Optimizing Disjunctive Association Rules

Dmitry Zelenko

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana IL 61801, USA,
zelenko@cs.uiuc.edu

Abstract. We analyze several problems of *optimizing disjunctive association rules*. The problems have important applications in data mining, allowing users to focus at interesting rules extracted from databases. We consider association rules of the form $\bigwedge_{j=1}^{n}(A_{i_j} = v_j) \rightarrow C_2$, where $\{A_{i_1}, A_{i_2}, \ldots, A_{i_n}\}$ is a subset of the *categorical* attributes of the underlying relation R, and C_2 is any fixed condition defined over the attributes of the relation R. An instantiation of the rule binds the variables v_j's to values from the corresponding attribute domains. We study several problems, in which we seek a collection of instantiations of a given rule that satisfy certain optimality constraints. Each of the problems can re-interpreted as looking for one optimized disjunctive association rule. We exhibit efficient algorithms for solving the *optimized support* and *optimized confidence* problems, the *weighted support/confidence* problem, and the *shortest rule* problem. We discuss time and space complexity of the designed algorithms and show how they can be improved by allowing for approximate solutions.

1 Introduction

The last decade computer technology coupled with improved data acquisition techniques resulted into a significant increase in the rate of growth of large databases. The bulk of data accumulated in such databases encompasses every sphere of human life. Having eliminated the paper-based technology, the modern businesses are now recognizing the necessity to remove the burden of analyzing the data from their employees. The amount of the data accrued within any major company obviously precludes their manual treatment. Thus, a new generation of data processing techniques are called for. The techniques have to assist users with analysis of the data and extraction of useful knowledge. These techniques and tools are the subject of the field of *knowledge discovery in databases*(KDD) or *data mining*[3].

Knowledge extracted from databases can be represented in many possible ways. A representation that received much attention within the community of KDD researches is that of *association rules*[1, 11, 13].

An association rule X, in its most general form, is an implication $C_1 \rightarrow C_2$, where C_1, C_2 are conditions defined over the attributes of an underlying database

J.M. Żytkow and J. Rauch (Eds.): PKDD'99, LNAI 1704, pp. 204–213, 1999.

relation R. For example, for a relation $R(Student, CourseType, Grade)$, we can define the association rule X:

$$(CourseType = \texttt{elective}) \rightarrow (Grade = \mathbf{A})$$

The fraction of the relation tuples satisfying C_1 is termed the *support* of the rule (and denoted $sup(X)$). The ratio of the number of tuples satisfying the $C_1 \wedge C_2$ to the number of tuples satisfying C_1 is termed the *confidence* of the rule (and denoted $conf(X)$). For the above example, the support of the rule is the percentage of `elective` course tuples, and the confidence is the fraction of the `elective` tuples with the grade `A`.

Association rules have been introduced as an attempt to capture correlations and regularities in the underlying database relation, with hope that the regularities prove useful for future decision making. Data mining algorithms aim at extracting such useful association rules. Extracting association rules usually means finding all rules satisfying pre-specified lower bounds on their confidence and support. For example, we can ask a data mining system to produce all rules for the above relation R with support greater than 20% and confidence greater than 80%. If the rule X satisfies the support and confidence restrictions, then it should be output by the system. However, if we do not place additional constraints on the conditions C_1 and C_2 of an association rule, the number of feasible rules may be very large. Moreover, most of the rules satisfying the conditions may be useless. Therefore, we have introduce a *representational bias* to restrict the form of a rule.

The most popular bias in the literature requires that the condition C_1 be a conjunction of binary-valued attributes and C_2 be a single binary-valued attribute[1, 11]. Recently, Rastogi and Shim[12] introduced a new kind of association rules of the form

$$\bigwedge_{j=1}^{n} (A_{i_j} = v_j) \rightarrow C_2, \tag{1}$$

where $\{A_{i_1}, A_{i_2}, \ldots, A_{i_n}\}$ is a subset of the relation attributes, v_j is variable to be bound to a value d_j from the domain of A_{i_j}, and the consequent C_2 of a rule is any fixed condition defined over the attributes of the relation R. For this kind of rules, extracting association rules means finding a tuple of d_j's so that the rule instantiated with the d_j's satisfies the given minimum support and confidence conditions. A tuple of d_j's is called an *instantiation* of the rule. For example, consider the uninstantiated version of the above rule X:

$$(CourseType = v_1) \rightarrow (Grade = \texttt{A})$$

The goal is to find a value d_1(e.g., `elective`) for the variable v_1 so that the rule, instantiated with the value, will satisfy the minimum support and confidence conditions.

We can generalize the setting by allowing more than one instantiation for an association rule. More precisely, we seek a collection $\mathcal{I}$ of instantiations

($\mathcal{I} = \{I_1, \ldots, I_k\}$) so that the cumulative support and confidence[1] of the association rule are above the given minimum values. We can view the collection of instantiations as a *disjunctive association rule*, because the rule antecendent can be written as a disjunction of conjunctions, where each conjunction corresponds to a different instantiation of (1). In our example, we are looking for several course types that cover sufficient number of tuples in the relation R as well as have the grade `A` in the sufficient proportion of the covered tuples.

We study three optimization problems introduced in [12], and another natural optimization problem, *the shortest rule problem*. Note that each of the problems can be recast as seeking one *disjunctive association rule*.

1. *The optimized support problem*: given a relation R and a rule (1), find a set $\mathcal{I}$ of instantiations maximizing the cumulative support of the rule with its confidence greater than the given minimum confidence.
2. *The optimized confidence problem*: given a relation R and a rule (1), find a set $\mathcal{I}$ of instantiations maximizing the cumulative confidence of the rule with its support greater than the given minimum support.
3. *The weighted support/confidence problem*: given a relation R and a rule (1), and two positive integers α, β, find a set $\mathcal{I}$ of instantiations maximizing $\alpha sup + \beta conf$, where sup and $conf$ are the cumulative support and confidence of the instantiated rule (1), respectively.
4. *The shortest rule problem*: given a relation R and a rule (1), find a minimum set $\mathcal{I}$ of instantiations, so that the cumulative support and confidence of the rule are greater than the given minimum support and confidence, respectively.

2 Related Work

Association rules were first studied in [1]. The paper led to plethora of research aimed at enumerating conjunctive association rules defined over boolean attributes (e.g., [11, 14, 8, 13, 6]). Association rule enumeration is linked to finding minimal hypergraph transversals[7], learning monotone DNF from membership queries [7], and minimizing the number of failing subqueries[5]. The related computational hardness results are presented in [6, ?].

The problem of finding optimized association rules was introduced in [4]. The problem was defined for numeric attributes, and allowed at most two uninstantiated attributes to be present in the rule antecedent. Moreover, the algorithms presented in the paper produced a single instantiation of the rule; therefore, the rules were essentially conjunctive. Rastogi and Shim [12] generalized the setting by allowing more than two attributes and multiple instantiations, thereby introducing the notion of disjunctive association rules (they did not use the terminology though). They posed three of the problems that we study in this paper, and proved that the optimized support and optimized confidence problems are

[1] "Cumulative" means that we count tuples that satisfy at least one instantiation of the rule. We formally define the cumulative support and confidence in section 4

NP-hard. However, the proof implicitly relied on the unnatural complexity parameter for both of the problems. Namely, if M is the number of records in the relation R, then the problems are NP-hard, if $\log M$ (instead of M) is used as a complexity parameter for the problems. We take M to be the natural complexity parameter for all of the problems (that is, we allow to scan all of the relation), and design efficient algorithms for all of the above problems. Our algorithms run linear in n, in contrast to the algorithms in [12], which are exponential in n. However, [12] also considers rules with numeric attributes, and exhibits algorithms for finding optimized rules in the setting, while we deal only with categorical attributes.

3 The Summary of the Results

Let M be the number of records in the relation R, and N be the number of attributes. Also, given a rule (1) and a relation R, let m be the number of distinct tuples $(d_{i_1}, d_{i_2}, \ldots, d_{i_n}) \in \Pi_{A_{i_1}, A_{i_2}, \ldots, A_{i_n}}(R)$. Then we have the following results:

1. The optimized support problem can be solved in time $O(M(N + m(\log M + size(\delta))))$ and space $O(M(\log M + size(\delta)))$, where $size(\delta)$ is the number of bits used to encode the given minimum confidence value δ.
2. There is a fully polynomial time approximation scheme for the optimized confidence problem with time complexity $O(M(N + m \log \frac{1}{\epsilon}(\log M + \log \frac{1}{\epsilon})))$, where ϵ is the absolute error. The approximation scheme can be converted into an exact algorithm that takes $O(M(N + m \log^2 M))$ time and $O(M \log M)$ space.
3. The weighted support/confidence problem can be solved in time $O(M(N + m \log M))$ and space $O(M \log M)$.
4. The shortest rule problem can be solved in time $O(M(N + M(p+q)m \log m))$ and space $O(M m \log m)$, where $\frac{p}{q} = \delta$ is the given minimum confidence value.
5. By allowing to produce an approximate solution we can reduce both time and space complexity of all of the algorithms.

In the following sections, Z denotes the set of integers, Z_+ stands for the set of nonnegative integers, and Q denotes the set of rational numbers. Also, for a finite set S, $|S|$ is the number of elements in S.

4 Preliminaries

In this section we introduce the notation and definitions in order to reformulate all of the stated problems in terms of mathematical programming. The reformulation highlights the relationship between them and classical problems of combinatorial optimization.

Let R be a relation over the attributes $A_1, A_2, \ldots, A_N$ with M tuples. Each of the attributes is categorical, that is, there is no partial order imposed on the domain $dom(A_i)$ of an attribute A_i, for each $i = 1, 2, \ldots, N$. Let $\mathcal{A}$ be a subset of $\{A_1, A_2, \ldots, A_N\}$, $\mathcal{A} = \{A_{i_1}, A_{i_2}, \ldots, A_{i_n}\}$.

Definition 1. *An (uninstantiated)* **association rule** $X(v_1, v_2, \ldots, v_n)$ *over* $\mathcal{A}$ *is an implication*

$$\bigwedge_{j=1}^{n} (A_{i_j} = v_j) \rightarrow C_2,$$

where v_j *are variables, and* $C_2(A_1, A_2, \ldots, A_N)$ *is any fixed formula defined over* $A_1, A_2, \ldots, A_N$.

We also assume that it takes linear (in N) time to evaluate C_2 for a tuple of R.

Let X be an uninstantiated association rule over $\mathcal{A}$. Then we define the **set of all instantiations** $Inst$ of X as the projection $\Pi_{\mathcal{A}}(R)$ of R on A. Note that $|Inst| \leq |R| = M$.

We instantiate an association rule X by picking an instantiation of $Inst$ and binding the variables of X to the elements of the instantiation. More precisely, take $I = (d_1, d_2, \ldots, d_n) \in Inst$. Then, $X(I) = X(d_1, d_2, \ldots, d_n)$ is the association rule X instantiated with I.

Let $Inst = \{I_1, I_2, \ldots, I_m\}$. Denote $\bigwedge_{j=1}^{n}(A_{i_j} = v_j)$ by $U(A_1, A_2, \ldots, A_N; v_1, v_2, \ldots, v_n)$. For a fixed instantiation $I_i \in Inst$ let

$$s_i = |\{t \in R : U(t; I_i)\}| \text{ and } c_i = |\{t \in R : U(t; I_i) \wedge C_2(t)\}|$$

Intuitively, s_i is the count of the tuples contributing to the support of X instantiated with I_i(note that $\sum_{i=1}^{m} s_i = M$), while c_i is the count of tuples contributing to the confidence of X instantiated with I_i. Note that we can compute all c_i's and s_i's in one pass over the relation R, which takes $O(NM)$ time. This is the preprocessing time for all our algorithms.

Since all attributes are categorical, a tuple can contribute to the count of only one instantiation for a fixed association rule. Therefore, it is possible to express the cumulative support and confidence for a set of instantiations using the quantities of s_i and c_i. For a set $\mathcal{I} \subseteq Inst$ of instantiations, the **cumulative support** of $\mathcal{I}$ is defined as:

$$sup(\mathcal{I}) = \frac{\sum_{I_i \in \mathcal{I}} s_i}{M}$$

The **cumulative confidence** of $\mathcal{I}$ is defined similarly:

$$conf(\mathcal{I}) = \frac{\sum_{I_i \in \mathcal{I}} c_i}{\sum_{I_i \in \mathcal{I}} s_i}$$

Finally, we introduce a set 0-1 valued variables $x_i, i = 1, 2, \ldots, m$, and equate each subset $\mathcal{I}$ of $Inst$ with a boolean assignment to x_i's. That is,

$$x_i = \begin{cases} 1 \text{ if } I_i \in \mathcal{I} \\ 0 \text{ otherwise} \end{cases}$$

Now all four introduced problems problems can be reformulated in terms of 0-1 mathematical programming. In the following sections we state each of the problems as a 0-1 programming problem, and then give efficient algorithms for solving the problems

5 Optimizing Support

The optimized support problem for an association rule X, a relation R, and a minimum confidence value δ can be stated as follows:

$$\sum_{i=1}^{m} s_i x_i \rightarrow \max$$

$$\frac{\sum_{i=1}^{m} c_i x_i}{\sum_{i=1}^{m} s_i x_i} \geq \delta \tag{2}$$

$$s_i \geq c_i \geq 0,\ c_i, s_i \in Z,\ \delta \in Q \cap [0,1],\ x_i \in \{0,1\},\ i = 1,2,\ldots,m$$

Let the minimum confidence δ be represented as a rational number $\frac{p}{q}$, where $p, q \in Z_+$. Then, (2) can be rewritten as

$$\sum_{i=1}^{m} (ps_i - qc_i)x_i \leq 0$$

After denoting $a_i = ps_i - qc_i, i = 1,2,\ldots,m$, the optimized support problem becomes:

$$\sum_{i=1}^{m} s_i x_i \rightarrow \max$$

$$\sum_{i=1}^{m} a_i x_i \leq 0$$

$$s_i \geq 0,\ a_i, s_i \in Z,\ x_i \in \{0,1\},\ i = 1,2,\ldots,m$$

The above problem is a variant of KNAPSACK, a classical optimization problem that can be solved in less than mM steps ($M = \sum_{i=1}^{m} s_i$) by dynamic programming(see, for example,[2]). Usually the complexity of each step for KNAPSACK is treated as constant. In our case, however, the complexity depends on the value of the minimum confidence parameter for the problem. Since the time complexity of one step is roughly equal to the time of writing down the value of $\sum_{i=1}^{m} a_i x_i$, we can bound it by bounding the number of bits used to encode $\sum_{i=1}^{m} a_i x_i$ for any $(x_1, x_2, \ldots, x_m) \in \{0,1\}^m$:

$$\log(|\sum_{i=1}^{m} a_i x_i|) \leq \log(\sum_{i=1}^{m} |ps_i - qc_i|) \leq \log(p+q) + \log(\sum_{i=1}^{m} s_i) \leq size(\delta) + \log M,$$

where $size(\delta) = \log p + \log q$ is the number of bits used to encode δ. Thus, the classical dynamic programming algorithm solves the optimized support problem in $O(mM(\log M + size(\delta)))$. Adding the preprocessing time of $O(NM)$ gives $O(M(N + m(\log M + size(\delta))))$.

6 Optimizing Confidence

The optimized confidence problem for an association rule X, a relation R, a minimum support $\sigma = \frac{B}{M},\ B \in Z_+$, can be stated as follows:

$$\frac{\sum_{i=1}^{m} c_i x_i}{\sum_{i=1}^{m} s_i x_i} \to \max \tag{3}$$

$$\sum_{i=1}^{m} s_i x_i \geq B$$

$$s_i \geq c_i \geq 0,\ c_i, s_i \in Z,\ B \in Z_+,\ x_i \in \{0,1\},\ i = 1, 2, \ldots, m$$

Consider the decision problem corresponding to the optimized confidence problem:

$$\sum_{i=1}^{m} s_i x_i \geq B,\ \ \frac{\sum_{i=1}^{m} c_i x_i}{\sum_{i=1}^{m} s_i x_i} \geq \delta$$

$$s_i \geq c_i \geq 0,\ c_i, s_i \in Z,\ B \in Z_+,\ \delta \in Q \cap [0,1],\ x_i \in \{0,1\},\ i = 1, 2, \ldots, m$$

We can treat the problem as a decision version of the optimized support problem and solve it by dynamic programming, as we described in the previous section. We consider this procedure as an oracle $DecConf(\delta)$ returning $\emptyset$, if the decision problem does not have a solution, and returning a solution x otherwise.

Then, given the bound ϵ on the absolute error of the required solution, we start with an initial interval $[0,1]$. The interval is repeatedly halved in a way that guarantees that the optimal confidence value lies within the halved interval. When the interval becomes less than ϵ we can be sure that any feasible solution lying within the interval has the absolute error less than ϵ. Thus, the $DecConf(\delta)$ returns an approximate solution x to the optimized confidence problem with the absolute error less than ϵ.

The number of iterations in the algorithm is $\log \frac{1}{\epsilon}$. The maximum size of δ is also $O(\log \frac{1}{\epsilon})$. Therefore, the running time of the algorithm is $O(mM \log \frac{1}{\epsilon}(\log M + \log \frac{1}{\epsilon}))$. Adding the preprocessing time of $O(NM)$ gives $O(M(N + m \log \frac{1}{\epsilon}(\log M + \log \frac{1}{\epsilon})))$. Since the running time of the algorithm is polynomial in all complexity parameters (including ϵ), we have designed a *fully polynomial approximation scheme [9] for the optimized confidence problem.*

Observe that we can convert the approximation scheme into an exact algorithm for the optimized confidence problem by choosing the absolute error of ϵ to be $\frac{\gamma}{(\sum_{i=1}^{m} s_i)^2}$, where $\gamma < 1$. Then, $\log \frac{1}{\epsilon} \leq 2 \log M - \log \gamma$, and the running time of the algorithm is $O(M(N + m \log^2 M))$.

7 Optimizing Weighted Support/Confidence

The weighted support/confidence problem for an association rule X, a relation R, and weights $\alpha, \beta \in Z_+$, can be stated as follows:

$$\alpha \sum_{i=1}^{m} s_i x_i + \beta \frac{\sum_{i=1}^{m} c_i x_i}{\sum_{i=1}^{m} s_i x_i} \to \max \tag{4}$$

$$s_i \geq c_i \geq 0,\ c_i, s_i, \alpha, \beta \in Z_+,\ x_i \in \{0,1\},\ i = 1, 2, \ldots, m$$

We can solve the problem by considering the following set of optimization problems:

$$\sum_{i=1}^{m} s_i x_i = B$$

$$\alpha B + \frac{\beta}{B} \sum_{i=1}^{m} c_i x_i \rightarrow \max$$

$$s_i \geq c_i \geq 0,\ c_i, s_i, \alpha, \beta \in Z_+,\ x_i \in \{0,1\},\ i = 1, 2, \ldots, m$$

Each of the problems corresponds to a different value of B, where $B \in Z_+,\ 0 < B \leq M$. In order to produce an optimal solution to the weighted support/confidence problem, we take the best solution from the M solutions to the above M problems. Each of the problems is again a restricted variant of KNAPSACK, and it can be solved by dynamic programming in time $O(mB \log M)$. Therefore, the naive algorithm that solves all M problems separately takes $O(mM^2 \log M)$ time.

We can reduce the running time by the factor of M by solving all of the B problems at the same time. More precisely, let $W(k, B)$ be the maximum value of $\sum_{i=1}^{m} c_i x_i$ produced using the variables $x_1, x_2, \ldots, x_k$ (i.e., $x_{k+1} = \cdots = x_m = 0$), such that $\sum_{i=1}^{k} s_i x_i = B$. If for all $(x_1, \ldots, x_k) \in \{0,1\}^k$, $\sum_{i=1}^{k} s_i x_i \neq B$, then we set $W(k, B) = -\infty$. Now we can specify the recurrence relation for $W(k, B)$ as follows:

$$W(k, 0) = 0,\ 0 \leq k \leq n,\ \ W(0, B) = -\infty,\ 0 < B \leq M$$

$$W(k, B) = \max(W(k-1, B - s_k) + c_k, W(k-1, B)),\ 1 \leq k \leq n,\ 0 < B \leq M$$

We can fill the table for $W(k, B)$ in $O(mM)$ steps, where each step takes at most $O(\log M)$ time (since $|\sum_{i=1}^{m} c_i x_i| \leq M$). An optimal solution to the weighted support/confidence problem corresponds to the maximum value among $\alpha B + \frac{\beta}{B} W(n, B),\ 0 < B \leq M$. We can find the value in M steps and then find the solution in less than m steps by using the above recurrence relation and going backwards from the cell of the table corresponding to the maximum value. Hence, the total running time of the algorithm is $O(mM \log M)$. Adding the preprocessing time of $O(NM)$ gives $O(M(N + m \log M))$.

8 Optimizing the Rule Length

The shortest rule problem for an association rule X, a relation R, a minimum support $\sigma = \frac{B}{M},\ B \in Z_+$, and a minimum confidence $\delta = \frac{p}{q},\ p, q \in Z_+$ can be stated as follows:

$$\sum_{i=1}^{m} x_i \rightarrow \min$$

$$\frac{\sum_{i=1}^{m} c_i x_i}{\sum_{i=1}^{m} s_i x_i} \geq \frac{p}{q},\ \ \sum_{i=1}^{m} s_i x_i \geq B$$

$$s_i \geq c_i \geq 0,\ c_i, s_i \in Z,\ B \in Z_+,\ x_i \in \{0,1\},\ i = 1, 2, \ldots, m$$

which is equivalent to

$$\sum_{i=1}^{m} x_i \to \min$$

$$\sum_{i=1}^{m} a_i x_i \geq 0,\ \ \sum_{i=1}^{m} s_i x_i \geq B \tag{5}$$

$$s_i \geq 0,\ a_i, s_i \in Z,\ B \in Z_+,\ x_i \in \{0,1\},\ i = 1, 2, \ldots, m$$

where $a_i = qc_i - ps_i, i = 1, 2, \ldots, m$. Recall that $|\sum_{i=1}^{m} a_i x_i| \leq (p+q)M$.

We again use ubiquitous dynamic programming to solve (5). Denote $C = (p+q)M$. Let $L(k, D, A)$ be the minimum number of nonzero variables in $(x_1, \ldots, x_k) \in \{0,1\}^k$, so that $\sum_{i=1}^{k} s_i x_i \geq D$ and $\sum_{i=1}^{k} a_i x_i \geq A$, where $0 \leq k \leq m$, $0 \leq D \leq B$ and $|A| \leq C$. Again, we set $L(k, D, A) = \infty$, if for all $(x_1, \ldots, x_k) \in \{0,1\}^k$, either $\sum_{i=1}^{k} s_i x_i \geq D$ or $\sum_{i=1}^{k} a_i x_i \geq A$ is not satisfied. Then,

$$L(k, 0, A) = 0,\ 0 \leq k \leq n,\ -C \leq A \leq 0$$

$$L(k, 0, A) = \infty,\ 0 < A \leq C$$

$$L(0, D, A) = \infty,\ 0 < D \leq B,\ |A| \leq C$$

$$L(k, D, A) = \min(L(k-1, D-s_k, A-a_k)+1, L(k-1, D, A)),\ 1 \leq k \leq n,\ |A| \leq C$$

We fill the 3-dimensional table for L in $O(m(p+q)M^2)$ where each step takes $O(\log m)$ time. Hence, the time complexity of filling in the table is $O(m \log m(p+q)M^2)$. Then, $L(m, B, 0)$ is the minimum length of a rule satisfying the support and confidence constraints. We can find the actual optimal solution (rule) in $O(m)$ steps by backtracking from $L(m, B, 0)$. The total time complexity of the shortest rule problem (including the preprocessing time) is $O(M(N + M(p+q)m \log m))$.

9 Discussion

We used dynamic programming to solve all of the above problems, Since a dynamic programming algorithm only needs to know the values in the previous row of the table, while filling in the current row, the memory complexity of the algorithm is the size of one table row. For the first three problems, the size is $O(M \log M)$ (M cells and $O(\log M)$ bits for a cell value). For the shortest rule problem, the size of the row is $O(mM \log m)$ (mM cells and $O(\log m)$ bits for a cell value).

We can reduce the time and space complexity of the above algorithms by allowing for suboptimal solutions. There is a number of standard approximation schemes for KNAPSACK[10], and any of them can be used to find approximate solutions for the above problems. One of the most common methods is to scale down each of objective function coefficients by a factor k. This will lead to both

running time and space that the algorithms take being reduced by the factor of k. The produced solution, however, will be suboptimal with the absolute error less than kL, where L is the length (number of instantiations) of the optimal rule.

10 Conclusions and Further Research

We analyzed several problems for optimizing association rules. The problems have important application in data mining, allowing users to focus at interesting rules extracted from databases. We exhibited efficient algorithms for solving all of the problems.

For further research, we plan to apply the algorithms to finding association rules in real world databases. We also intend to study and try to develop efficient algorithms for optimizing other variants of association rules.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. *SIGMOD*, 22(2), 1993.
2. G. B. Danzig. *Linear Programming and Extensions.* Princeton University Press, 1963.
3. U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Anvances in Knowledge Discovery and Data Mining.* AAAI Press / The MIT Press, Menlo Park, California, 1996.
4. T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Mining optimized association rules for numeric attributes. In *PODS*, 1996.
5. P. Godfrey. Minimization in cooperative response to failing database queries. Technical Report CS-TR-3348, University of Maryland, College Park, 1994.
6. D. Gunopoulos, H. Mannila, and S. Saluja. Discovering all most specific sentences by randomized algorithms. In *ICDT*, 1997.
7. D. Gunopulos, H. Mannila, R. Khardon, and H. Toivonen. Data mining, hypergraph transversals, and machine learning (extended abstract). In *PODS*, 1997.
8. J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *VLDB*, 1995.
9. D. S. Hochbaum, editor. *Approximation Algorithms for NP-hard problems.* PWS Publishing Company, 1997.
10. E. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, 4, 1979.
11. H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. In *KDD*, 1994.
12. R. Rastogi and K. Shim. Mining optimized association rules for categorical and numeric attributes. In *ICDE*, 1998.
13. R.Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *ICMD*, 1996.
14. A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *VLDB*, 1995.