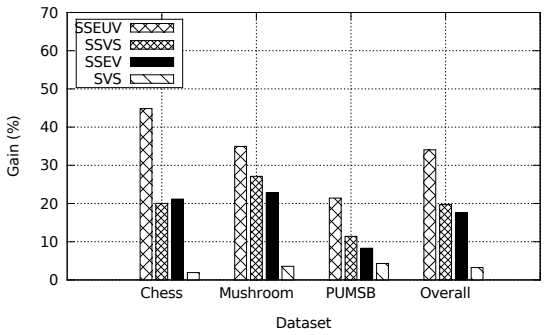


Figure 12: Gains from optimizations.

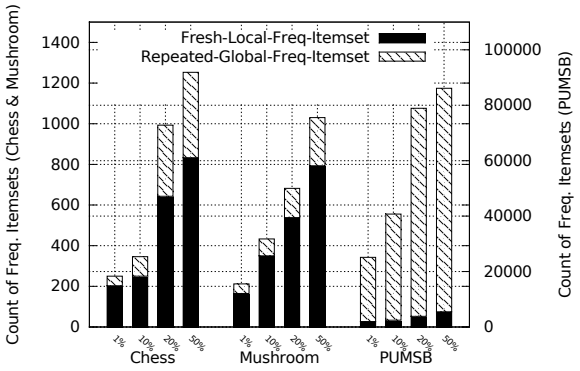


Figure 13: Average local versus global CFIs.

averages over several runs, the arrow indicates the plan that was chosen in majority. Thus, we find that COLARM indeed almost always selects the most efficient plan. Out of the total 108 distinct tested request scenarios (3 datasets and 36 parameter settings), COLARM makes erroneous decisions for only three cases, namely, in Figure 10.(d) for minsupp = 75% and in Figure 11.(c) for minsupp 88% and 91%. Thus, COLARM is more than 93% accurate. When failing to select the optimal plan, COLARM selects a plan with at most 5% extra cost compared with the optimal.

5.2 Measuring the Optimization Benefits

Here, we zoom into the optimized operators of the MIP-index based mining plans. We use the *S-E-V plan* as the baseline and compare its execution costs against our optimized plans, namely, *S-VS*, *SS-E-V*, *SS-VS* and *SS-E-U-V*. For a plan P, the gain is computed as $\frac{(CPU_{SEV} - CPU_P)}{CPU_{SEV}}$. Figure 12 depicts the % gain for each optimized plan. In Figure 12, the benefits of pushing selection up, i.e., the VS operator, are minor. On the other hand, the plans using the SS operator show 8% to 44% gains, indicating the success of the *supported R-tree filter*. Particularly, SS-E-U-V exhibits high gains ($\sim$22% to 44%).

5.3 Comparing Local vs. Global Rules

In Figure 13 we summarize the average number of closed frequent itemsets (CFIs) mined in our tested cases. Here it is important to understand that these local frequent itemsets were captured in the offline MIP-index construction due to the use of low primary support, namely, 60% for chess, 5% for mushroom and 80% for PUMSB datasets, respectively. These local itemsets qualify at high support ($\geq$80% for chess, $\geq$69% for mushroom, and $\geq$85% for PUMSB) only at the 1%-50% subset granularities. These local CFIs will be either missed when reasonable global minsupp (> primary threshold) values are input, or will be hidden within a large number of itemsets if the user inputs a global minsupp less than the primary threshold.

Compared with the global itemsets mined at reasonable minsupp values 80% for chess and 60% for mushroom, we find that majority of the CFIs mined in the tested query scenarios are lo-

cal CFIs, providing strong evidence for Simpson's paradox. For example, for the mushroom dataset, when a subset of mushroom samples with attribute-value pair *stalk-shape=tapering* is chosen, we observe 32 local CFIs emerge that have a low global support of $\simeq$39%. Two example CFIs are {*stalk-surface-below-ring=smooth, veil-color=white*} and {*stalk-surface-above-ring=smooth, veil-type =partial, ring-number=one*}. These 32 CFIs are hidden in the global context as 625 other CFIs exist in the global context with higher support values. For this subset of mushroom samples satisfying attribute-value pair *stalk-shape=tapering*, these CFIs have local support values $\geq$69%. For PUMSB the itemsets mined with global minsupp = 85% are in majority also for the local queries, yet several additional prominent local CFIs are discovered as well.

Experimental conclusions. Our main experimental findings are summarized as follows:

• The experimental evaluation establishes the *accuracy* of the cost model as the trends observed are coherent with the predictions of the cost model.

• In most of our tested cases, the *SS-E-U-V plan* consistently outperforms the other plans. In some cases for PUMSB, the ARM plan marginally outperforms the others. Overall, no single mining plan is a clear winner.

• The COLARM optimizer successfully identifies the most efficient plan with more than 93% accuracy for all our tested cases including extreme cases where the traditional ARM plan is most efficient.

• Among the MIP-index-based mining plans, the plans using the SS operator are significantly less expensive indicating the success of the supported R-tree filter.

• We observe strong evidence for Simpson's paradox in the context of local rule mining requests.

6. RELATED WORK

Aggarwal et al. [1] identify the importance of localized rules. Yet, they propose a one-time clustering-based summary of the market basket dataset with a *fixed* minsupport value. Han et al. [17] propose a 2-phased processing framework for online *constraint-based mining*. However, both these works improve existing query-time rule mining algorithms rather than precomputing itemsets for online mining of localized rules. These improved algorithms are thus orthogonal to our efforts and could in fact be used in conjunction with our proposed strategies.

Two recent works on mining rules on multidimensional data [8, 22] relate to our work. However, a closer analysis reveals that they solve a different problem altogether. First, they assume that in a *transactional* data model (e.g., market basket dataset [1]), there is a set of *window* attributes, such as location and time of transaction, on which partitions are created. Thus, in their data model, the transaction data forming the itemsets are required to be distinct from the partitioning attributes. In contrast, our work focuses on a *relational* data model, such as in [20], where the subset attributes and the items used for forming rules are from a common pool of attribute-value pairs. The offline assumptions to aid the precomputation as done in [8] are unrealistic in our model. Thus, these existing approaches [8,22] are inapplicable to the relational model.

7. CONCLUSION AND FUTURE WORK

We address the novel problem of *online localized rule mining*. We design a novel MIP-index that makes POQM feasible for our target problem. Using the efficient two-layered MIP-index, we design several alternate mining plans that employ sophisticated optimization strategies such as *selection pushup*, *supported R-tree filter* and *differential treatment of contained versus partially overlapped MIPs*. Our **COLARM** optimizer selects the most efficient plan for processing each mining request. Our extensive experiments using benchmark datasets establish the effectiveness of our COLARM framework in a rich variety of tested cases. We also find strong evidence of Simpson's paradox in the context of localized rule mining. In future, we plan to tackle two additional problems related to localized association rule mining, namely, (a.) mining the range, support and confidence parameters from the data in an automatic and efficient way; and (b.) multi-query optimization in the context of localized association rule mining.

8. REFERENCES

[1] C. C. Aggarwal, C. Procopiuc, and P. S. Yu. Finding localized associations in market basket data. *IEEE Trans. on Knowl. and Data Eng.*, pages 51–62, 2002.

[2] C. C. Aggarwal and P. S. Yu. Online generation of association rules. In *ICDE*, pages 402–411, 1998.

[3] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, pages 207–216, 1993.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, pages 487–499, 1994.

[5] A. Asuncion and D. Newman. UCI ML repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 2007.

[6] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD*, pages 255–264, 1997.

[7] T. Calders, C. Rigotti, and J.-F. Boulicaut. A survey on condensed representations for frequent sets. In *Constraint-Based Mining and Inductive Databases*, pages 64–80, 2004.

[8] M. Das, D. P, P. Deshpande, and R. Kannan. Fast rule mining over multi-dimensional windows. In *SDM*, pages 582–593, 2011.

[9] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley, 1994.

[10] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. VLDB, pages 420–431, 1995.

[11] I. Kamel and C. Faloutsos. On packing r-trees. In *ACM CIKM*, pages 490–499, 1993.

[12] M. Leleu, C. Rigotti, J.-F. Boulicaut, and G. Euvrard. Constraint-based mining of sequential patterns over datasets with consecutive repetitions. In *PKDD*, pages 303–314, 2003.

[13] X. Lin, A. Mukherji, E. A. Rundensteiner, C. Ruiz, and M. O. Ward. Paras: Parameter space framework for online association mining. In *PVLDB*, volume 6 (3), pages 193–204, 2013.

[14] K. Morik, J.-F. Boulicaut, and A. Siebes, editors. *Local Pattern Detection, International Seminar, Dagstuhl Castle, Germany, April 12-16, 2004, Revised Selected Papers*, volume 3539 of *Lecture Notes in Computer Science*. Springer, 2005.

[15] A. Mukherji, X. Lin, C. R. Botaish, J. Whitehouse, E. A. Rundensteiner, M. O. Ward, and C. Ruiz. Paras: interactive parameter space exploration for association rule mining. In *SIGMOD*, pages 1017–1020, June 2013.

[16] B. Nag, P. M. Deshpande, and D. J. DeWitt. Using a knowledge cache for interactive discovery of association rules. In *KDD*, pages 244–253, 1999.

[17] R. Ng, L. V. S. Lakshmanan, J. Han, and T. Mah. Exploratory mining via constrained frequent set queries. *SIGMOD Rec.*, pages 13–24, 1999.

[18] E. H. Simpson. The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society (Series B)*, 13:238–241, 1951.

[19] R. Srikant and R. Agrawal. Mining generalized association rules. In *VLDB*, pages 407–419, 1995.

[20] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *SIGMOD Rec.*, volume 25, pages 1–12, 1996.

[21] Y. Theodoridis, E. Stefanakis, and T. K. Sellis. Efficient cost models for spatial queries using r-trees. *IEEE Trans. Knowl. Data Eng.*, 12(1):19–32, 2000.

[22] C.-Y. Wang, S.-S. Tseng, and T.-P. Hong. Flexible online association rule mining based on multidimensional pattern relations. *Inf. Sci.*, pages 1752–1780, 2006.

[23] T. Wu, Y. Chen, and J. Han. Association mining in large databases: A re-examination of its measures. In *PKDD*, pages 621–628, 2007.

[24] M. J. Zaki and C.-J. Hsiao. Charm: An efficient algorithm for closed itemset mining. In *SIAM SDM*, 2002.