ELSEVIER

Information Processing Letters 71 (1999) 199–204

Information Processing Letters

www.elsevier.com/locate/ipl

A fast algorithm for building lattices

Lhouari Nourine *, Olivier Raynaud [1]

Département d'Informatique Fondamentale, LIRMM, Université Montpellier II, CNRS UMR C09928, 161 rue Ada, 34392 Montpellier Cedex 5, France

Received 12 May 1999; received in revised form 1 September 1999
Communicated by L. Boasson

Abstract

This paper presents a simple, efficient algorithm to compute the *covering graph* of the lattice generated by a family $\mathcal{B}$ of subsets of a set X. The implementation of this algorithm has $O((|X| + |\mathcal{B}|) \cdot |\mathcal{B}|)$ time complexity per lattice element. This improves previous algorithms of Bordat (1986), Ganter and Kuznetsov (1998) and Jard et al. (1994). This algorithm can be used to compute the Galois (concept) lattice, the maximal antichains lattice or the Dedekind–MacNeille completion of a partial order, without increasing time complexity.

Keywords: Family of sets; Lattice; Data structure; Algorithms

1. Preliminaries

Lattices are mathematical structures which are used for many applications in computer science [1,4,8–11, 15,17]. For example, Wille's group in Darmstadt [17, 18] has built a theory, and software, for knowledge representation using the concept lattice (also known as the Galois lattice). Concept lattices are also used for data mining (i.e., discovering knowledge in databases) [19].

Many authors have considered the construction problem for the concept lattice, the Dedekind–MacNeille completion and the maximal antichain lattice of a partially ordered set [2,3,7,14,16].

In this paper, we will reformulate the problem of building these lattices as a more general problem. We propose an efficient algorithm which improves algorithms in Ganter [7] for building the Dedekind–MacNeille completion and that of Jard et al. [14] for the maximal antichains lattice. More generally, the proposed algorithm can be used whenever we have a closure operator.

To do this, we assume that the reader is familiar with standard definitions for partially ordered sets. For definitions and proofs not given here, we refer to [5].

Let X be a set. A *basis* $\mathcal{B}$ is a set of subsets of X. We denote by $\mathcal{F}_{\mathcal{B}}$ the family generated by union from the basis $\mathcal{B}$, i.e.,

$$\mathcal{F}_{\mathcal{B}} = \left\{ \bigcup_{B \in I} B \mid I \subseteq \mathcal{B} \right\}.$$

We also say that $\mathcal{B}$ is a generator of $\mathcal{F}_{\mathcal{B}}$. The family $\mathcal{F}_{\mathcal{B}}$ is said to be *closed under union*, and it is well known that when ordered by inclusion $\mathcal{F}_{\mathcal{B}}$ is a sup-sublattice of the powerset lattice 2^X and thus a complete lattice. When $\mathcal{B}$ is understood, we use $\mathcal{F}$ instead of $\mathcal{F}_{\mathcal{B}}$ for simplicity.

The family $\mathcal{F} = \{\{\}, \{134\}, \{14\}, \{234\}, \{1234\}, \{25\}, \{12345\}, \{1245\}, \{2345\}\}$ generated by the ba-

* Corresponding author. Email: nourine@lirmm.fr.
[1] Email: raynaud@lirmm.fr.

0020-0190/99/$ – see front matter
PII: S0020-0190(99)00108-8

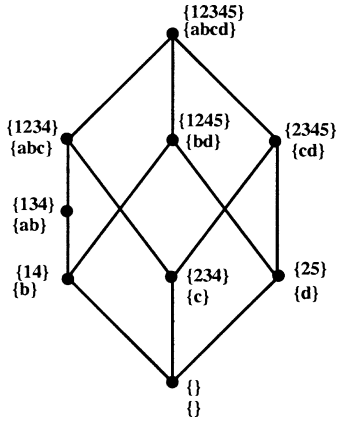

Fig. 1.

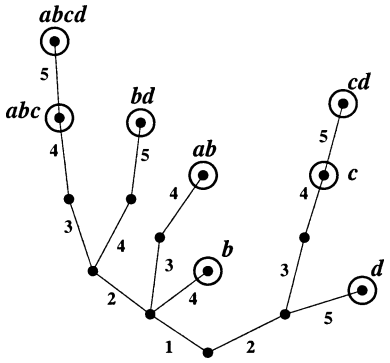

Fig. 2. The nodes with circle correspond to elements of $\mathcal{F}$, where the sequence $a_1a_2..a_k$ are elements of $\mathcal{B}$ with $\bigcup_{i=1}^{k} a_i$ is the set of labels on the path to the root.

sis $\mathcal{B} = \{a = \{134\}, b = \{14\}, c = \{234\}, d = \{25\}\}$ is shown in Fig. 1.

We consider the following problem:

Building Lattice Problem. Given a basis $\mathcal{B}$, compute the covering graph $G = (\mathcal{F}, \subseteq)$ of the family $\mathcal{F}_{\mathcal{B}}$ generated by $\mathcal{B}$, when ordered by inclusion.

Remark 1. In practice, a basis is often given by a bipartite graph $G = (\mathcal{B}, X, E)$ where $[B, x] \in E$, $B \in \mathcal{B}$, $x \in X$ iff $x \in B$.

We present an algorithm with $O((|X| + |\mathcal{B}|) \cdot |\mathcal{B}| \cdot |\mathcal{F}|)$ time complexity to compute the covering graph of $(\mathcal{F}, \subseteq)$. Our implementation generalizes and improves algorithms of Chein [3], Norris [16], Ganter and Kuznetsov [7], Bordat [2], and Jard et al. [14]. For more details on the improvement see Section 4.

Our algorithm uses a two-step approach:

(1) Generate the family $\mathcal{F}$ represented in a lexicographic tree.
(2) Compute the covering relations of elements of $\mathcal{F}$ using Theorem 2.

This paper is structured in four sections. Section 2 deals with the family generation and the lexicographic tree. Section 3 gives the covering properties of lattice elements. In Section 4 we discuss the connection between the family $\mathcal{F}$ and the concept lattice of a binary relation, maximal antichains lattice and Dedekind–MacNeille completion of an ordered set. We also give here the comparison of our algorithm with related work.

2. A naïve algorithm to generate $\mathcal{F}_{\mathcal{B}}$ from $\mathcal{B}$

The following naïve algorithm computes the lexicographic tree $T_{\mathcal{F}}$ representing the family $\mathcal{F}$ generated from $\mathcal{B}$. The lexicographic tree was described by Habib and Nourine [13,12].

Let $\mathcal{B}$ be a basis and σ a total ordering of X. Each element of $\mathcal{F}_{\mathcal{B}}$ is represented by a pair $(F, \gamma(F))$ where $\gamma(F) = \{B \in \mathcal{B} \mid B \subset F\}$.

For the purposes of describing the algorithm it is useful to view each set $F \in 2^X$ as a word on the alphabet X, with the symbols in increasing order. Clearly there is a bijection $F = \{a_1, a_2, \ldots, a_k\} \to a_1a_2 \ldots a_k$ with $a_1 < a_2 < \cdots < a_k$. We can therefore, abuse notation and speak of a set $a_1a_2 \ldots a_k$.

Fig. 2 shows the lexicographic tree corresponding to the family in Fig. 1.

Let us now show how to build the lexicographic tree from $\mathcal{B} = \{B_1, B_2, \ldots, B_m\}$. We denote by $\mathcal{F}^i$ the union-closed family generated by $\mathcal{B}^i = \{B_1, B_2, \ldots, B_i\}$.

- The root corresponds to the empty set, i.e., $\mathcal{F}^0$.
- At step i, we compute the union-closed family $\mathcal{F}^i$ from $\mathcal{F}^{i-1}$, by setting

$$\mathcal{F}^i = \mathcal{F}^{i-1} \cup \left\{F \cup B_i \mid F \in \mathcal{F}^{i-1}\right\}.$$

Clearly $\mathcal{F} = \mathcal{F}^m$.

Algorithm 1. Tree($\mathcal{B}$).

Data: A basis $\mathcal{B}$.
Result: The lexicographic tree T of $\mathcal{F}$.

```
begin
    Let F = {∅}; {the root of T}
    for each B ∈ B do
        for F ∈ F do
1           F' = F ∪ B;
2           if F' ∉ F then
                F = F ∪ F;
3           γ(F') = γ(F) ∪ {B};
end
```

Theorem 1. *Algorithm* 1 *computes the lexicographic tree of the family* $\mathcal{F}$ *generated by* $\mathcal{B}$ *in* $O((|X| + |\mathcal{B}|) \cdot |\mathcal{B}| \cdot |\mathcal{F}|)$ *steps, with space in* $O((|X| + |\mathcal{B}|) \cdot |\mathcal{F}|)$.

Proof. Clearly lines 1 and 3 can be done in $O(|X| + |\mathcal{B}|)$. Line 2 checks if F' is already in the tree and possibly inserts F'. This can be implemented in $O(|X|)$, since the children of any node are sorted according to σ. So the total complexity of the internal **for** is $O((|X| + |\mathcal{B}|) \cdot |\mathcal{F}|)$.

Since the internal **for** is repeated $|\mathcal{B}|$ times, we obtain the announced result. □

Now let us show how this tree can be used to compute the covering graph of the lattice $(\mathcal{F}, \subseteq)$, without increasing the time complexity.

3. Computing the covering graph

First let us state the covering theorem for $(\mathcal{F}, \subseteq)$. Let F and $F' \in (\mathcal{F}, \subseteq)$ be such that $F \subseteq F'$. We define $\Delta(F', F) = \gamma(F') \setminus \gamma(F)$. We denote by $\prec$ the covering relation.

Theorem 2. *Let* F *and* $F' \in \mathcal{F}$ *with* $F \subset F'$. *Then*

$$F \prec F' \quad \text{iff} \quad \text{for all } B_1, B_2 \in \Delta(F', F) \quad B_1 \setminus F = B_2 \setminus F.$$

Proof. Let $F, F' \in \mathcal{F}$ with $F \subset F'$. Then F' could be written as

$$F' = F \cup \{B \mid B \in \Delta(F', F)\}.$$

(1) Assume that F is covered by F'. Let us show that for all B_1 and B_2 in $\Delta(F', F)$ we have $B_1 \setminus F = B_2 \setminus F$.

Suppose $B_1 \setminus F \not\subseteq B_2 \setminus F$. Thus $F'' = F \cup B_1 \subset F' = F \cup \{B \mid B \in \Delta(F', F)\}$. Therefore $F \subset F'' \subset F'$ is a contradiction with F is covered by F', thus $B_1 \setminus F \subseteq B_2 \setminus F$. Similarly we show that $B_1 \setminus F \supseteq B_2 \setminus F$.

(2) Conversely, suppose that for all $B_1, B_2 \in \Delta(F', F)$, $B_1 \setminus F = B_2 \setminus F$.

Let $F'' \in \mathcal{F}$ such that $F \subset F'' \subset F'$. Clearly $\gamma(F) \subset \gamma(F'') \subset \gamma(F')$ and therefore $F'' = F \cup \{B \mid B \in \gamma(F'') \setminus \gamma(F)\} = F'$ since $\gamma(F'') \setminus \gamma(F) \subseteq \gamma(F') \setminus \gamma(F) = \Delta(F', F)$. □

The following corollary is a direct consequence of Theorem 2.

Corollary 1. *Let* F *and* $F' \in \mathcal{F}$ *with* $F \subseteq F'$. *Then*

$$F \prec F' \quad \text{iff} \quad F' = F \cup B \quad \text{for all } B \in \Delta(F, F').$$

Let us now describe how to compute the covering graph of $(\mathcal{F}, \subseteq)$.

We consider the family $\mathcal{F}_\mathcal{B}$ generated by a basis $\mathcal{B}$ using Algorithm 1. The strategy of our algorithm is to compute the set of covering elements, denoted by $ImSucc(F)$ of each F in the family $\mathcal{F}_\mathcal{B}$. Clearly $F' \in \mathcal{F}_\mathcal{B}$ is a candidate if $F \subset F'$ and F' can be computed as $F \cup B$ for some B in $\mathcal{B} \setminus \gamma(F)$. Let us denote by $S = \{F \cup B \mid B \in \mathcal{B} \setminus \gamma(F)\}$ the multi-set of candidates for covering F (S can have redundant sets).

The algorithm explores the set S and decides that $F' \in S$ is a covering of F if and only if F' is found $|\Delta(F', F)|$ times in S. To do so, we compute the set S and maintain the number of occurrences of each element F' in S in a counter $COUNT(F')$.

Then, for each element $F' \in S$ we check if the cardinal of $\Delta(F', F)$ is equal to $COUNT(F')$; if so then F' covers F, by Corollary 1.

Now, given a lexicographic tree of a family $\mathcal{F}_\mathcal{B}$ the following algorithm will build the covering graph of $(\mathcal{F}_\mathcal{B}, \subseteq)$.

Algorithm 2. Covering-Graph($\mathcal{F}_B$).

Data: The lexicographic tree of $\mathcal{F}_B$, and $\gamma(F)$ for each $F \in \mathcal{F}_B$.
Result: The Adjacency lists *ImSucc* of the covering graph of the lattice $(\mathcal{F}_B, \subseteq)$

```
begin
    Initialize COUNT(F) to 0 for any F ∈ F_B;
    for each F ∈ F_B do
        for each B ∈ B \ γ(F) do
1           F' = F ∪ B;
            COUNT(F') + +;
2           if |γ(F')| = COUNT(F') + |γ(F)|
                then
                ImSucc(F) = ImSucc(F) ∪ {F'}
        Reset COUNT.
end
```

Algorithm 3. UCS($\mathcal{B}$).

Data: A basis $\mathcal{B}$ of size n.
Result: The Adjacency lists *ImSucc* of the covering graph of the lattice $(\mathcal{F}_B, \subseteq)$

```
begin
    F_B = Tree(B);
    Covering-Graph(F_B);
end
```

Theorem 3. *Algorithm* 2 *computes the adjacency lists of the covering graph for the lattice* $(\mathcal{F}_B, \subseteq)$ *in* $O((|X| + |\mathcal{B}|) \cdot |\mathcal{B}| \cdot |\mathcal{F}_\mathcal{B}|)$ *steps.*

Proof. Clearly Algorithm 2 computes the covering graph of the lattice $(\mathcal{F}_B, \subseteq)$ by Corollary 1.

In line 2, we need to compute $|\gamma(F')|$ and $|\gamma(F)|$, which can be done in time $O(|X|+|\mathcal{B}|)$, using a search in the tree. So the total complexity of the internal **for** is $O((|X| + |\mathcal{B}|) \cdot |\mathcal{B}|)$.

Resetting *COUNT* to all elements computed by line 1; this can be done in $O(|\mathcal{B}|)$ by keeping them in a linked list.

It is clear now that the total complexity of our algorithm is $O((|X| + |\mathcal{B}|) \cdot |\mathcal{B}| \cdot |\mathcal{F}_\mathcal{B}|)$. □

4. Applications

We will show in this section the use of our algorithm for computing the covering graph for several lattices. To do so, we define, for an ordered set $P = (X, \leqslant)$, $A^u = \{x \in X \mid a \leqslant x, a \in A\}$, $A^l = \{x \in X \mid a \geqslant x, a \in A\}$ and $\downarrow x = \{y \in X \mid y \leqslant x\}$.

4.1. Lattice of ideals of a poset

Let $P = (X, \leqslant)$ be a poset. Let I be a subset of X, I is an ideal of P if $y \in I$ and $x \leqslant y$ implies $x \in I$. We denote by $I(P)$ the set of ideals of P. The set of ideals of a poset when ordered by inclusion forms a distributive lattice (lattice of ideals of P) denoted by $I(P) = (I(P), \leqslant)$.

Corollary 2. *Let* $P = (X, <)$ *be a poset and* $\mathcal{B} = \{\downarrow x, x \in X\}$ *a basis. Then* $I(P)$ *is isomorphic to* $(\mathcal{F}_\mathcal{B}, \subseteq)$.

Proposition 1. *Algorithm* 3 *computes the adjacency lists of the covering graph for the lattice of ideals* $I(P) = (\mathcal{F}_B, \subseteq)$ *of* $P = (X, \leqslant)$ *in time* $O(|X|^2 \cdot |\mathcal{F}_\mathcal{B}|)$.

Proof. Clearly $|\mathcal{B}| = |X|$. □

The algorithm in [12] has linear time complexity in the size of the lattice $I(P)$. This is due to regularities of the tree and the lattice of ideals which is distributive.

4.2. Dedekind–MacNeille completion

Let $P = (X, \leqslant)$ be a poset. A *cut* of P is a pair (A, B) with $A, B \subseteq X$ with $A^u = B$ and $A = B^l$. It is well known that the cuts, ordered by

$$(A_1, B_1) \leqslant (A_2, B_2) :\Leftrightarrow A_1 \subseteq A_2 \text{ or } (B_1 \supseteq B_2)$$

form a complete lattice, the Dedekind–MacNeille completion of P, denoted by $DM(P)$ [5]. It is the smallest complete lattice containing P as a suborder.

Corollary 3. *Let* $P = (X, <)$ *be a poset and* $\mathcal{B} = \{X \setminus \downarrow x \mid x \in X\}$ *a basis. Then* $DM(P)$ *is isomorphic to* $(\mathcal{F}_\mathcal{B}, \supseteq)$.

Proposition 2. *Algorithm* 3 *computes the adjacency lists of the covering graph for the Dedekind–MacNeille completion* $DM(P) = (\mathcal{F}_B, \supseteq)$ *of* $P = (X, \leqslant)$ *in* $O(|X|^2 \cdot |\mathcal{F}_\mathcal{B}|)$.

Proof. Clearly $|\mathcal{B}| = |X|$. □

The algorithm in [7] has time complexity $O(|X|^3 \cdot |\mathcal{F}_\mathcal{B}|)$.

4.3. Lattice of maximal antichains

Let $P = (X, \leqslant)$ be a poset. We denote by $AM(P)$ the set of maximal antichains of P. The lattice of maximal antichains of P, $AM(P) = (AM(P), \leqslant)$ is defined by

$$A \leqslant B \Leftrightarrow A^l \subseteq B^l \quad \text{for } A, B \in AM(P).$$

Corollary 4. *Let $P = (X, <)$ be a poset and $\mathcal{B} = \{\downarrow x \setminus \{x\} \mid x \in X\}$ a basis. Then $AM(P)$ is isomorphic to $(\mathcal{F}_\mathcal{B}, \supseteq)$.*

Proposition 3. *Algorithm* 3 *computes the adjacency lists of the covering graph for the lattice of maximal antichains $AM(P) = (\mathcal{F}_B, \supseteq)$ of $P = (X, \leqslant)$ in $O(|X|^2 \cdot |\mathcal{F}_\mathcal{B}|)$.*

Proof. Clearly $|\mathcal{B}| = |X|$. □

The algorithm in [14] has time complexity $O(|X|^3 \cdot |\mathcal{F}_\mathcal{B}|)$.

4.4. Galois lattice

Let $R = (J, M, E)$ be a binary relation. A concept of R is a pair (A, B) with $A \subseteq J$ and $B \subseteq M$ with $A^u = B$ and $A = B^l$. It is well known that the concepts, ordered by

$$(A_1, B_1) \leqslant (A_2, B_2) :\Leftrightarrow A_1 \subseteq A_2 \text{ or } (B_1 \supseteq B_2)$$

form a complete lattice, called the Galois lattice of R and denoted by $Gal(R)$.

Corollary 5. *Let $R = (J, M, E)$ be a binary relation and $\mathcal{B} = \{J \setminus \downarrow m \mid m \in M\}$ a basis. Then $Gal(R)$ is isomorphic to $(\mathcal{F}_\mathcal{B}, \subseteq)$.*

Theorem 4. *Algorithm* 3 *computes the adjacency lists of the covering graph for the lattice $Gal(R) = (\mathcal{F}_\mathcal{B}, \subseteq)$ in $O((|J| + |M|) \cdot |M| \cdot |\mathcal{F}|)$ time complexity.*

The algorithm in [2] has time $O(|J| \cdot |E| \cdot |\mathcal{F}_\mathcal{B}|)$.

5. Conclusion

Our method leads to algorithms whose complexity is lower than that of those in [2,7,14]. One open question is the enumeration of the family generated by a basis (without computing the tree or the lattice) with the same complexity. A fast enumeration algorithm uses $O(|X|^3)$ per element [6].

Acknowledgment

The authors wish to thank Geňa Hahn from University of Montréal, for his suggestions which have greatly improved the presentation of the paper.

Appendix A

A lexicographic tree $T_\mathcal{F}$ is an arborescence $(V(T_\mathcal{F}), A(T_\mathcal{F}))$ with root r, labeled by

$$a : A(T_\mathcal{F}) \to X \quad \text{and} \quad b : V(T_\mathcal{F}) \to X^* \times 2^\mathcal{B},$$

where X^* is the set of finite words on the alphabet X satisfying

(1) $b(r) = (\varepsilon, \emptyset)$ corresponds to the empty set.
(2) If $a((x, y)) = c$ then $a((y, z)) > c$ for each child z of y. Also, $a((y, z)) \neq a((y, z'))$ for all children $z \neq z'$ of y. This gives a natural order on the set of children of y.
(3) $b(x) = ((P_x), \gamma((P_x)))$ where P_x is the unique path from the root r to x and where (P_x) is obtained by concatening in order, the labels of the arcs on P_x. Clearly by (2), (P_x) is a word whose symbols occur in increasing order.
(4) The nodes x such that $(P_x) \in \mathcal{F}$ are marked.

For our purpose, we combine (3) and (4) to

(3′) $b(x) = \emptyset$ if $(P_x) \notin \mathcal{F}$ otherwise $b(x) = \gamma((P_x))$.

References

[1] H. Aït-Kaci, R. Boyer, P. Lincoln, R. Nasr, Efficient implementation of lattice operations, ACM Trans. Program. Languages Systems 11 (1) (1989) 115–146.
[2] J.P. Bordat, Calcul pratique du treillis de Galois d'une correspondance, Math. Sci. Hum. 96 (1986) 31–47.
[3] M. Chein, Algorithme de recherche de sous-matrice première d'une matrice, Bull. Math. R. S. Roumanie 13 (1969).
[4] V. Dahl, A. Fall, Logical encoding of conceptual graph lattice, in: G.W. Mineau, B. Moulin, J.F. Sowa (Eds.), Proc. 1st Internat. Conference on Conceptual Structures, August 4–7, Université Laval, Quebec, 1993.

[5] B.A. Davey, H.A. Priestley, Introduction to Lattices and Orders, 2nd edn., Cambridge University Press, Cambridge, 1991.
[6] B. Ganter, Two basic algorithms in concept analysis, Technical Report, No. 831, Technische Hochschule Darmstadt, 1984.
[7] B. Ganter, S.O. Kuznetsov, Stepwise construction of the Dedekind–MacNeille, in: Proc. Conceptual Structures: Theory, Tools and Applications, Lecture Notes in Computer Sci., Vol. 1453, Springer, Berlin, 1998, pp. 295–302.
[8] R. Godin, J. Gecsei, Lattice model of browsable data spaces, Inform. Sci. 40 (1996) 89–116.
[9] R. Godin, H. Mili, Building and maintaining analysis-level class hierarchies using Galois lattices, in: Proc. OOPSLA'93, 1993, pp. 394–410.
[10] R. Godin, R. Missaoui, H. Alaoui, Learning algorithms using a Galois lattice structure, in: Proc. 1991 IEEE International Conference on Tools for AI, San Jose, CA, 1991, pp. 22–29.
[11] R. Godin, R. Missaoui, A. April, Experimental comparison of navigation in Galois lattice with conventional information retrieval methods, Internat. J. Man. Machine Studies 38 (1993) 747–767.
[12] M. Habib, R. Medina, L. Nourine, G. Steiner, Efficient algorithms on distributive lattices, RR, LIRMM 95-033, June 1995.
[13] M. Habib, L. Nourine, Tree structure for distributive lattices and its applications, Theoret. Comput. Sci. 165 (2) (1996) 391–405.
[14] C. Jard, G.-V. Jourdan, J.-X. Rampon, Computing on-line the lattice of maximal antichains of posets, Order 11 (3) (1994) 197–210.
[15] M. Missikoff, M. Scholl, An algorithm for insertion into lattices: Application to type classification, June 1989.
[16] E.M. Norris, An algorithm for computing the maximal rectangles of a binary relation, J. ACM 21 (1974) 356–366.
[17] R. Wille, Restructuring lattice theory: An approach based on hierarchies of contexts, in: I. Rival (Ed.), Ordered Sets, NATO ASI, No. 83, Reidel, Dordrecht, 1982, pp. 445–470.
[18] R. Wille, Lattices in data analysis: How to draw them with a computer, in: I. Rival (Ed.), Algorithms and Orders, Kluwer Academic Publishers, Dordrecht, 1989, pp. 33–58.
[19] M.J. Zaki, S. Parthasarathy, M. Ogihara, W. Li, New algorithms for fast discovery of association rules, Technical Report, University of Rochester, 1997.