Very fast instances for concept generation

Anne Berry[1], Ross M. McConnell[2], Alain Sigayret[1], and Jeremy P. Spinrad[3]

[1] LIMOS (CNRS UMR 6158), Université Clermont-Ferrand II, Ensemble scientifique des Cézeaux, 63177 Aubière Cedex, France. berry@isima.fr, sigayret@isima.fr
[2] Computer Science Department, Colorado State University, Fort Collins, CO 80523-1873 USA. rmm@cs.colostate.edu
[3] EECS Department, Vanderbilt University, Nashville, TN 37235 USA. spin@vuse.vanderbilt.edu

Abstract. Computing the maximal bicliques of a bipartite graph is equivalent to generating the concepts of the binary relation defined by the matrix of this graph. We study this problem for special classes of input relations for which concepts can be generated much more efficiently than in the general case; in some special cases, we can even say that the number of concepts is polynomially bounded, and all concepts can be generated particularly quickly.

1 Introduction

One of the important current directions of research related to Formal Concept Analysis deals with the generation of item sets, whether these are defined as 'frequent item sets' or using other more complex criteria. These problems are closely related to concept generation, which has given rise to recent publications (see e.g. [1]).

The problem of concept generation has been shown to be equivalent to various graph problems: computing the maximal transversals of a hypergraph or finding the maximal bicliques of a bipartite graph ([17]). More recently, [4] showed that concept generation is equivalent to generating the minimal separators of a co-bipartite graph. On all three of these problems there exist publications which may well yield algorithmic improvements for concept generation.

In this paper, we aim to use graph results which are related to the form of the matrix representing the relation defined by a context. In some cases we require the matrix to be input in a certain form, in other cases there are good graph algorithms which re-order the rows and columns of the matrix so that the result is in the desired form when the input relation permits it.

We address the issue of generating concepts more quickly than in the general case on special binary matrices defined by a context $(\mathcal{O}, \mathcal{P}, \mathcal{R})$, with the requirement that only polynomial space is used to encounter all the concepts, whether or not their number is polynomial. The best current complexity for the general case of generating all concepts using only polynomial space is of $O(|\mathcal{R}|\cdot|\mathcal{P}|)$ per concept using Ganter's algorithm [9], or $O(|\mathcal{P}|^{\alpha})$ per concept using the version of Bordat's algorithm introduced by Berry, Bordat and Sigayret which uses only

polynomial space [3], where n^α is the time required to perform matrix multiplication, currently $\alpha = 2.376$ ([7]).

We start with the case in which the relation has the *consecutive ones property*: the columns of the matrix representation can be permuted so that in every row, the ones form a consecutive block (such a permutation is called a *consecutive ones arrangement*). We show that in this case, the number of concepts is $O(|\mathcal{R}|)$, and all these concepts can be found in global $O(|\mathcal{R}|)$ time. We generalize from this starting point in several ways.

One form of generalization is to use the decomposition of a general matrix into a PQR-tree ([13]), which efficiently finds submatrices which have the consecutive ones property.

Other forms of generalization come from using natural extensions of the consecutive ones property. Perhaps the most natural is the *circular ones property*. Although the number of concepts can become exponential in this case, we show that the concepts can be generated in $O(|\mathcal{P}|)$ time per concept, which is optimal, since it takes $\Theta(|\mathcal{P}|)$ space to represent a concept in this case. There also are fast algorithms for permuting rows and columns to obtain a circular ones ordering, if such an ordering exists.

Finally, we generalize to orderings in which the number of blocks of ones is at most constant for every row. If we are given an ordering of this type, we can still generate the set of concepts in $O(|\mathcal{P}|)$ time per concept, although no polynomial algorithm is known for finding such an ordering when it is not given as part of the input.

In these time analyses, as in all the rest of the discussions in this paper, $\mathcal{P}$ and $\mathcal{O}$ can be freely interchanged by duality of rows and columns.

2 Background and previous results

In this paper, we consider contexts $(\mathcal{O}, \mathcal{P}, \mathcal{R})$, where $\mathcal{O}$ is the set of objects, $\mathcal{P}$ is the set of properties, and both $\mathcal{O}$ and $\mathcal{P}$ are finite. We will work on the 0-1 matrix of $\mathcal{R}$; we will refer to the elements of the relation as *ones*, and to the non-elements as *zeroes*. We will use the following classical set notations: $+$ denotes the union of two disjoint sets, $-$ denotes set difference.

A *concept* or *closed set*, also called a *maximal rectangle* of $\mathcal{R}$, is a sub-product $A \times B \subseteq \mathcal{R}$ such that $\forall x \in \mathcal{O} - A, \exists y \in B \,|\, (x, y) \notin \mathcal{R}$, and $\forall y \in \mathcal{P} - B, \exists x \in A \,|\, (x, y) \notin \mathcal{R}$. Given a subset $\mathcal{P}_1$ of $\mathcal{P}$ and a subset $\mathcal{O}_1$ of $\mathcal{O}$, we will say that the set $\mathcal{R} \cap (\mathcal{O}_1 \times \mathcal{P}_1)$ is a *sub-relation* of $\mathcal{R}$ which we will denote $\mathcal{R}(\mathcal{O}_1, \mathcal{P}_1)$.

We will also use finite undirected graphs. Such graphs are classically denoted $G = (V, E)$, where V is the vertex set and E is the edge set. A *bipartite graph* is a graph $G = (V_1 + V_2, E)$ where V_1 and V_2 induce edgeless subgraphs (i.e. V_1 and V_2 are independent sets). A *maximal biclique* (or *maximal complete bipartite subgraph*) is a subgraph $H = (W_1 + W_2, F)$, with $W_1 \subseteq V_1, W_2 \subseteq V_2$, such that all edges are present between any vertex of W_1 and any vertex of W_2, and which is maximal for this property. A graph $G = (V, E)$ is said to be an *interval graph* if there exists a one-to-one mapping I from V to a family of intervals of the

real line such that $xy \in E$ iff the corresponding intervals $I(x)$ and $I(y)$ are intersecting.

Example 1. $\mathcal{O} = \{1,2,3,4,5,6\}$, $\mathcal{P} = \{a,b,c,d,e,f\}$. Relation $\mathcal{R}$ is presented below.

$\mathcal{R}$	a	b	c	d	e	f
1		×	×	×	×	
2	×	×	×			
3	×	×				×
4				×	×	
5			×	×		
6	×					

The maximal rectangles/maximal bicliques/concepts are:
$\mathcal{O} \times \emptyset$, $\{2,3,6\} \times \{a\}$, $\{1,2,3\} \times \{b\}$, $\{1,2,5\} \times \{c\}$, $\{1,4,5\} \times \{d\}$, $\{2,3\} \times \{a,b\}$, $\{1,2\} \times \{b,c\}$, $\{1,5\} \times \{c,d\}$, $\{1,4\} \times \{d,e\}$, $\{3\} \times \{a,b,f\}$, $\{2\} \times \{a,b,c\}$, $\{1\} \times \{b,c,d,e\}$, $\emptyset \times \mathcal{P}$.
For example, $\{1,2\} \times \{b,c\}$ is a maximal rectangle of the matrix given above, or, equivalently, a maximal biclique of the graph of Figure 1.

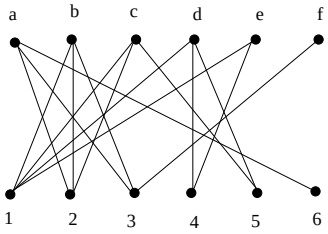

Fig. 1. $\{1,2\} \times \{b,c\}$ is a maximal biclique of graph G associated with relation $\mathcal{R}$ of Example 1.

3 Consecutive Ones Matrices

The starting point for this research is the *consecutive ones property*. A matrix is said to have the consecutive ones property if its columns can be permuted so that in each row all the ones are consecutive. This property has been studied heavily in the context of graph theory: Fulkerson and Gross ([8]) showed that a graph is an interval graph iff the vertex-clique incidence matrix can be permuted to have the consecutive ones property (the vertex-clique incidence matrix shows the maximal cliques of the graph which each vertex belongs to). Booth and Lueker ([6]) gave a linear-time algorithm for finding a consecutive ones arrangement

of a given matrix if such an arrangement exists, as part of their interval graph recognition algorithm.

Let us assume that we have a context $(\mathcal{O}, \mathcal{P}, \mathcal{R})$ whose 0-1 matrix has the consecutive ones property. It takes $O(|\mathcal{P}| + |\mathcal{O}| + |\mathcal{R}|)$ time to find a consecutive ones ordering of the columns ([14]). We claim that every concept of this relation will have a column set that is consecutive in this ordering. Therefore, we can associate with each concept a unique *starting column*, namely, the leftmost of the concept's columns in the ordering.

In this section we will use the common notations for 0-1 matrices; thus R will denote a set of rows in a matrix, C will denote a set of columns. m will denote the number of ones in the matrix.

Example 2. $\mathcal{O} = \{1, 2, 3, 4, 5, 6, 7\}$, $\mathcal{P} = \{a, b, c, d, e, f\}$. Matrix $\mathcal{M}$, presented below, has the consecutive ones property.

	a	b	c	d	e	f
1	×	×	×			
2	×	×	×	×		
3	×	×	×	×	×	×
4		×				
5		×	×	×		
6		×	×	×	×	
7				×	×	

The corresponding concept lattice is presented in Figure 2.

We may delete rows and columns that are all zeros or all ones, compute the lattice for the remaining submatrix, and then correct the elements of this lattice in a trivial way. Henceforth, we will assume that the matrix has no rows that are all zeros or all ones. This implies that the minimal element of the lattice is $\mathcal{O} \times \emptyset$ and the maximal element is $\emptyset \times \mathcal{P}$. We give a recursive algorithm that finds the remaining concepts.

Let $(c_1, c_2, ..., c_p)$ be the order of the columns in a consecutive ones arrangement. The input to each call is a submatrix of the initial matrix given by $\{c_i, c_{i+1}, ..., c_p\}$, the rows that have at least one 1 in any of these columns. In addition, there is a mark on each row that has a 1 preceding column c_i in the original matrix. The purpose of the marks on the rows is to allow the recursive call to avoid returning concepts of the submatrix that are not maximal in the original matrix.

The algorithm finds all concepts of the original matrix that have i as their starting column, and makes a recursive call on columns c_{i+1} through c_p to find the concepts that have their starting column anywhere in $\{c_{i+1}, c_{i+2}, ..., c_p\}$.

Example 3. In Figure 3, the lefthand table corresponds to matrix $\mathcal{M}$ of Example 2 passed to the initial call of the algorithm, with starting column a. The initial matrix does not have any marked row. The next table shows the submatrix, $\mathcal{M}$ minus the first column, passed to the second call (starting column b). In

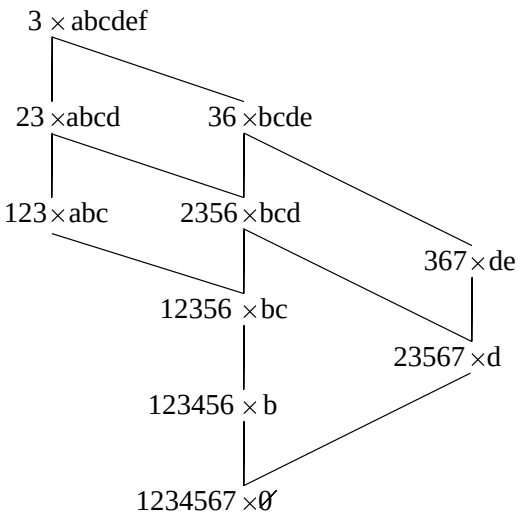

Fig. 2. Concept lattice associated with the consecutive ones matrix of Example 2. The set notations have been simplified for more clarity.

this second call, rows 1, 2 and 3 are marked as their starting column is a. The third table corresponds to a submatrix with starting column c, all rows but 7 are marked. The fourth table corresponds to a submatrix with starting column d; row 1 has disappeared as it has no one in columns d,e,f. There will be no more recursive call.

Let the *ending column* of a row be the rightmost column where the row has a 1. In the recursive call on columns $\{c_i, ..., c_p\}$, let R_i be the set of rows that have a 1 in column i. We permute the rows so that the rows in R_i appear in ascending order of ending column. When the rows of a set R_i are tied for ending column, we place the marked ones before the unmarked ones.

Example 4. In Figure 3, $R_1 = \{1, 2, 3\}$, as these objects have the same starting column a, and these rows appear in ascending order of ending column (respectively c, d, f). In the second call, on columns $\{b, c, d, e, f\}$, we will have $R_2 = \{4, 1, 2, 5, 6, 3\}$. The members of R_2 appear in ascending order of ending column, and the tie between rows 2 and 5 (ending column c) has been broken in favor of 2, since row 2 is marked and row 5 is not.

Let $(r_1, r_2, ..., r_q)$ be the resulting ordering of R_i. The ones in row r_j of the submatrix extend from c_i to c_k for some $k \geq i$. Because of the way R_i has been ordered, every row after r_j in the ordering also has ones in every column in $\{c_i, ..., c_k\}$. Therefore, $\{r_j, ..., r_q\} \times \{c_i, ..., c_k\}$ is a rectangle. It is easy to see that this rectangle is maximal in the submatrix, hence is a concept of the submatrix, if and only if $j = 1$ or r_j extends farther to the right than its predecessor, r_{j-1}.

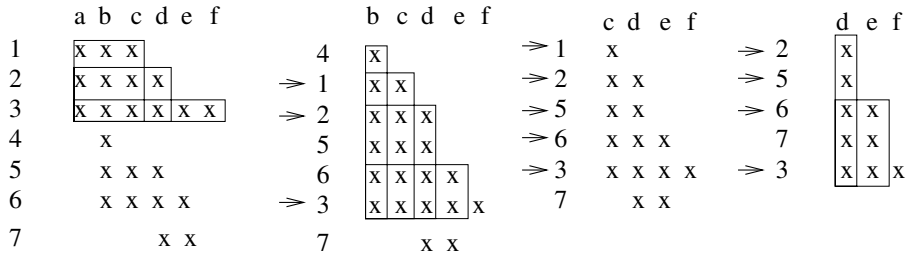

Fig. 3. Finding the concepts in the consecutive ones matrix of Example 3. The corresponding lattice is presented in Figure 2.

A concept of the submatrix can fail to be a concept of the original matrix if and only if it can be extended to a larger rectangle in the whole matrix by adding column c_{i-1}. This is easy to detect: a concept of the submatrix fails to be a concept in the original matrix if and only if all of its rows are marked.

Example 5. In Figure 3, the initial call on rows $(1, 2, 3, 4, 5, 6, 7)$ and starting column a gives rise successively to concepts:
$\{1, 2, 3\} \times \{a, b, c\}$ (as 1 has ending column c, this concept can not be extented to the right), $\{2, 3\} \times \{a, b, c, d\}$ (as 2 has ending column d and 1 has not, this concept can not be upper extended), and $\{3\} \times \{a, b, c, d, e, f\}$.
The second call on rows $(4, 1, 2, 5, 6, 3)$ and starting column b generates:
$\{1, 2, 3, 4, 5, 6\} \times \{b\}$, $\{1, 2, 3, 5, 6\} \times \{b, c\}$, $\{2, 3, 5, 6\} \times \{b, c, d\}$, and $\{3, 6\} \times \{b, c, d, e\}$. As 3 is marked, row set $\{3\}$ generates no new concept: $\{3\} \times \{b, c, d, e, f\}$ is a sub-rectangle of concept $\{3\} \times \{a, b, c, d, e, f\}$ obtained in the initial step.
The third call, on starting column c, generates no concept, as all the rows are marked.
The fourth call, on starting column d, generates $\{2, 3, 5, 6, 7\} \times \{d\}$, and $\{3, 6, 7\} \times \{d, e\}$, as at least one object (7) is unmarked in row set $\{2, 3, 5, 6, 7\}$ and in row set $\{3, 6, 7\}$. $\{3\}$ fails to contain an unmarked row and gives rise to no new concept.
The corresponding lattice is presented in Figure 2. Concepts are generated in this figure from left to right and from bottom to top.

Note that, in handling R_i, the algorithm never generates two concepts with the same upper-left hand corner. This gives a one-to-one mapping from the concepts the algorithm generates to the ones in the matrix, so that there can be at most m of these concepts. The algorithm can only fail to generate directly the top and bottom elements of the lattice, so we get the following bound:

Theorem 1. *If a matrix with m ones has the consecutive ones property, then the corresponding relation has at most $m + 2$ concepts.*

That the bound is tight is illustrated by the identity matrix, where each 1 is itself a concept, and $\mathcal{O} \times \emptyset$ and $\emptyset \times \mathcal{P}$ are also concepts. The number of concepts

can be $\Omega(m)$ in dense matrices, as it can be seen by running the algorithm on an $(n/2) \times n$ matrix where row i has ones in columns i through $i + n/2$: half of the ones are the upper-left corner of a concept.

The proof of correctness is elementary, given the foregoing observations. In order to obtain an $O(m)$ time bound, it suffices to spend time proportional to $(\sum_{i=1}^{p} R_i) \in O(m)$. This is also easy to accomplish by elementary techniques. In particular, each concept can be represented by giving its column set as an interval of $(c_1, ..., c_p)$, and giving its row set as an interval on the ordering of R_i computed in the recursive call on columns $(c_i, ..., c_p)$.

4 Matrices with Bounded PQR Diameter

In this section, we will show that the number of concepts may be bounded by the size of the corresponding relation, provided some property on the PQR-decomposition of its matrix.

The *PQ-tree* of a consecutive ones matrix is a way of representing all consecutive-ones orderings of the columns ([6]). It is a tree whose leaves are the columns of the matrix. The tree has two kinds of internal nodes: Q-nodes, whose children are ordered left-to-right, and P nodes, whose children are not ordered. Assigning a left-to-right ordering to children of each P node and reversing the left-to-right ordering of children of any subset of the Q nodes imposes a leaf order on the leaves, hence an order on columns of the matrix. Such an ordering is always a consecutive ones ordering. Conversely, all consecutive ones orderings can be obtained in this way.

[13] provides a generalization of PQ-trees to arbitrary matrices, called the *PQR-tree*, as illustrated in Figure 4. The third types of nodes, *R nodes*, appear if and only if the matrix does not have the consecutive ones property, otherwise the PQR-tree is a PQ-tree. The PQR-tree can be constructed in time proportional to the number of ones of the matrix. One of the interesting aspects of PQR-trees is that is gives a compact representation of all possible PQR arrangements of a matrix.

The PQR-tree has a set-theoretic definition. Let us consider the leaves to be a set V of 'properties', and let us consider each internal node to represent a subset of V, namely the set of leaf descendants of the node. Such a subset corresponds to a row (an 'object') of the matrix, more precisely to the property set associated with this object. We say that two subsets X and Y of V *overlap* if they intersect, but neither contains the other. Let $\mathcal{F}$ be the family of nonempty subsets of V that do not overlap with any row, and let $\mathcal{F}_0$ be the set of members of $\mathcal{F}$ that do not overlap with any other member of $\mathcal{F}$: then $\mathcal{F}_0$ is the set of nodes of the PQ-tree.

As a direct consequence of this definition, every row of a matrix is a union of one or more children of the node of the PQR-tree, as a row that is not such a union must overlap some member of $\mathcal{F}_0$, hence of $\mathcal{F}$, a contradiction. To each internal node of the PQR-tree, we may assign a *quotient matrix*, as follows.

Let $ch(X)$ denote the children of X in the tree, let $Row(X)$ denote the set of rows that are given by the union of more than one member of $ch(X)$, and let $Col(X)$ denote a set of columns with one representative column from each member of $ch(X)$. The quotient at X is the submatrix induced by rows $Row(X)$ and columns $Col(X)$.

The exact choice of representatives from $ch(X)$ to obtain $Col(X)$ is irrelevant, since all choices yield isomorphic submatrices. It is possible that $Row(X)$ is empty, but the family of sets given by $\{Row(X) \neq \emptyset \mid X \text{ is an internal node}\}$ is a partition of the rows.

Example 6. Figure 4 shows a PQR-decomposition of a matrix. Traditionally, the Q nodes are drawn as horizontal bars, the P nodes are drawn as points, and R nodes are drawn as filled ovals.

In this decomposition, the children of $V = \{a, b, c, d, e, f, g, h, i, j, k, l\}$ are $W = \{a, b, c, d, e\}$, $\{f\}$ and $Y = \{g, h, i, j, k, l\}$ (Q decomposition), rows 1 and 2 are the rows that are unions of more than one of these children, and these rows, together with a selection of one column in each of W, $\{f\}$ and Y, yield the two-row quotient whose rows are $(1, 1, 0)$ and $(0, 1, 1)$.

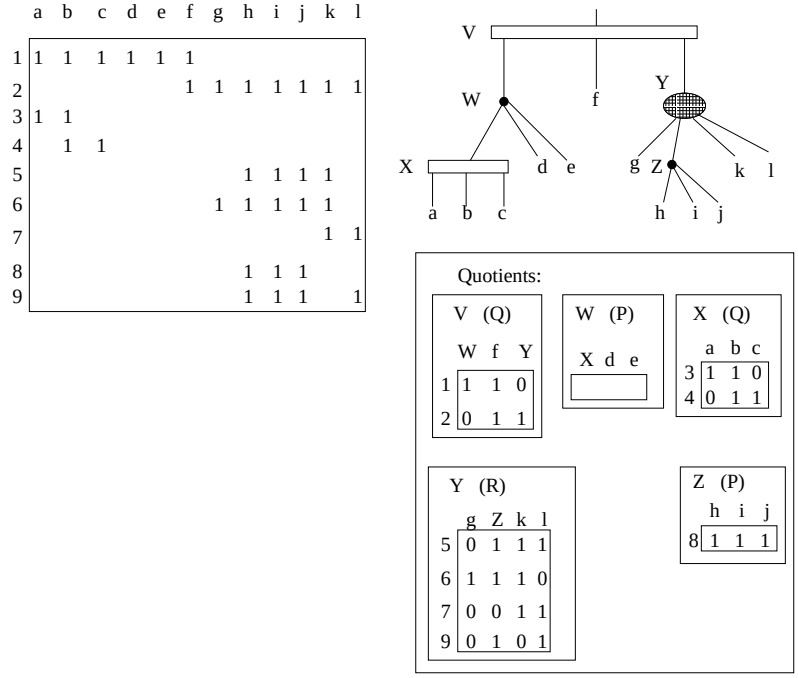

Fig. 4. The matrix of Example 6 and the corresponding PQR-tree and quotients.

Lemma 1. *In a PQR-tree, a node is a P node if and only if its quotient matrix is a rectangle of ones.*

The matrix can be uniquely reconstructed from the quotients by inverting this process. We now illustrate a similar operation by which the concepts of the original matrix can be obtained from the concepts that occur in the quotients. In Example 6, rows $\{5, 6\}$ and columns $\{Z, k\}$ are a concept in the quotient at Y. Substituting $\{h, i, j\}$ for Z in the column set, and adding rows labeled Y in Y's parent V yields the concept with rows $\{2, 5, 6\}$ and columns $\{h, i, j, k\}$.

In general, each concept $A_1 \times B_1$ in the original matrix is obtained from a concept $A_0 \times B_0$ in a quotient at a node X, by expanding the sets that appear as column labels in B_0 and adding rows to A_0 that have a 1 in a column labeled with an ancestor of X in a quotient at an ancestor of X.

As in the case of consecutive ones matrices, we may use a compact representation of each concept found. From results shown in [13] we can derive that, on the path from the quotient of $A_0 \times B_0$ to the root, at least every other quotient has at least one 1 in every column. Each of these contributes at least one row to A_1. The length of the path to the root is $O(|A_1|)$, and the time to list out the elements of A_1 and B_1, given $A_0 \times B_0$, is $O(|A_1| + |B_1|)$. Therefore, we may let $A_0 \times B_0$ serve to represent $A_1 \times B_1$.

Definition 1. *The* decomposition diameter *of a binary matrix M is the maximum number of children of an R node, or 1 if there are no R nodes.*

The following is a consequence of the foregoing observations:

Theorem 2. *If a matrix has a bounded decomposition diameter, the corresponding relation $\mathcal{R}$ has $O(|\mathcal{R}|)$ concepts, and they can be found in $O(|\mathcal{R}|)$ time.*

5 Circular Ones Property

Perhaps the more natural generalization of the consecutive ones property is the circular ones property. When we first considered this generalization, we were discouraged to find that the number of concepts becomes exponential, for example, a matrix with only a diagonal of zeroes, which clearly has the circular ones property, has a lattice isomorphic to that of the power set of $\mathcal{P}$ or $\mathcal{O}$.

To generate concepts efficiently in this case, we will use the concept generation process described in [3]: the algorithm starts with the minimum element of the lattice, and recursively processes the direct successors of each concept. Since in a concept lattice, all the concepts containing a given property form a sub-lattice, we can store in the recursive stack information necessary to avoid re-processing a concept. Another useful feature is that each concept $A \times B$, $(A \subseteq \mathcal{O}, B \subseteq \mathcal{P})$ is the minimal element of a sub-lattice described by sub-relation $\mathcal{R}(A, \mathcal{P} - B)$. Finally, the direct successors of concept $A \times B$ are described by the properties X which in $\mathcal{R}(A, \mathcal{P} - B)$ are not properly contained in another; the elements of X which have identical columns are then grouped together to

be added to A to form one of the successors of $A \times B$. The bottleneck of this algorithm is computing the containments; this requires $O(|\mathcal{P}|^2 \cdot |\mathcal{O}|)$ in general.

However, we will show that the time for generating all concepts can be significantly reduced if the matrix has a circular ones ordering.

First, we note that although PQ-trees are generally viewed as a tool for finding a consecutive ones ordering, they can also be used to find a circular ordering in $O(|\mathcal{P}|+|\mathcal{O}|+|\mathcal{R}|)$ time. Thus we may assume that we have a circular arrangement of objects, and for each property, we are given the circular ordering of its objects in concise form. For example, if there are 100 objects, the objects of a property p_1 may be given as $o_{19} - o_{54}$, while those of p_2 may be given as $o_1 - o_{23}$ and $o_{93} - o_{100}$. Given this form of storage, for each property p_i, it is easy to determine whether p_i contains property p_j in constant time. This simple observation, together with the fact that an arrangement remains circular as objects are deleted, is the key to reducing the time for concept generation using polynomial space, from $O(|\mathcal{P}|^2 \cdot |\mathcal{O}|)$ time per generated concept to $O(|\mathcal{P}|)$ per generated concept.

6 Constant Number of Blocks

Our last extension of the consecutive ones property is to relations given as a matrix in which the number of blocks of ones entries in every row is bounded by a constant.

For each property, we maintain a concise representation of the ones in the property; that is, for a property p_i with k_i blocks, the starting column number and the ending column number of each block: $j_1 - j_2$, $j_3 - j_4$, ..., $j_{2k_i-1} - j_{2k_i}$.

As long as the number of blocks is constant, it is still possible to test containment between properties in constant time. It is also clear that properties continue to have at most a fixed number of blocks of ones as objects are deleted from the current universe of discourse. These observations allow us to generate the concepts in $O(|\mathcal{P}|)$ time per concept, using the same strategy as in the circular ones case.

In one sense, going to a constant number k of consecutive blocks seems to be a huge generalization of the previous algorithms: the consecutive ones property corresponds to $k = 1$, while circular ones are a special case of $k = 2$. One key advantage of the earlier cases discussed in this paper is the existence of polynomial time algorithms to permute rows and columns of a matrix to obtain a matrix with the consecutive/circular ones property, if such a permutation exists. No such algorithm is known for k blocks of consecutive ones. An obvious open question is to investigate whether this problem is polynomial or NP-complete. However, in applications such as social science yes/no questionnaires ([2]), this kind of arrangement may often arise naturally, for example when some questions are logically related to others.

7 Conclusion

This paper shows that for a number of special classes of matrices, the matrix properties can be used to design efficient algorithms for concept generation. Indeed, even in the general case, if we use a recursive concept-generation algorithm such as the one described in Section 5, we can afford to check on each sub-relation encountered whether the matrix has the consecutive ones property, or the circular ones property, and in this case speed up the rest of the remaining recursive call from $O(|\mathcal{P}|^2 \cdot |\mathcal{O}|)$ time to $O(|\mathcal{P}| \cdot |\mathcal{O}|)$, $O(|\mathcal{P}| + |\mathcal{O}|)$ or even constant time per generated concept.

These results show that for special classes of input, the number of concepts may be much less than exponential, notwithstanding the matrix density.

One of the questions which arises is how to embed a relation within one of our special classes, while adding or removing a small or inclusion-minimal set of ones.

There are many other classes of matrices and bipartite graphs to consider for this problem. One example which springs to the mind is that of 'Gamma-free matrices' which are obtainable if, and only if, the corresponding bipartite graph is weakly chordal and bipartite (also called 'chordal bipartite'). In this case, [11] showed that all the maximal bicliques can be identified in $\min(|\mathcal{R}| \cdot \log(|\mathcal{O}| + |\mathcal{P}|), (|\mathcal{O}| + |\mathcal{P}|)^2)$ time. Other results may later appear using this relationship between graphs and lattices.

References

1. Alexe G., Alexe S., Crama Y., Foldes S., Hammer P.L., Simeone B.: Consensus algorithm for the generation of all maximal bicliques. *Discrete Applied Mathematics*, **145** (2004), 11–21.
2. Barbut M., Monjardet B.: *Ordre et classification*. Classiques Hachette, (1970).
3. Berry A., Bordat J-P., Sigayret A.: Concepts can't afford to stammer. *INRIA Proc. International Conference "Journées de l'Informatique Messine" (JIM'03), Metz (France)*, (Sept. 2003). Submitted as 'A local approach to concept generation.'
4. Berry A., Sigayret A.: Representing a concept lattice by a graph. *Discrete Applied Mathematics*, **144(1-2)** (2004) 27–42.
5. Bordat J-P.: Calcul pratique du treillis de Galois d'une correspondance. *Mathématiques, Informatique et Sciences Humaines*, **96** (1986) 31–47.
6. Booth S., Lueker S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.*, **13** (1976) 335–379.
7. Coppersmith D., Winograd S.: On the Asymptotic Complexity of Matrix Multiplication. *SIAM J. Comput.*, **11**:3 (1982) 472–492.
8. Fulkerson D.R., Gross O.A.: Incidence matrices and interval graphs. Pacific J. Math. **15** (1965) 835–855.
9. Ganter B.: Two basic algorithms in concept analysis. *Preprint 831, Technische Hochschule Darmstadt*, (1984).
10. Ganter B., Wille R.: *Formal Concept Analysis*. Springer, (1999).
11. Kloks T., Kratsch D.: Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph. Information Processing Letter **55** (1995) 11–16.

12. Kuznetsov S. O., Obiedkov S. A.: Comparing performance of algorithms for generating concept lattices. *Journal for Experimental and Theoretical Artificial Intelligence (JETAI)*, **14:2-3** (2002) 189–216.
13. McConnell R. M.: A certifying algorithm for the consecutive ones property. *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA04)*, **15**, (2004) 761–770.
14. Paige R., Tarjan R. E.: Three Partition Refinement Algorithms. *SIAM Journal on Computing*, **16** (1987) 973–989.
15. Spinrad J. P.: *Efficient Graph Representation*. Fields Institue Monographs; 19. American Mathematical Society, Providence (RI, USA), (2003).
16. Spinrad J. P.: Doubly Lexical Orderings of Dense 0-1 Matrices. *Information Processing Letters*, **45** (1993) 229–235.
17. Zaki M. J., Parthasarathy S., Ogihara M., Li W.: New Algorithms for Fast Discovery of Association Rules. *Proceedings of 3rd Int. Conf. on Database Systems for Advanced Applications*, (April 1997).