Efficient Mining of Constrained Correlated Sets*

Gösta Grahne
Concordia University
grahne@cs.concordia.ca

Laks V. S. Lakshmanan
Concordia University & IIT – Bombay
laks@cs.concordia.ca

Xiaohong Wang
Concordia University
wang_x@cs.concordia.ca

Abstract

In this paper, we study the problem of efficiently computing correlated itemsets satisfying given constraints. We call them valid *correlated itemsets. It turns out constraints can have subtle interactions with correlated itemsets, depending on their underlying properties. We show that in general the set of minimal valid correlated itemsets does not coincide with that of minimal correlated itemsets that are valid, and characterize classes of constraints for which these sets coincide. We delineate the meaning of these two spaces and give algorithms for computing them. We also give an analytical evaluation of their performance and validate our analysis with a detailed experimental evaluation.*

1. Introduction

Ever since the introduction of association rules [1], researchers have studied various problems related to mining interesting patterns from large databases. These include developing faster (both sequential and parallel) algorithms for associations, their quantitative variants, sequential patterns, extensions, and generalizations ([2, 10, 8, 20, 11] are some representative works) and use of partitioning and sampling techniques [18, 15, 22]. More recently, several researchers have argued for the integration of data mining technologies with database management systems (e.g., see [5, 9, 19, 13]). Indeed, Sarawagi et al. [17] study the suitability of different architectures for the integration of association mining with DBMS and study the relative performance tradeoffs. Tsur et al. [23] explore the question of how techniques like the well-known Apriori algorithm can be generalized beyond their current applications to a generic paradigm called query flocks. In previous work [14, 12], Ng et al. identified the following fundamental problems with the present-day model of mining: (i) lack of user exploration and guidance (e.g., expensive computation undertaken without user's approval), and (ii) lack of focus (e.g., cannot limit computation to just a subset of rules of interest to the user). Based on the idea that computation of frequent itemsets forms a fundamental core step in the mining of several kinds of rules, such as association rules, they addressed [14, 12] the above problems in the context of finding frequent itemsets. The main idea is to let the user express her focus, using constraints drawn from a rich class of constraint constructs including domain, class, and SQL-style aggregate constraints, that can capture application semantics. Their algorithms exploit properties of these constraints in pruning the search space, for efficiently computing frequent sets that satisfy user-specified application specific constraints. Srikant et al. [21] have considered mining associations satisfying constraints corresponding to a taxonomy.

It has been recognized that associations are not appropriate for all situations (e.g., see [4]). There is a need to explore alternate patterns/rules. One such notion is *correlation*: Brin et al. [4] have studied the problem of efficiently finding (minimal) correlated (or dependent) sets of objects from large databases., and Silverstein et al. [20] have extended the work further for mining causality. Brin et al. base their definition of correlation on the chi-squared metric, which is widely used by statisticians for testing independence. The idea is that a set is said to be correlated with probability α provided its chi-squared metric exceeds the expected chi-squared value corresponding to the probability α. In analogy to the classical framework of associations, where frequency (support) of itemsets is used as a measure of statistical significance, Brin et al. use a notion of CT-support[1] as a measure of statistical significance of an itemset.

Applications for mining minimal correlated sets satisfying given constraints naturally arise.[2] E.g., a manager of a supermarket may want to verify whether customers who do not want to spend a lot of money overall, only buy the cheaper items. [3] The conjunction of constraints $S.\texttt{price} < c$ & $sum(S.\texttt{price}) < maxsum$ captures this situation. Both constraints are *anti-monotone*, meaning if a set satisfies the constraint, then so does every subset. In addition, the first

* Work supported in part by grants from NSERC, IRIS, and Concordia FRDP.

[1] CT stands for contingency table, explained later.

[2] For the sake of concreteness, we pick a specific application domain – market basket analysis. Our remarks hold for other applications as well.

[3] For the same total price, they prefer to buy more cheap items than fewer expensive items.

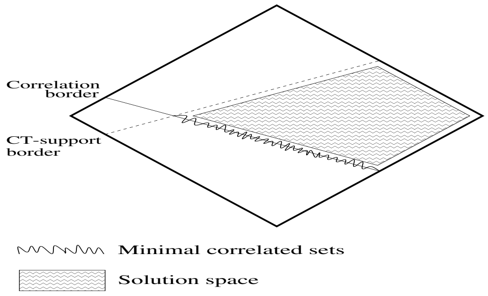


Figure A. The borders of correlation and CT-support in the itemset lattice

constraint satisfies a property called *succinctness*, first defined in [14]. This intuitively means this constraint can be pushed deep down an Apriori-style level-wise algorithm so that it effects pruning even before anti-monotonicity takes effect. As another example, the manager may just want to find whether there is any correlation among items of a single type, for use in mapping items to departments and in shelf planning. The constraint $|S.\mathtt{type}| = 1$ corresponds to this situation. This constraint is also anti-monotone. In a third case, the manager may especially be interested in the correlations of those items whose total price is greater than a certain value, described by the constraint $sum(S.\mathtt{price}) > minsum$. This constraint is neither anti-monotone nor succinct. (See [14, 12] for a thorough analysis of various constraints and their use in pruning optimization of constrained frequent set queries.)

In this paper, we are concerned with pruning optimization of constrained correlation queries. We address the question: *given a set of constraints, how can we efficiently find itemsets that are CT-supported, correlated, and are valid w.r.t. given constraints?* At first sight, a straightforward extension of the techniques in [14] might seem to solve this problem. If we ask for *all* such itemsets, this is indeed the case. However, Brin et al. [4] show that being correlated is a monotone (upward closed) property: all supersets of a correlated set are also correlated, while being CT-supported is an anti-monotone (downward closed) property: all subsets of a CT-supported set are CT-supported. Based on this, they make a case for computing just the minimal correlated (and CT-supported) sets. Figure 1 shows the solution space corresponding to itemsets that are both correlated and CT-supported. The lower border of the figure corresponds to the minimal itemsets in this space. The rationale then is that the user might be interested in the smallest "objects" in the space rather than in all of them. Indeed, knowing that, say bread and butter are correlated is informative, while given this, it is less interesting to know additionally that bread, butter, and cereal are correlated, or that the set consisting of bread, butter, cereal, and toothpaste is statistically insignificant.

Now, consider the problem of finding all itemsets that are CT-supported, correlated, and valid w.r.t. given constraints, which in addition, are *minimal*. The first difficulty is that there are two ways of interpreting this minimality. This leads to two notions of answer sets – (i) valid minimal correlated and CT-supported itemsets and (ii) minimal valid correlated and CT-supported itemsets. As we will show, these are not always identical. There might be interest in computing either of these answer sets, depending on the application. Different techniques are called for depending on which of these answer sets is desired by the user. We shall show that there are circumstances under which both answer sets coincide. A second difficulty comes from constraints which are monotone. Two examples of monotone constraints from the market basket domain are $sum(S.\mathtt{price}) \geq 1000$ and $min(S.\mathtt{price}) \leq 50$, for itemset S. We will see that a direct application of the techniques in [14] for the problem studied in this paper may yield incorrect answers when monotone constraints are considered. Intuitively, monotone constraints exhibit a behavior similar to the property of being correlated, and this should be reflected in the way they are handled in pruning the search space. There is no analog of this in the framework of [14, 12].

In this paper, we make the following contributions:

- We show that in general the answer set (i) valid minimal correlated and CT-supported itemsets is a proper subset of (ii) minimal valid correlated and CT-supported itemsets, and that they coincide whenever all constraints in the user query are anti-monotone.
- We develop techniques for computing either of the answer sets above. Based on this, we propose a basic algorithm as well as an efficient algorithm for computing either answer set—Algorithms BMS+ and BMS++ for answer set (i) and BMS* and BMS** for answer set (ii).
- We analytically articulate the reason why Algorithm BMS++ (resp., BMS**) is more efficient than Algorithm BMS+ (resp., BMS*). Besides, for the case where all constraints in the user query are anti-monotone, we show that Algorithm BMS++ is the most efficient among the four.
- We conducted a series of experiments to validate our analysis, using synthetic data generated using two different methods. We present performance results from our experiments.

2. Constrained Correlations

2.1 Correlation Queries

Brin et al. [4] approach correlation through the notion of dependence. Two items are dependent provided the probability of occurrence of one given the other is different from the absolute probability of the first. They show dependence is

CT	Doughnuts	$\overline{\text{Doughnuts}}$	Row Sum
coffee	30	39	69
$\overline{\text{coffee}}$	20	11	31
Col Sum	50	50	100

Figure B. Example Contingency Table.

a monotone property in that every superset of a dependent itemset is also dependent. Thus, there is interest in finding minimal dependent sets. For measuring dependence, they use the chi-squared statistic. This can be obtained by constructing the contingency table for the itemset in question. Intuitively, the *contingency table* of an itemset S is a table that lists the count, in a given database D, of every minterm involving S. E.g., Figure B, adapted from [4], shows a possible contingency table for the itemset $\{\text{coffee}, \text{doughnuts}\}$.

The chi-squared statistic itself is calculated from a contingency table as $\chi^2 = \Sigma_{r \in \text{minterms}(S)} (O(r) - E(r))^2 / E(r)$, where $O(r)$ is the observed number of occurrences of the minterm r, while $E(r)$, its expected value, is calculated under the independence assumption [4]. Corresponding to the contingency table, there is a degree of freedom, which is always 1 for boolean variables. In addition, there is a corresponding p value, a value in $[0, 1]$, which indicates the probability of witnessing the observed counts if the items in question were really independent. A low p value is thus grounds for rejecting the independence hypothesis. More concretely, we say an itemset is dependent (or correlated) at significance level α provided the p value corresponding to the chi-squared statistic calculated for this set is at most $1 - \alpha$.

Besides being correlated, a set must exhibit some kind of statistical significance. In [4], the authors impose the following measure of significance. Let s be a user specified minimum support threshold and $p\%$ be a user supplied cutoff percentage value. To be considered statistically significant, an itemset S must be such that at least $p\%$ of the cells in the contingency table for S must have their support not less than s. This is a property called *CT-supportedness* which, like frequency, can be readily shown to be anti-monotone. Brin et al. [4] give an efficient algorithm for finding all minimal correlated and statistically significant sets, where the parameters $\alpha, s, p\%$ are all chosen by the user.

2.2 Adding Constraints

Intuitively, a *constrained correlation query* asks for itemsets that are CT-supported and correlated, and further satisfy a set of constraints. We refer the reader to [14] for an exposition of the constraint language used and illustrate it via an example. The query $\{S \mid S \text{ is CT-supported and correlated } \ \& \ \mathit{snacks} \notin S.\mathtt{type} \ \& \ \{\mathit{soda}, \mathit{frozenfood}\} \subset S.\mathtt{type} \ \& \ \mathit{max}(S.\mathtt{price}) < 50 \ \& \ \mathit{sum}(S.\mathtt{price}) \geq 100\}$ asks for CT-supported and correlated itemsets which do not include any snack items, include at least one soda item and at least one frozen food item, and further with a maximum price less than \$50 and a total price of at least \$100. Formally, a *constrained correlation query* is an expression of the form $\{S \mid S \text{ is correlated and CT-supported } \ \& \ S \text{ satisfies } \mathcal{C}\}$, where $\mathcal{C}$ is a conjunction of constraints drawn from the class of domain, class, and SQL-style aggregation constraints.

We recall some basic notions from previous literature. A constraint C is *monotone* provided every superset of a set that satisfies C also satisfies C. It is *anti-monotone* provided every subset of a set that satisfies C also satisfies C. In this paper we only consider constraints that either anti-monotone or monotone. The following lemma shows that this still allows for a rich variety of constraints to choose from.

Lemma 1 Let C be any constraint of one of the following forms:

1. $agg(S.A) \ \theta \ c$, where *agg* is one of *max, min, sum, count* θ is one of $\leq, <, \geq, >$, and A is an attribute with a non-negative domain, and c is a value from it.
2. $CS \ \theta \ S.A$, where CS is a constant set drawn from a domain compatible with that of attribute A, and θ is one of $\subset, \not\subset$.
3. $CS \cap S.A \ \theta \ \emptyset$, where CS is a constant set drawn from a domain compatible with that of attribute A and θ is one of $=, \neq$.

Then C is either anti-monotone or monotone. ■

Indeed, a large portion of the constraints allowed in the constraint language introduced in [14] are either monotone or anti-monotone, according to Lemma 1. Two notable kinds of exceptions are: (i) constraints involving average and (ii) those of the form $agg(S.A) = c$, where *agg* is one of *min, max, sum, count*. We will discuss average constraints in Section 6. Note that a constraint of the form $agg(S.A) = c$ can be broken into $agg(S.A) \leq c \ \& \ agg(S.A) \geq c$. From the proof of the lemma above, it can be shown that one of the conjuncts must be monotone and the other anti-monotone. The techniques we develop in this paper can handle any conjunction of such constraints. Thus, our focus on constraints which are either monotone or anti-monotone does not restrict the expressive power too much.

We next define answer sets for constrained correlation queries. Brin et al. [4] argue that minimal CT-supported and correlated sets capture the essence of answering correlation queries (without constraints). In keeping with this rationale, it is appropriate to build in some notion of minimality in defining the answer sets. The first definition is obtained by taking the definition of Brin et al. and imposing the condition that itemsets must be valid w.r.t. the constraints in the query.

Definition 1 Let $Q = \{S \mid S \subset \mathtt{Item} \ \& \ \mathcal{C}\}$ be a constrained correlation query. Then the set of *valid minimal answers* of Q is given by $\textsc{ValidMin}(Q) = \{S \mid$

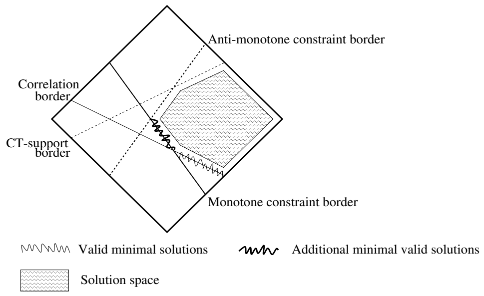


Figure C. The valid minimal solutions and minimal valid solutions.

S is a minimal correlated and CT-supported itemset & S satisfies $\mathcal{C}\}$.

As an example, consider the query $Q_1 = \{S \mid S \subset \texttt{Item}\ \&\ max(S.\texttt{price}) \geq 100\}$. VALIDMIN($Q_1$) consists of all those minimal correlated and CT-supported itemsets, which in addition, have their maximum price not less than \$100.

A second definition of answer sets is obtained by considering the space of all correlated, CT-supported, and valid itemsets and asking for the minimal ones among them. This is meaningful only when the space of all answers is is a single region bounded from above and below by well-defined borders. This is indeed the case for the scenario studied by Brin et al. [4] wherein correlation forms the lower border and CT-support forms the upper border, (see Figure 1). When we throw in constraints, we can continue to have such a well-defined space as long as each constraint considered is either monotone (like correlation) or anti-monotone (like CT-support). Constraints of this type form either a lower or an upper border as the case may be and the answers we are looking for are within the region bounded by all these borders. It thus makes sense to ask for the minimal answers in this space, as they intuitively give us the smallest objects which are interesting.

Definition 2 Let Q be a constrained correlation query $Q = \{S \mid S \subset \texttt{Item}\ \&\ \mathcal{C}\}$ such that each constraint in $\mathcal{C}$ is either monotone or anti-monotone. Then the set of *minimal valid answers* is given by MINVALID(Q) $= \{S \mid S \subset \texttt{Item}\ \&\ S$ satisfies $\mathcal{C}$ & S is CT-supported and correlated & S is minimal$\}$.

For the query Q_1 above, MINVALID(Q_1) consists of all answers which are valid, CT-supported, and correlated, and are minimal among all such objects, i.e. none of their proper subsets satisfy these properties.

The two sets VALIDMIN and MINVALID are illustrated in Figure 2.2. It is easy to see that for any query Q, VALIDMIN(Q) $\subseteq$ MINVALID(Q): any minimal CT-supported and correlated sets that are also valid must be minimal sets that satisfy all three of these conditions. However, there are cases where VALIDMIN(Q) is a proper subset of MINVALID(Q). To see why, consider an example where the domain of `Item` has five items, 1, ..., 5, representing milk, bread, butter, cereal, cheese. For simplicity, let item i have price \$$i$. Suppose that all itemsets of size 2 are CT-supported and correlated, and further assume that all itemsets up to size 4 are CT-supported. Let the constraint be $C \equiv max(S.\texttt{price}) \geq 5$. Then VALIDMIN($Q$) $= \{\{i,j\} \mid i \geq 5 \vee j \geq 5\}$. In particular, the set $\{milk, bread\}$, which is both CT-supported and correlated, is not valid. However, the set $\{milk, bread, cheese\}$ is valid. It is also CT-supported and correlated. So, $\{milk, bread, cheese\} \in$ MINVALID(Q) but $\{milk, bread, cheese\} \notin$ VALIDMIN(Q).

In summary we have the following result:

Theorem 1 Let Q be a constrained correlation query $Q = \{S \mid S \subset \texttt{Item}\ \&\ \mathcal{C}\}$ such that each constraint in $\mathcal{C}$ is either monotone or anti-monotone. Then

1. VALIDMIN(Q) $\subseteq$ MINVALID(Q).
2. If all constraints are anti-monotone, then VALIDMIN(Q) = MINVALID(Q). ■

An additional property of constraints that we shall exploit in this paper is that of succinctness [14]. Let us denote the solution space of a constraint C as $\text{SAT}_C(\texttt{Item}) = \{S \mid S \subset \texttt{Item}\ \&\ S$ satisfies $C\}$. A constraint C is *succinct* provided there are itemsets $I_1, \ldots, I_k \subset \texttt{Item}$ such that: (i) each I_j can be expressed as $I_j = \sigma_{p_j}(\texttt{Item})$ for some selection condition p_j, $1 \leq j \leq k$, and (ii) the solution space $\text{SAT}_C(\texttt{Item})$ can be written as an expression involving powersets of $I_1, \ldots, I_k$ using union and minus. An example is the constraint $C_1 \equiv max(S.\texttt{price}) \leq 100$. Let $I_1 = \sigma_{\texttt{price} \leq 100}(\texttt{Item})$. Then $\text{SAT}_{C_1}(\texttt{Item}) = 2^{I_1}$.[4] As another example, for the constraint $C_2 \equiv \{beer, chips\} \subset S.\texttt{type}$, define $I_1 = \sigma_{\texttt{type}=beer}(\texttt{Item})$, $I_2 = \sigma_{\texttt{type}=chips}(\texttt{Item})$, and $I_3 = \sigma_{\texttt{type}\neq beer\ \&\ \texttt{type}\neq chips}(\texttt{Item})$. Then $\text{SAT}_{C_2}(\texttt{Item}) = 2^{\texttt{Item}} - 2^{I_1} - 2^{I_2} - 2^{I_3} - 2^{I_1 \cup I_3} - 2^{I_2 \cup I_3}$. In this expression, all itemsets that violate C_2 are eliminated from $2^{\texttt{Item}}$. The main value of succinctness is that for a succinct constraint C, we can generate all and exactly the itemsets in the solution space $\text{SAT}_C(\texttt{Item})$ without recourse to generating all possible itemsets and testing them one by one for constraint satisfaction. It was shown in [14] that all succinct constraints C have a *member generating function* (MGF) of the form $\text{SAT}_C(\texttt{Item}) = \{X_1 \cup \cdots \cup X_k \mid X_j \subset \sigma_{p_j}(\texttt{Item}), 1 \leq j \leq k\ \&\ X_j \neq \emptyset, 1 \leq j \leq m,$ for some $m \leq k\}$. As an example, $\text{SAT}_{C_1}(\texttt{Item}) = \{X \mid X \subset \sigma_{\texttt{price} \leq 100}(\texttt{Item})$. As another example, $\text{SAT}_{C_2}(\texttt{Item}) = \{X_1 \cup X_2 \cup X_3 \mid X_1 \subset \sigma_{\texttt{type}=beer}(\texttt{Item})\ \&\ X_2 \subset \sigma_{\texttt{type}=chips}(\texttt{Item})\ \&\ X_3 \subset$

[4] Excluding empty sets is a simple technicality.

Algorithm BMS+
Input: A chi-squared significance level α, support s, support fraction p, a set of constraints $\mathcal{C}$, and basket data D.
Output: The set of all valid minimal correlated itemsets from D.
Method:

1. Run Algorithm BMS up to step (5) and compute the set SIG containing all minimal CT-supported and correlated sets;
2. output those sets in SIG that satisfy the constraints $\mathcal{C}$;

Figure D. Algorithm BMS+.

```
while CAND_k ≠ ∅ {
    for each S ∈ CAND_k {
        if (S satisfies C_{am s̄}) {
          construct CT(S);
          if (S has CT-support ≥ p)
            if (CT(S) has chi-squared value ≥ χ²_α) {
              if (S satisfies C_{m s̄}) {
                add S to SIG; }
            else add S to NOTSIG; } }
    }
}
```

Figure E. SIG and NOTSIG for Algorithm BMS++.

$\sigma_{\texttt{type}\neq beer\ \&\ \texttt{type}\neq chips}(\texttt{Item})\ \&\ X_1 \neq \emptyset\ \&\ X_2 \neq \emptyset\}$. It was also shown that MGFs for individual succinct constraints can be combined into an MGF for their conjunction [14].

3. Algorithms for Constrained Correlations

We first review Brin et al.'s algorithm, referred to as Algorithm BMS below, for computing minimal correlated and CT-supported sets. This algorithm exploits the properties that CT-supportedness is anti-monotone and being correlated is monotone. The former property is exploited in an Apriori-style pruning: process sets from the itemset lattice from the bottom up, level by level, pruning candidate sets which cannot be CT-supported. The latter property is exploited by arguing that minimal sets capture the essence of the answers to correlation queries. Thus, the moment we find a (minimal) correlated and CT-supported set, there is no need to consider its supersets. For lack of space, we refer the reader to [4] for details. We recall their convention that SIG refers to the set of minimal correlated sets, while NOTSIG is the set of CT-supported sets which are not correlated, found so far.

3.1 Computing Valid Minimal Answers

Our first algorithm for computing valid minimal answers of a constrained correlation query is obtained by a straightforward adaptation to Algorithm BMS.

Clearly, Algorithm BMS+ is naive in that it completely ignores the selectivity and potential pruning power that may be effected by the constraints. Our second algorithm is obtained by making the following modifications to Algorithm BMS.

I. Preprocessing: Algorithm BMS considers all pairs of frequent items as candidate sets of size 2. When constraints are present, we can improve matters as follows. Firstly, split the set of query constraints into $\mathcal{C} = \mathcal{C}_{ams} \cup \mathcal{C}_{am\overline{s}} \cup \mathcal{C}_{ms} \cup \mathcal{C}_{m\overline{s}}$, respectively into constraints that are succinct and anti-monotone, anti-monotone but not succinct, succinct and monotone, and monotone but not succinct. We sometimes denote $\mathcal{C}_{ams} \cup \mathcal{C}_{am\overline{s}}$ as $\mathcal{C}_{am}$, the set of anti-monotone constraints and $\mathcal{C}_{ms} \cup \mathcal{C}_{m\overline{s}}$ as $\mathcal{C}_m$, the set of monotone constraints.

$\text{GOOD}_1 = \{i \mid i \in \texttt{Item}\ \&\ \{i\} \text{ satisfies } \mathcal{C}_{am}\}$ denotes the 1-itemsets that satisfy all anti-monotone constraints. Let $\text{CAND}_1^+ = \{i \mid i \in \text{GOOD}_1\ \&\ \{i\} \text{ satisfies } \mathcal{C}_{ms}\}$ and let $\text{CAND}_1^- = \text{GOOD}_1 - \text{CAND}_1^+$. Now, let $L_1^+ = \{i \mid i \in \text{CAND}_1^+\ \&\ O(i) \geq s\}$ and $L_1^- = \{i \mid i \in \text{CAND}_1^-\ \&\ O(i) \geq s\}$.

In the preprocessing stage, our new algorithm computes the sets L_1^+ and L_1^- as suggested above. This can be done in one scan of the database.

II. Forming candidate sets: For the sake of ease of exposition, we will assume that the MGF for the conjunction of all succinct constraints in the query is of the form $\{X_1 \cup X_2 \mid X_1 \subset \sigma_{p_1}(\texttt{Item})\ \&\ X_2 \subset \sigma_{p_2}(\texttt{Item})\ \&\ X_1 \neq \emptyset\}$. Extension to the general forms of MGFs (see Section 2) is straightforward.[5] Candidate sets of size two are formed as follows.

$\text{CAND}_2 = \{\{i_1, i_2\} \mid i_1 \in L_1^+\ \&\ i_2 \in (L_1^+ \cup L_1^-)\}$.

More generally, for $k > 2$, we set CAND_k to contain all k-itemsets S such that:
$\forall S' : (S' \subset S\ \&\ |S'| = k-1\ \&\ S' \cap L_1^+ \neq \emptyset \Rightarrow S' \in \text{NOTSIG})$. The rationale is as follows. Consider a k-itemset S and two $(k-1)$-subsets of $S - S_1, S_2$ such that $S_1 \cap L_1^+ \neq \emptyset$ and $S_2 \cap L_1^+ = \emptyset$. Now, by virtue of the way L_1^+ and L_1^- are computed and the way candidate sets are formed, we can see that $S_1 \in \text{CAND}_{k-1}$, but $S_2 \notin \text{CAND}_{k-1}$. In other words, we would construct contingency tables for all subsets of S that are valid with respect to $\mathcal{C}_{am} \cup \mathcal{C}_{ms}$, and for none of its subsets which are invalid with respect to these constraints. Thus the candidate formation logic of Algorithm BMS is modified to reflect this.

III. Computation of the SIG and NOTSIG sets: The main difference is that we have to check each set that is a potential member of SIG for satisfaction of all non-succinct constraints. Of these, the anti-monotone constraints are handled similarly to the way CT-support test is performed, while the monotone ones are handled similarly to the way correlation is checked. The following pseudocode summarizes this modification.

[5] A subtle point, however, is that if a monotone succinct constraint requires more than one witness, then we cannot include it in L_1^+. It should be enforced later, much like $\mathcal{C}_{m\overline{s}}$, so we correctly compute valid minimal answers.

We refer to the algorithm obtained by applying modifications I-III above to Algorithm BMS as Algorithm BMS++, or "Constrained BMS for valid minimal answers."

3.2 Computing Minimal Valid Answers

First, we give a straightforward algorithm for computing minimal valid answers. In this algorithm, we will need to use the sets SIG and NOTSIG in a context different from that used by Algorithm BMS. So, assume the sets used by Algorithm BMS are renamed to SIG' and NOTSIG'. The idea is that when Algorithm BMS finishes, it computes all minimal CT-supported and correlated itemsets and leaves them in SIG'. Of these, we can add those that satisfy the constraints to the set SIG, which will eventually contain all minimal valid answers. At this point, SIG will contain all valid minimal answers.[6] The difficulty now is that the remaining minimal valid answers *cannot* be found directly from SIG', NOTSIG'. For this, we need to perform an upward, level-by-level sweep of the itemset lattice again, once again checking for CT-support and satisfaction of all monotone constraints. Notice that we do *not* need to check for correlation (i.e. chi-squared test) since the sets being examined now are supersets of sets known to be correlated. The complete naive algorithm is given in Figure F.

Next, we give an algorithm that exploits the pruning effected by the query constraints as early as possible. As with Algorithm BMS++, we will present this algorithm by describing the modifications to be made for Algorithm BMS.

I. Preprocessing: This is identical to that for Algorithm BMS++. In particular, we compute the sets L_1^+, L_1^- as outlined there.[7]

II. Formation of candidate sets: The formation of CAND_k, for $k \geq 2$, is identical to that in Algorithm BMS++.

III. Computation of the SIG and NOTSIG sets: First, we compute supported sets, $\text{SUPP}_k, k \geq 2$, using only CT-support and anti-monotone constraints and then use it along with the monotone constraints to compute NOTSIG and SIG (see Figure G).

We refer to the algorithm obtained by applying modifications I-III described in this section to Algorithm BMS, as Algorithm BMS** or "Constrained BMS for minimal valid answers."

We have the following result, showing the correctness of the various algorithms presented in this paper.

Theorem 2 1. Algorithms BMS+ and BMS++ correctly compute all and only valid minimal answers.
2. Algorithms BMS* and BMS** correctly compute all and only minimal valid answers. ■

[6] Recall that all valid minimal answers are also minimal valid answers.

[7] Regardless of the number of witnesses involved, we can incorporate all succinct constraints in L_1^+, unlike for Algorithm BMS++.

Algorithm BMS*
Input: A chi-squared significance level α, support s, support fraction p, a set of constraints $\mathcal{C}$, basket data D.
Output: The set of all minimal valid correlated itemsets from D.
Method:

1 Run Algorithm BMS to compute the sets SIG and NOTSIG; rename those sets to SIG' and NOTSIG';

2 NOTSIG = SIG = $\emptyset$;

3 for each $S \in \text{SIG}'$ {

 3.1 if (S satisfies $\mathcal{C}_{am}$)
  if (S satisfies $\mathcal{C}_m$)
   add S to SIG;
  else add S to NOTSIG;}

4 Let k = (the least cardinality of any set in NOTSIG) + 1;

5 Set CAND_k to contain all k-sets S such that $(\forall S' : S' \subset S \;\&\; |S'| = k-1 \Rightarrow S' \in \text{NOTSIG})$;
(i.e. all $(k-1)$-subsets of S satisfy $\mathcal{C}_{am}$ but not $\mathcal{C}_m$.)

6 while $\text{CAND}_k \neq \emptyset$ {

 6.1 for each $S \in \text{CAND}_k$ {

  6.1.1 construct $CT(S)$;
  6.1.2 if (S has CT-SUPPORT $\geq p$)
   if (S satisfies $\mathcal{C}_m$)
    add S to SIG;
   add S to NOTSIG; }

 7 $k++$;

 8 Set CAND_k to contain all k-sets S such that all $(k-1)$-subsets of S are in NOTSIG;}

9 output SIG;

Figure F. Algorithm BMS*.

$k = 2$;

while ($\text{CAND}_k \neq \emptyset$) {

 $\text{SUPP}_k = \emptyset$;
 for each $S \in \text{CAND}_k$ {
  if (S satisfies $\mathcal{C}_{am\overline{s}}$) {
   construct $CT(S)$;
   if (S has CT-support $\geq p$)
    add S to SUPP_k; }
  }
 $k++$;
 Form CAND_k;
 }

$k = 2$; $\text{SIG} = \text{NOTSIG} = \emptyset$; $C_k = \text{SUPP}_k$;

while $C_k \neq \emptyset$ {

 for each $S \in C_k$ {

  if ($CT(S)$ has chi-squared $\geq \chi^2_\alpha$ && S satisfies $\mathcal{C}_{m\overline{s}}$)
   add S to SIG;
  else add S to NOTSIG; }

 $k++$;
 (a) Set C_k to contain all k-sets in SUPP_k such that $\forall S' : S' \subset S \;\&\; |S'| = k-1 \Rightarrow S' \in \text{NOTSIG}$; }

Figure G. SIG and NOTSIG for Algorithm BMS.**

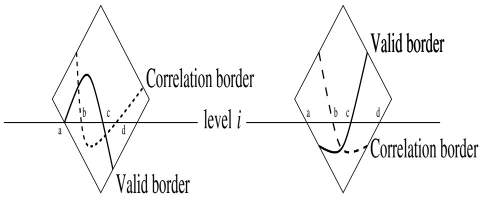


Figure H. The interplay between constraints and correlation

3.3 Analysis

We will now analyze the number of sets each of the four algorithms needs to consider. Clearly, the number of sets is the dominating parameter since it involves database scanning. Computing CT-support etc. are cpu operations, and much less expensive.

Assume a fixed set I, a fixed database D, and some fixed cut-offs α, s, and p. Let c_i be the number of correlated sets at level i in the itemset-lattice. Likewise, let v_i be the number of valid sets at level i. Also, let cv_i be the set of correlated and valid sets at level i. Clearly $cv_i \leq c_i$ and $cv_i \leq v_i$, for any level i. This is illustrated in Figure H. The lattice on the left uses anti-monotone (downward closed) constraints, while the lattice on the right uses monotone (upward closed) constraints. In both lattices, the interval $[b, d]$ corresponds to c_i, the interval $[a, c]$ corresponds to v_i, and the interval $[b, c]$ corresponds to cv_i.

Let k be the highest level on which there are (minimal) correlated sets, and let l be the highest level on which there are valid sets Let |BMS+| be the number of sets algorithm BMS+ needs to consider. The meaning of |BMS++|, and |BMS*|, |BMS**| is similar, mutatis mutandis. We can show that $|\text{BMS+}| = \sum_{i=1}^{k} c_i$, $|\text{BMS}++| = \sum_{i=1}^{min(k,l)} cv_i$, $|\text{BMS}*| = \sum_{i=1}^{k} c_i + \sum_{i=k}^{l} v_i$, $|\text{BMS}**| = \sum_{i=1}^{l} v_i$.

We can now draw the following conclusions: If the query Q contains monotone constraints, algorithms BMS+ and BMS++ compute the set $\text{ValidMin}(Q)$, while algorithms BMS* and BMS** compute the set $\text{MinValid}(Q)$, From the formulas above we see that $|\text{BMS}++| \leq |\text{BMS+}|$. The relationship between BMS* and BMS** depends depends on the relative distributions of the c_i:s and v_i:s. If the selectivity of the constraint is low, it means that $\sum_i v_i \leq \sum_j c_j$. Then $|\text{BMS}**| \leq |\text{BMS}*|$, and we can expect Algorithm BMS** to perform better that Algorithm BMS*. If constraint selectivity is high, the inverse quantitative relationship holds, and Algorithm BMS* is expected to perform better than Algorithm BMS**.

When the query Q contains only anti-monotone constraints, $\text{ValidMin}(Q) = \text{MinValid}(Q)$. We observe that $|\text{BMS}++| \leq |\text{BMS+}|$, and that $|\text{BMS}++| \leq |\text{BMS}**|$. The relationship between between BMS* (BMS+) and BMS** depends as above constraint selectivity. With high selectivity, we expect Algorithm BMS** to perform better that Algorithm BMS*, and with low constraint selectivity Algorithm BMS* is expected to perform better than Algorithm BMS**.

Indeed, as we will see in the next section, our experimental evaluation verifies these expectations.

4 Experiments

Test data: To evaluate the various algorithms presented, synthetic data was used, generated using two different methods. The first method was developed at IBM Almaden Research Center by Aggrawal and Srikant [2], to which we refer the reader for details. In the second method, test data is generated according to a set of prespecified correlation rules, following standard practice in machine learning experiments [7, 6]. While the purpose of the first method is to simulate the "real world", that of the second is to verify that our algorithms do really correctly mine out all the correlation rules, which are known in advance.

In the first method, we varied the number of baskets from 10,000 to 100,000 to study the pruning effect of different algorithms on basket numbers, while keeping the other parameters the same. The average basket size is set to be 20, the average size of large itemsets is set to be 4. The number of items is 1000. In the second method, the synthetic data was generated based on ten given correlation rules. For each rule r_i, the significance level α_i is set to 0.95, the support threshold s_i is a random value between 70% and 90% of the number of baskets. Therefore, each basket contains a subset of the correlation rules. All other parameters are the same as mentioned above. Randomized items are picked up in case the correlation rules do not generate enough items for a particular basket. To see the effects of different constraints on mining correlations, constraint selectivity was also varied through separate experimental sets, using the same data.

In all experiments the minimum support and CT- support were kept constant. The threshold for support and CT- support were set to be 25%.[8] We used a confidence level of 0.9 for the χ^2-tests. All the experiments were conducted on a Pentium PC with a 200 MHz processor and 64 MB of memory.

4.1 Experimental evaluation

Anti-monotone and succinct constraint: In the first set of experiments, the anti-monotone and succinct constraint $\max(S.price) \geq v$ was used to compare the algorithms. Figure 1(a & b) shows the cpu usage as a function of the num-

[8] We ran the experiments for other thresholds too and observed little variation in the trends of the results. We only show the results for one choice here.

ber of baskets for the three algorithms when the constraint-selectivity (proportion of items with price at least v) is set to be 50%, and the number of items is 1000. We chose a conservatively high selectivity to test the behavior of the algorithms. As mentioned previously, for such constraints, BMS* becomes BMS+, and all four algorithms compute the same results.

For all the algorithms, the results showed a similar linear trend of the cpu usage. The correlations that were mined out only contained sets with less than four items. Figure 1(a) (resp., 1(b)) shows the result using synthetic data generated according using method 1 (resp., method 2). As the number of baskets increases, the cpu usage of BMS++ in Figure 1(a) becomes much smaller than the other two algorithms. Compared to BMS+, BMS++ can speed up the process by a factor of 10 to 50 for the experimental range tested. In Figure 1(b), the cpu usage for BMS** is close to BMS++, which is much lower than BMS+. We believe this is caused by the method of generating data.

Figure 2 shows the results of cpu usage as a function of the selectivity of constraints when the basket number is 100,000. In the constraint $\max(S.price) \geq v$, we used different values of v to correspond to different selectivities of the constraints. For BMS+ we see that the cpu usage stays constant as the selectivity increases. On the other hand, the cpu usage for BMS** and BMS++ decrease dramatically as the selectivity decreases. When the selectivity is below 30%, the speed-up from using anti-monotone and succinct constraints can be as high as 50 to 100. Even when the selectivity is 80%, the performance for BMS++ is much better than BMS+, which demonstrates that anti-monotone and succinct constraints can greatly improve the mining performance.

Anti-monotone constraint: Figures 3 (a) and (b) show the amount of cpu usage as a function of number of baskets, for the two types of synthetic data, respectively. In this experiment series, the anti-monotone but not succinct constraint $sum(S.price) \leq Maxsum$ was used. Recall that for anti-monotone constraints, all four algorithms compute the same answer. The results presented in Figures 3 (a) and (b) have a constraint selectivity of 50%. Similar to Figure 1, a linear increasing trends is exhibited for all the algorithms. As the number of baskets increases, the difference of cpu usage between BMS++ and BMS+ increases. At a basket number of 100,000, the cpu usage for BMS++ is about 1/3 that of BMS+, while that for BMS** is either the same as that for BMS+ or about 3/4 that for BMS+ depending on the data set used.

The effect of constraint selectivity is also examined. Figures 4 (a) and (b) show these results when the basket number is set to be 100,000. Unlike the previous constraints, the notion of constraint selectivity does not directly make sense in this case. Instead, we assign the price of each item to be its item number. So item 1 has a price of \$1. At lower values of Maxsum, both BMS** and BMS++ have better performance than BMS+. When the value of Maxsum is close to 4000, there is no pruning effect from the constraint anymore. So BMS+ and BMS++ begin to have the same performance, and the performance for BMS** is much worse than the other two. However, under all circumstances, BMS++ always gives the best performance. BMS** and BMS+ have a cross-over point, below which BMS** has a better performance, and above which BMS+ gives a better performance.

Succinct and monotone constraint: When the constraint is not anti-monotone, the results of valid minimal computation and minimal valid computation is not the same anymore. So, we have to examine the four algorithms separately. As we have mentioned in Theorem 1, Algorithm BMS+ and BMS++ correctly compute all and only valid minimal answers. Algorithm BMS** and BMS* correctly compute exactly the minimal valid answers. The constraint that used is $\min(S.price) \leq v$, which is monotone and succinct.

(i) Valid minimal answers. Figure 5 shows the performance of Algorithms BMS+ and BMS++. The selectivity is set to be 50%. Figure 5 (a) is the result using the first synthetic data set. Figure 5 (b) is the result using the second synthetic data set. At a basket number of 100,000 the cpu usage for BMS++ is about 70% of the cpu usage of BMS+, in spite of this high selectivity.[9]

Figure 6 shows the selectivity effect on these two algorithms when the basket number is 100,000. When the selectivity is 10%, the cpu usage for BMS++ is only 1/3 of that for BMS+. But when the selectivity is above 70%, the pruning effect of the constraint is negligible, and the performance of BMS++ becomes similar to that of BMS+.

(ii) Minimal valid answers. Figure 7 shows the performance of Algorithms BMS* and BMS**. The selectivity is set to be 50%. Figure 7 (a) is the result using the first synthetic data set. Figure 7 (b) is the result using the second synthetic data set. Unlike Figure 5, the gap between the two algorithms is much bigger.

Figure 8 shows the selectivity effect on these two algorithms when the basket number is 100,000. Unlike the computation of valid minimal answers, where BMS++ always performs better than BMS+, for minimal valid answers both algorithms BMS* and BMS** are affected by the selectivity. When the selectivity is below 20%, BMS** has better performance. Above this point, the performance for BMS* becomes better. In Figure 7, we deliberately show this situation when the selectivity is unfairly high at 50%. Figure 8 shows the cross-over point.

Summary: From all the experiments, it can be concluded that for BMS+, the constraint type does not influence the overall performance. Algorithm BMS* is slightly affected by the selectivity. With the increase of the selectivity, the

[9]Note that the higher the selectivity, the less selective the constraint.

cpu usage of BMS* decreases. The performance of algorithm BMS** is heavily dependent on the constraint to do the pruning work. So when the constraint selectivity is low, BMS** has better performance than BMS+, especially when the constraint is both anti-monotone and succinct. When constraint selectivity is high, BMS** does not perform well, it can become 2 to 3 times slower than BMS+ in the worst case. BMS++ shows the best performance under all circumstances. When constraint selectivity is low, BMS++ relies on the constraint to prune the candidate itemsets, and when the constraint selectivity is high, it relies on the upward closed property of being correlated, to do the pruning.

5 Related Work

A general review of relevant literature appears in Section 1. Here, we mainly compare our work with that of Brin et al. [4] and the constrained frequent set framework of [14, 12]. Brin et al. [4] defined their answer set as minimal correlated and CT-supported sets. They claimed this completely characterizes the solution space. Technically, this is true only when one also returns, as part of the answer, some description of the upper border (in their case, CT-support border). In adding constraints to this framework, we have shown that when different kinds of constraints (e.g., monotone, anti-monotone, etc.) are considered, a proper understanding of the solution space is needed before we can even advance minimal sets as a meaningful definition of the answer set. Notice that simply returning minimal answers does not completely cover all answers, unless we also know where the upper border is. On the other hand, when the solution space is a single region (this is the case when all constraints are monotone or anti-monotone), there are is an intuitive appeal to returning minimal answers, as they are in some sense the "smallest objects" in the solution space.

In [14, 12], monotonicity was not exploited, since the answer set there was *all* frequent valid sets. This is in keeping with the classical framework of associations, where all frequent sets are computed (and used for forming associations). Handling both monotone and anti-monotone constraints is a novelty to our work. While it may be argued that [4] already did handle such constraints (since being correlated is monotone and being CT-supported is anti-monotone), we also handle their interaction with succinct constraints, which to our knowledge, is done for the first time.

6 Summary and Future Work

We motivated the problem of finding correlated sets satisfying user-specified constraints. Constraints may be application specific and help the user focus and control the mining task undertaken by the system. We extended Brin et al.'s [4] principle of finding minimal sets to two useful semantics of answering constrained correlation queries – all minimal answers that are valid, *vs.* all minimal valid answers. We gave algorithms for computing the two kinds of answers and brought out their relative performance and the tradeoffs via detailed experiments.

Several questions remain open. Firstly, it is not clear how constraints such as $avg(S.\mathtt{price})\ \theta\ c$, which are neither monotone nor anti-monotone, can be handled. The solution space may not be a single region and instead may have holes in it. Under these conditions, blindly returning only minimal valid answers does not make sense. Defining meaningful answer sets and computing them efficiently for such constraints is an interesting problem. Another question is how can constraints help in mining causations? Finally, as suggested by one of the anonymous referees, it seems possible to optimize Algorithm BMS** even further. We are currently investigating these issues.

Acknowledgement

Thanks to Laurian Staicu for squeezing the graphs down to one page.

References

[1] R. Agrawal et al. Mining association rules between sets of items in large databases. SIGMOD 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB 1994.

[3] R. Agrawal and R. Srikant. Mining sequential patterns. ICDE 1995.

[4] S. Brin et al. Beyond market basket: Generalizing association rules to correlations. SIGMOD 1997.

[5] S. Chaudhuri. Data mining and database systems: Where is the intersection? *Data Engineering Bulletin*, 21:4–8, March 1998.

[6] J. Cheng et al. Improved Decision Trees: A Generalized Version of ID3. In *Proc. of the Fifth International Conference on Machine Learning*. 1988.

[7] P. Clark, and T. Niblett. The CN2 Induction Algorithm. *Machine Learning* 3: 261–283, 1989.

[8] E.-H. Han et al. Scalable parallel data mining for association rules. SIGMOD 1997.

[9] T. Imielinski and H. Mannila. A database perspective on knowledge discovery. CACM 1996.

[10] M. Klemettinen et al. Finding interesting rules from large sets of discovered association rules. CIKM 1994.

[11] F. Korn et al. Ratio rules: A new paradigm for fast, quantifiable data mining. VLDB 1998.

[12] L. V. S. Lakshmanan et al. Optimization of constrained frequent set queries: 2-var constraints. SIGMOD 1998.

[13] H. Mannila et al. in event sequences. *Data Mining and Knowledge Discovery*, 1:259-289, 1997.

[14] R. Ng et al. Exploratory mining and pruning optimizations of constrained associations rules. SIGMOD 1998.

[15] J.S. Park et al. An effective hash-based algorithm for mining association rules. SIGMOD 1995.

[16] J.S. Park et al. Efficient parallel mining for association rules. CIKM 1995.

[17] S. Sarawagi et al. Integrating association rule mining with relational database systems: Alternatives and implications. SIGMOD 1998.

[18] A. Savasere et al. An efficient algorithm for mining association rules in large databases. VLDB 1995.

[19] A. Silberschatz and S. Zdonik. Database systems – breaking out of the box. SIGMOD Record, 26, pp 36–50, 1997.

[20] C. Silverstein et al. Scalable techniques for mining causal structures. VLDB 1998.

[21] R. Srikant et al. Mining association rules with item constraints. KDD 1997.

[22] H. Toivonen. Sampling large databases for association rules. VLDB 1996.

[23] D. Tsur et al. Query flocks: A generalization of association-rule mining. SIGMOD 1998.

Fig 1a: data 1(a.m. & succ)

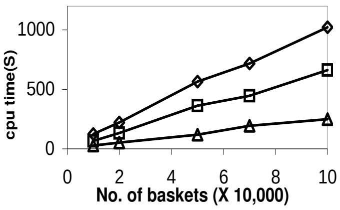


Fig 2a: data 1 (a.m. & succ)

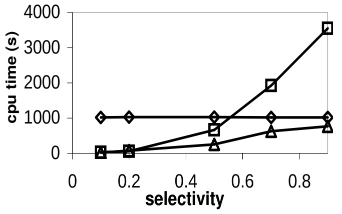


Fig 3a: data 1 (a.m.)

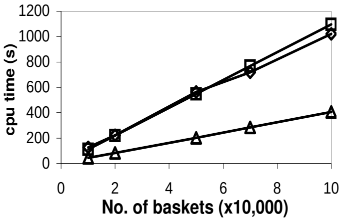


Fig 1b: data 2(a.m. & succ)

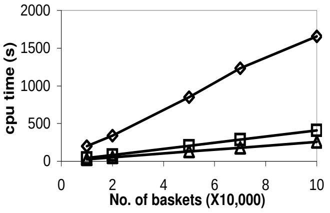


Fig 2b: data 2(a.m. & succ)

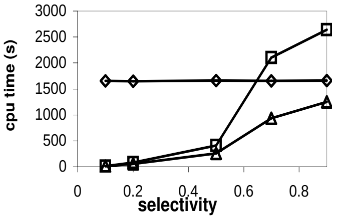


Fig 3b: data 2 (a.m.)

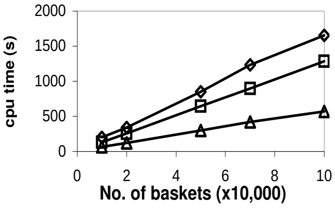


Fig 4a: data 1 (a.m.)

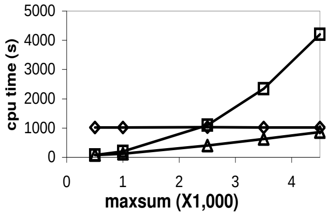


Fig 5a:data 1(mono & succ)

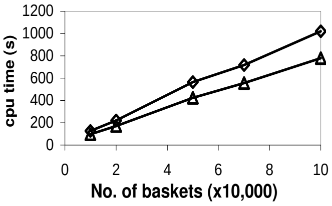


Fig 6a: data1(mono & succ)

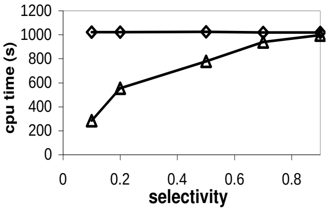


Fig 4b: data 2 (a.m.)

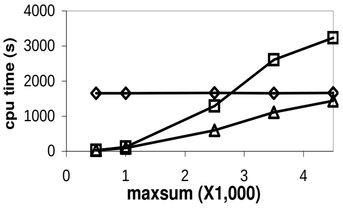


Fig 5b: data 2(mono & succ)

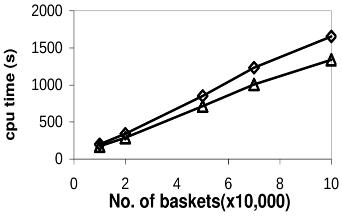


Fig 6b: data2(mono & succ)

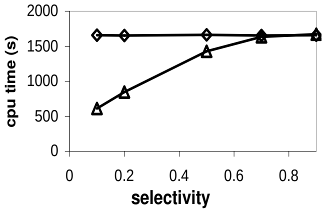


Fig 7a: data1(mono & succ)

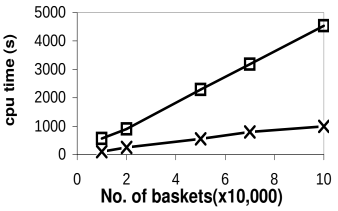


Fig 8a:data 1(mono & succ)

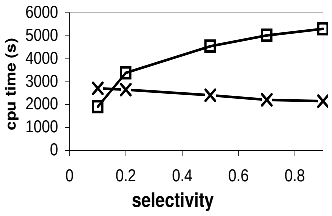


Fig 7b: data2(mono & succ)

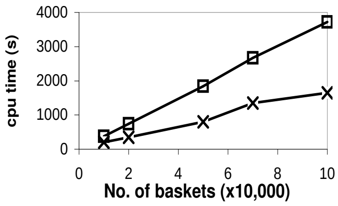
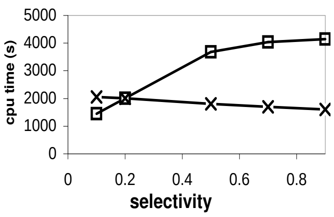


Fig 8b: data2(mono & succ)



BMS+ ◇
BMS++ △
BMS* X
BMS** □