Interactive Mining of Correlations – A Constraint Perspective

Gösta Grahne
Concordia University
grahne@cs.concordia.ca

Laks V.S. Lakshmanan*
Concordia University
laks@cs.concordia.ca

Xiaohong Wang
Concordia University
wang_x@cs.concordia.ca

Abstract

The problem of finding minimal sets of objects, that are correlated and have statistically significant number of occurrences in a database, has recently received considerable attention, as an alternative to association rules. A case for making mining human-centered, interactive, and exploratory via application specific constraints was recently made in the papers [14, 12]. The technical results of these papers were established in the context of efficient computation of frequent sets satisfying user specified constraints.

In this preliminary report, we examine the problem of efficient computation of correlated sets satisfying given constraints. We discuss three algorithms for this purpose and report on the results of their 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 (see [2, 10, 8, 20, 11] for some representative works) and use of partitioning and sampling techniques [18, 15, 21]. More recently, people have recognized that associations are not always appropriate for many situations (e.g., see [4]) and have explored alternate patterns/rules. One such notion is *correlation*. In particular, Brin et al. [4] have studied the problem of efficiently finding (minimal) correlated (or dependent) sets of objects from large databases. They 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. Motwani et al. [20] have extended this work further for mining causality.

It has been recognized recently that the integration of data mining technologies with database management systems is of crucial importance [5]. Furthermore, it has been argued that the fundamental distinction of a data mining system from a statistical analysis program or a machine learning system *should* be that the former: (i) offers an *ad-hoc mining* query language and (ii) supports efficient processing and *optimization of mining queries* [9, 19, 13]. 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. [22] 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], we have argued that for effective ad-hoc data mining, the following fundamental problems with the present-day model of mining need to be solved: (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 that are of interest to the user). It is well-known that the computation of frequent sets forms the fundamental core in the mining of many important types of rules. For instance, formation of associations can be done relatively easily and efficiently once the frequencies of frequent sets are known. Based on this observation, in [14, 12], we solved the aforementioned problems in the context of computing frequent sets. 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. The system then exploits properties of these constraints in pruning the search space for computing frequent sets that satisfy given application specific constraints. We also proposed a two-phase architecture for computing constrained rules. The user can adjust various "parameters" including minimum thresholds for measures like support, confidence, etc., and any of the constraints, and refine phase I computation as often as required. When satisfied with the outcome of phase I, she can instruct the system to go on to phase II, where the significance metric required (e.g., confidence, correlation, etc.) can be specified. Example mining queries in this framework include finding associations between cheaper items and more expensive items, or between sets of items whose type sets are disjoint, etc. While for concreteness of illustration [14, 12] focused on associations, it was claimed there that the framework and optimization opportunities easily extend for other metrics like correlations.

In addition to being correlated, a set must have some kind of statistical significance. In the classical association mining framework, frequency is used as a measure of statistical significance. In [4], the authors consider an alternative measure of statistical significance called *CT-support*. They show that for binary variables the property of being CT-supported is *downward closed* (sometimes called *anti-monotone* [14]), meaning every subset of a CT-supported set is necessarily CT-supported. They also show that the property of being correlated is *upward closed* (*monotone*), meaning every superset of a correlated set is necessarily correlated. Thus, the mining problem becomes one of finding all minimal CT-supported, correlated sets. They provide an efficient algorithm for finding all such sets.

In this preliminary report, we consider the following question: *how can we efficiently find minimal, CT-supported and correlated sets that satisfy given constarints?* We propose and study three algorithms for this purpose. We report on the results of the experiments we ran for comparing the relative performance of these algorithms. Roughly speaking, the first of these algorithms is a naive one and does not take advantage of the pruning effected by the constraints. The second one, while exploiting constraint pruning, adheres to the structure of Apriori-syle algorithms which have a clear separation between phase I, where frequent itemsets are computed, and phase II, where correlations are formed. The last algorithm tries to exploit properties of constraints as well as that of correlation itself, and is based on a tight interaction between phase I and II. We report on the results of our experiments evaluating the relative efficiency of the three algorithms.

2 Background

In this section, we briefly review the following: (i) the constrained mining queries framework introduced in [14, 12] and (ii) the correlation framework introduced in [4].

The key technical contribution of [14, 12] is the use of constraints for (i) capturing application semantics and foucus in mining queries and (ii) their analysis and use in pruning optimizations. A *constrained frequent set query* (CFQ) is an expression of the form $\{(S_1, S_2) \mid \mathit{freq}(S_1) \& \mathit{freq}(S_2) \& \mathcal{C}\}$, where $\mathit{freq}(S)$ is a shorthand for saying the support of itemset S exceeds a user specified minimum threshold, i.e. that S is frequent, and $\mathcal{C}$ is a conjunction of constraints drawn from domain, class, and `SQL`-style aggregate constraints. The following example asks for pairs of frequent sets (S_1, S_2) such that the maximum price of an item in S_1 is no more than \$100, and the average price of an item in S_2 is 500. $\{(S_1, S_2) \mid \mathit{freq}(S_1) \;\&\; \mathit{freq}(S_2) \;\&\; \mathit{max}(S_1.\mathtt{price}) \leq 100 \;\&\; \mathit{avg}(S_2.\mathtt{price}) \geq \mathit{500}\}$. We refer the reader to [14, 12] for many more examples.

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 item. They show that dependence is a monotone property in that every superset of a dependent itemset is also dependent. Thus, one would be interested in finding minimal dependent sets. For measuring dependence, they use the chi-squared statistic. This statistic 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 A, reproduced from [4], shows a possible contingency table for the itemset $\{\text{coffee}, \text{doughnuts}, \text{tea}\}$.

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 count of the number of occurrences of the minterm r, while $E(r)$, its expected value, is calculated under the independence assumption, as outlined in [4]. Corresponding to the contingency table, there is a degree of freedom, which is always 1 for boolean variables. Besides, 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 where 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

Doughnuts	Tea	$\overline{\text{Tea}}$	row sum
coffee	8	40	48
$\overline{\text{coffee}}$	1	2	3
col sum	9	42	51

$\overline{\text{Doughnuts}}$	Tea	$\overline{\text{Tea}}$	row sum
coffee	10	35	45
$\overline{\text{coffee}}$	2	2	4
col sum	12	37	49

Figure A: Example Contingency Table.

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.

The rest of this paper is organised as follows: In section 3, we give examples of mining queries that motivate the usefulness of constraints in mining correlations. In Section 4, We briefly describe three algorithms for computing constrained correlation queries. Section 5 discusses our experiments with the three algorithms on two synthetic data sets.

3 Motivation: Why Constraints on Correlations?

For the sake of concreteness, we pick a specific application domain – market basket analysis. Our remarks hold for other applications as well. For example, a manager of a supermarket may want to find out if is it true that customers who do not want to spend a lot of money overall, only buy the cheaper items. [1] The conjunction of constraints $S.\texttt{price} < c$ & $sum(S.\texttt{price}) < maxsum$ characterizes this situation. Both constraints are anti-monotone. In addition, the first constraint satisfies a property called *succinctness*, first defined in [14]. This intuitively means that 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 between items of a certain type for use in mapping items to departments and shelf planning. The constraint is $|S.\texttt{type}| = 1$ corresponds to this situation. This constraint is anti-monotone and succinct. 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 $sum(S.\texttt{price}) > minsum$. This constraint is neither anti-monotone nor succinct. The reader is referred to [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 primarily concerned with similar optimization of constrained correlation queries: *find all minimal correlated sets satisfying given constraints.*

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

4 Algorithms for Computing CCQs

For lack of space, we only briefly describe the algorithms. The complete details of the algorithms as well as an analysis of their cost will be presented in a forthcoming paper. We call an itemset *valid* provided it satisfies user supplied constraints.

Algorithm 1: The first algorithm is a naive one. We apply the algorithm of Brin et al. [4] for finding (minimal) correlated sets and then check which of them satisfy given constraints. Clearly, this algorithm does not take advantage of given constraints and the potential pruning effected by them.

Algorithm 2: This algorithm is inspired by the constrained Apriori (CAP) algorithm proposed in [14]. The basic idea is to use an Apriori-style algorithm for computing frequent sets, while at the same time applying given constraints (or weaker ones induced by them) at the earliest possible opportunity. This will generate frequent valid sets quite efficiently. We subsequently use this information to find all minimal correlated sets among them. It should be pointed out that CAP was shown in [14, 12] to feature orders of magnitude speedup compared to a naive algorithm based on computing all frequent sets and then checking their validity.

Algorithm 3: Algorithm 2, while taking advantage of given constraints for pruning the search space, ignores a similar pruning effect offered by the upward closed property of dependent sets. Our third algorithm is obtained by integrating the pruning optimization strategies within the algorithm of Brin et al. [4], thus combining pruning effects of constraints and CT-supportedness.

In the next section, we report on our experimental evaluation of the three algorithms.

5 Experiments

5.1 Test data

To evaluate the various algorithms that present above we used synthetic data. Two different methods were used to generate the test data. The first method was developed at IBM Almaden Research Center by

Agrawal and Srikant [2]. In the second method test data was generated according to a set of prespecified correlation rules. This generation method has been widely used, in particular in experiments for machine learning of classification rules [7, 6].

In all experiments the minimum support threshold was set to 25% of the number of transactions. The threshold for CT- support was also set to be 25%. We used a confident level of 90% for the χ^2-tests. All the experiments were conducted on a Pentium PC with a 200 MHz processor and 64 MB of memory.

5.2 Experimental evaluation

In the first set of experiments, the anti-monotone and succinct constraint $\min(S.price) \geq v$ was used to compare the algorithms. Figure 1 shows the cpu usage as a function of the number of baskets for the three algorithms when the constraint-selectivity (proportion of items with price at least v) is set to be 70% and the number of items is 10. Experiments with 50 and 300 items were also conducted, and these results showed the same trends of the cpu usage for the three algorithms. Figure 1(a) is the result using synthetic data generated according to Agrawal and Srikant's method. Figure 1(b) shows the results for the data that generated based on given correlation rules. All the correlation rules that were used in generating the data were indeed found in the results, as well as some other weak correlation rules. The latter are caused by the randomly distributed items in the transactions. Figure 1(b) shows a trend that is much smoother than Figure 1(a) because of the synthetic data used in Figure 1 (b) has less random fluctuation than the data in Figure 1(a). In Figure 1(a), the cpu usage for algorithm 1 and 2 are similar while in figure 1(b), the cpu usage for algorithm 2 is close to algorithm 3, much lower than algorithm 1. We believe that this is caused by the method of generating data. However, from both of the figures we can see that when the number of baskets is small, the cpu usage for all the algorithms are similar. As the number of baskets increases, the cpu usage of algorithm 3 becomes much smaller than the other two algorithms. Compared to algorithm 1, algorithm 3 can speed up the process by a factor of 3 when the number of baskets is 10,000.

Figure 2 shows the results of cpu usage as a function of the selectivity of constraints when the basket number is 10,000. In the constraint $\min(S.price) \geq v$, different values of v correspond to different selectivities of the constraints. An $x - \%$ selectivity means that $x - \%$ of the items have a price larger than v, whatever that value may be. The y-axis represents the cpu time the various algorithms needed to complete the mining. For algorithm 1, both Figure 2(a) and Figure 2(b) shows the cpu usage is not affected by the selectivity. The cpu usage stays constant as the selectivity increases. On the other hand, the cpu usage for algorithm 2 and 3 decrease dramatically as the selectivity decreases. Because anti-monotone and succinct constraints have a strong effect for data mining pruning and both algorithm 2 and 3 take this advantage. When the selectivity is below 30%, the speed-up from using anti-monotone and succinct constraints can be as high as 40. Even when the selectivity is 80%, the performance for algorithm 3 is much better than algorithm 1, which demonstrates that anti-monotone and succinct constraints can greatly improve the mining performance.

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, we use the succinct but not anti-monotone constraint $\min(S.price) \leq v$. The selectivity of the constraint is set to be 50%. The effect of constraint selectivity is also examined and Figures 4 (a) and (b).

The last experiment serial is to investigate the algorithms with an anti-monotone non-succint constraint $sum(S.price) \leq Maxsum$. The results are present Figures 5 (a) and (b) with constraint selectivity 50%, and in Figures 6 (a) and (b) where the constraint selectivity is varying and number of baskets is fixed at 10,000.

From all the above figures, it can be concluded that for algorithm 1, the constraint type does not influence the overall performance, while for algorithm 2, the performance is heavily dependent on the constraint to do the pruning work. So when the constraint selectivity is low, algorithm 2 has better performance than algorithm 1, especially when the constraint has both the anti-monotone and succinct properties. When constraint selectivity is high, algorithm 2 does not perform well, it can become 2 to 3 times slower than algorithm 1 in the worst case. However, algorithm 3, the combination of algorithm 1 and 2, shows the best performance in all cases. When constraint selectivity is low, algorithm 3 relies on the constraint to prune the candidate itemsets, and when the constraint selectivity is high, algorithm 3 relies on the CT-support and chi-square value to do the pruning.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. 1993 ACM-SIGMOD*, pp 207–216.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 1994 VLDB*, pp 487–499.

[3] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. 1995 Int. Conf. Data Engineering*, pages 3-14, Taipei, Taiwan, March 1995.

[4] S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. In *Proc. 1997 ACM-SIGMOD*, pp 265–276.

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

[6] J. Cheng,, U. M. Fayad, K. B. Irani, and Z. Quian. 1988. Improved Decision Trees: A Generalized Version of ID3. In *Proc. of the Fifth International Conference on Machine Learning*. Morgan Kaufmann. San Mateo, California. pp. 100–107, 1988

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

%bibitemFMMT96

[8] E.-H. Han, G. Karypis, and V. Kumar. Scalable parallel data mining for association rules. In *Proc. 1997 ACM-SIGMOD*, pp 277–288.

[9] T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communications of ACM*, 39:58–64, 1996.

[10] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A.I. Verkamo. Finding interesting rules from large sets of discovered association rules. In *Proc. 3rd Int. Conf. Information and Knowledge Management*, pp 401–408, 1994.

[11] F. Korn, A. Labrinidis, Y. Kotidis, and C. Faloutsos. Ratio rules: A new paradigm for fast, quantifiable data mining. In *Proc. 1998 VLDB*, pp 582–593.

[12] L. V. S. Lakshmanan, R. Ng, J. Han, and A. Pang. Optimization of constrained frequent set queries: 2-var constraints. To appear in: *Proc. 1998 ACM-SIGMOD*, May 1999.

[13] H. Mannila, H Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1:259-289, 1997.

[14] R. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proc. 1998 ACM-SIGMOD*, pp. 13–24.

[15] J.S. Park, M.S. Chen, and P.S. Yu. An effective hash-based algorithm for mining association rules. In *Proc. 1995 ACM-SIGMOD*, pp 175–186.

[16] J.S. Park, M.S. Chen, and P.S. Yu. Efficient parallel mining for association rules. In *Proc. 4th Int. Conf. Information and Knowledge Management*, pp 31–36, 1995.

[17] S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. In *Proc. 1998 ACM-SIGMOD*, pp 343–354.

[18] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proc. 1995 VLDB*, pp 432–443.

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

[20] C. Silverstein, S. Brin, R. Motwani, and J. Ullman. Scalable techniques for mining causal structures. In *Proc. 1998 VLDB*, pp 594–605.

[21] H. Toivonen. Sampling large databases for association rules. In *Proc. 1996 VLDB*, pp 134–145.

[22] D. Tsur, J. D. Ullman, S. Abitboul, C. Clifton, R. Motwani, and S. Nestorov. Query flocks: A generalization of association-rule mining. In *Proc. 1998 ACM-SIGMOD*, pp 1–12.

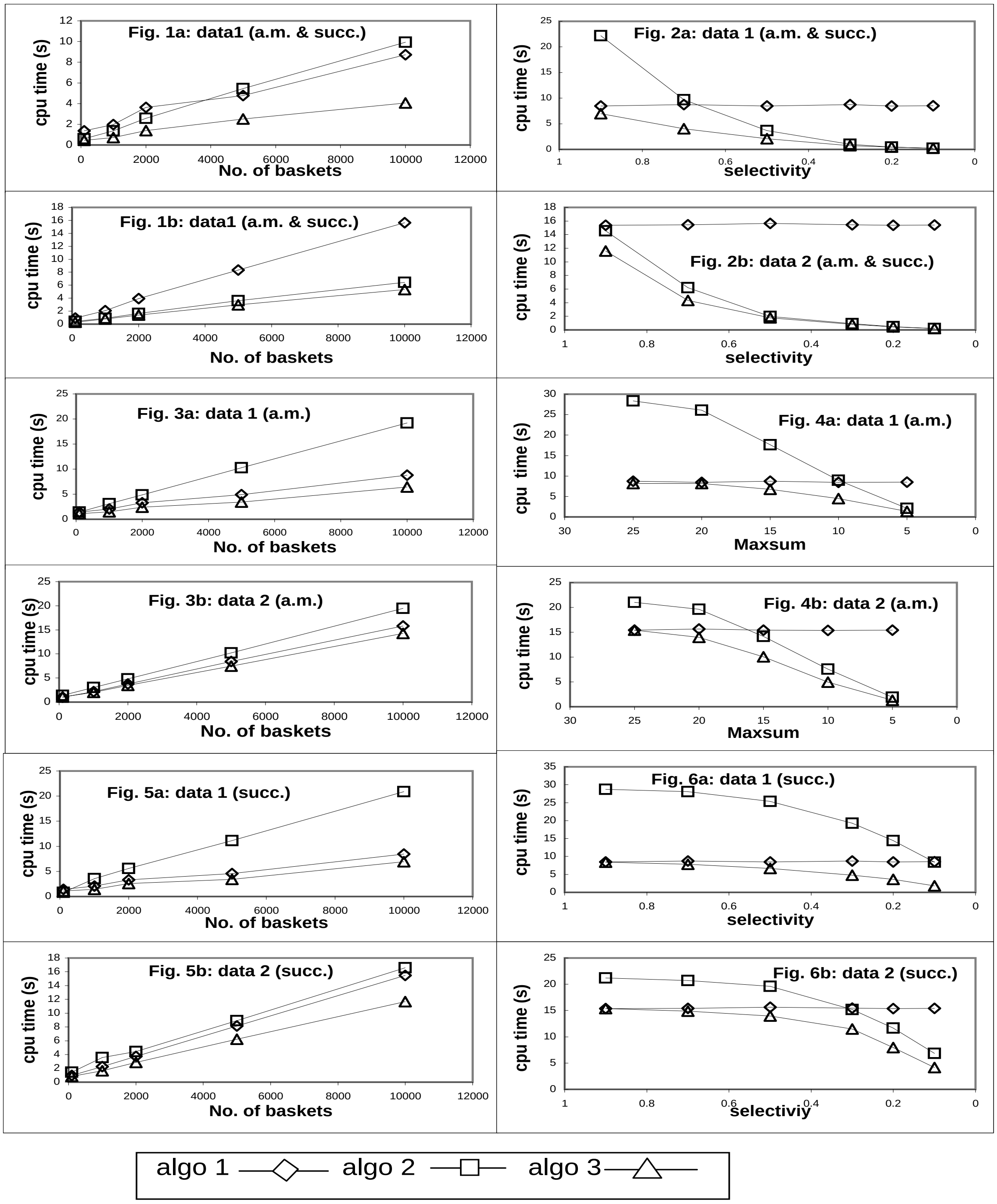

Figure B: Performance of Three Algorithms for Computing Constrained Correlation Queries