Schwender, Holger

Working Paper

Minimization of Boolean expressions using matrix algebra

Technical Report, No. 2007,09

Provided in Cooperation with:
Collaborative Research Center 'Reduction of Complexity in Multivariate Data Structures' (SFB 475), University of Dortmund

Suggested Citation: Schwender, Holger (2007) : Minimization of Boolean expressions using matrix algebra, Technical Report, No. 2007,09, Universität Dortmund, Sonderforschungsbereich 475 - Komplexitätsreduktion in Multivariaten Datenstrukturen, Dortmund

This Version is available at:
https://hdl.handle.net/10419/24994

Minimization of Boolean Expressions Using Matrix Algebra

Holger Schwender

Collaborative Research Center SFB 475

University of Dortmund

holger.schwender@udo.edu

Abstract

The more variables a logic expression contain, the more complicated is the interpretation of this expression. Since in a statistical sense prime implicants can be interpreted as interactions of binary variables, it is thus advantageous to convert such a logic expression into a disjunctive normal form consisting of prime implicants.

In this paper, we present two algorithms based on matrix algebra for the identification of all prime implicants comprised in a logic expression and for the minimization of this set of prime implicants.

1 Introduction

Boolean combinations of binary variables with outcome true or false can be employed as predictors for the response in classification and regression problems (Ruczinski et al., 2003). Such *logic expressions* are of particular interest in genetic association studies in which high-order interactions of biological variables such as SNPs (Single Nucleotide Polymorphisms) are assumed to be more important for the prediction than the variables themselves.

SNPs are characterized by the possibility of (typically two) different bases at a specific single base-pair position in the DNA sequence, where each of the bases has to occur in at least 1% of the population to distinguish this variation from, e.g., spontaneous mutations. Since the human genome consists of pairs of chromosomes, the less frequent variant can occur in the DNA sequence of none, one, or two of the two chromosomes. Thus, examples for binary variables in association studies are

S_1 : "At least one of the bases explaining SNP S is of the less frequent variant,"

and

S_2 : "Both bases explaining SNP S are of the less frequent variant."

These variables can be negated by the operator ^{C} (e.g., S_1^C means "None of the bases explaining S is of the less frequent variant."), and combined to a logic expression using the operators $\wedge$ (AND) and $\vee$ (OR).

A disadvantage of logic expressions is that it becomes more complicated to interpret them, the more variables they contain. Since we are interested in interactions, a representation of a logic expression would be preferable – and easier to interpret – that reveals directly which variables interact with each other. We, therefore, propose to convert each logic expression into a *disjunctive normal form* (*DNF*), i.e. an OR-combination (*disjunction*) of AND-combinations (*conjunctions*), as conjunctions can be interpreted as interactions of variables.

To avoid redundancy, such a DNF should only consist of *prime implicants*, i.e. conjunctions of minimum order/length. If, e.g., both $A \wedge B \wedge C$ and $A \wedge B \wedge C^C$ are part of a DNF, then C will be redundant, as the DNF will be true if $A \wedge B$ is true, no matter whether C is true or false. These two conjunctions can thus be combined to the prime implicant $A \wedge B$.

If the goal of a study is the detection of interactions associated with the covariate of interest (as, e.g., in Schwender and Ickstadt, 2006), we would

stop here, since we would like to consider all identified interactions, i.e. all prime implicants. For other purposes such as classification, it might be better to reduce the set of prime implicants to a minimum size, as measuring genetic variables is very costly, and we are in general interested in having a classification rule consisting of as few variables as possible. Minimizing the DNF would not lead to a worse classification, as the outcome of the DNF is the same no matter if it is minimized or not.

The classical procedures for minimizing Boolean expressions are the *Karnaugh mapping* (Karnaugh, 1953) and the *Quine-McCluskey algorithm* (Quine, 1952, McCluskey, 1956). Since Karnaugh maps are hard to use if a logic expression contains more than four variables, but we are interested in expressions containing up to about 16 variables, we will only consider the Quine-McCluskey method in the following that consists of the two above-mentioned steps:

1. Identify all prime implicants belonging to a logic expression.

2. Minimize the set of prime implicants.

In the first step, the *minterms* for which the logic expression is true are recursively reduced (e.g., $A \wedge B \wedge C$ and $A \wedge B \wedge C^C$ are combined to $A \wedge B$) to generate the set of all prime implicants, where a minterm is one of the 2^m conjunctions of length m composed of the otucomes of the m binary variables comprised by the logic expression. In the second step, this set is minimized by removing prime implicants from the set that cover the minterms that are also explained by other prime implicants.

In this paper, we present two algorithms based on matrix algebra for the identification of the prime implicants and the minimization of a logic expression. While the algorithm for the second step employs the ideas of the Quine-McCluskey algorithm, the algorithm for the first step follows the opposite way: Instead of successively combining conjunctions, we first consider all variables individually to identify those variables that are prime implicants.

Then, two-way interactions are considered to detect the prime implicants consisting of two variables, and so on.

This paper is organized as follows: In Section 2, the Quine-McCluskey procedure is presented, whereas in Section 3 we introduce the matrix algebra based algorithms. Besides a formal description, all approaches are explained using examples. Finally, Section 4 contains a discussion of the new algorithms.

2 Quine-McCluskey Algorithm

Typically, the starting point of the Quine-McCluskey algorithm is either a logic expression L, or directly a table $\mathbf{T}$ in which each row represents a minterm, i.e. one of the 2^m possible combinations of the m variables for which L is true.

Example 2.1 Assume that $\mathbf{T}$ is given by

	X_1	X_2	X_3	X_4
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
9	1	0	0	1
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

The fifth row of this table, i.e. the row named 9, represents, e.g., the minterm $X_1 \wedge X_2^C \wedge X_3^C \wedge X_4$, where the name of each row of $\mathbf{T}$ is the decimal number

corresponding to the binary number composed by the entries of this row.

An obvious disjunctive normal form of the logic expression corresponding to **T** is

$$\begin{aligned} L = {} & (X_1^C \wedge X_2^C \wedge X_3^C \wedge X_4^C) \ \vee \ (X_1^C \wedge X_2^C \wedge X_3^C \wedge X_4) \ \vee \\ & (X_1^C \wedge X_2^C \wedge X_3 \wedge X_4^C) \ \vee \ (X_1^C \wedge X_2^C \wedge X_3 \wedge X_4) \ \vee \\ & (X_1 \wedge X_2^C \wedge X_3^C \wedge X_4) \ \vee \ (X_1 \wedge X_2^C \wedge X_3 \wedge X_4) \ \vee \\ & (X_1 \wedge X_2 \wedge X_3^C \wedge X_4^C) \ \vee \ (X_1 \wedge X_2 \wedge X_3^C \wedge X_4) \ \vee \\ & (X_1 \wedge X_2 \wedge X_3 \wedge X_4^C) \ \vee \ (X_1 \wedge X_2^C \wedge X_3^C \wedge X_4), \end{aligned}$$

i.e. a disjunction of all the minterms in **T**. However, there are many redundant conjunctions in this DNF that can be removed. For example, the minterms $X_1 \wedge X_2 \wedge X_3 \wedge X_4^C$ and $X_1 \wedge X_2^C \wedge X_3^C \wedge X_4$ can be combined to $X_1 \wedge X_2 \wedge X_3$ without loosing any information.

The process of successively joining minterms to identify the prime implicants, i.e. conjunctions that cannot be further combined, is the first step of the Quine-McCluskey algorithm.

2.1 Identification of the Prime Implicants

For an easier evaluation, the n_T rows of **T** are ordered by the number of 1's they contain. Each pair of rows is considered to determine whether they can be combined or not. Since a pair can only be joined if $m - 1$ entries of the two rows are identical, it is actually only necessary to compare each row from one block with each row of the neighbor block, i.e. each row containing no 1's is compared with each row comprising exactly one 1, each row that sums up to 1 is compared with each row showing two 1's, and so on.

If two rows can be joined, then the combined row will be added to a new table containing all minterms that are combined once. For each of these new rows, the entry that differs between the parent rows is set to "–", whereas all the other entries are equal to the respective elements of the parent rows.

For Example 2.1, the ordered version of **T** and the new table are shown on the left and the right hand side, respectively, of

	X_1	X_2	X_3	X_4
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
9	1	0	0	1
12	1	1	0	0
11	1	0	1	1
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

$\longrightarrow$

	X_1	X_2	X_3	X_4
$(0,1)$	0	0	0	–
$(0,2)$	0	0	–	0
$(1,3)$	0	0	–	1
$(1,9)$	–	0	0	1
$(2,3)$	0	0	1	–
$(3,11)$	–	0	1	1
$(9,11)$	1	0	–	1
$(9,13)$	1	–	0	1
$(12,13)$	1	1	0	–
$(12,14)$	1	1	–	0
$(11,15)$	1	–	1	1
$(13,15)$	1	1	–	1
$(14,15)$	1	1	1	–

where, e.g., row $(0,1)$ of the new table is a combination of the rows named 0 and 1 in **T**.

Afterwards, the new table is considered to identify pairs of rows that can be combined. Again, each row of a block (marked by the solid lines within the tables) is compared to each row of the neighbor block, and if two rows can be merged, then they will be stored in a new table.

This procedure is repeated until no rows of the latest table can be combined with each other anymore. Any of the rows in any of the tables that could not be merged with another row of this table corresponds to a prime implicant.

In Example 2.1, the procedure stops after the second iteration, as

	X_1	X_2	X_3	X_4
$(0, 1, 2, 3)$	0	0	–	–
$(1, 3, 9, 11)$	–	0	–	1
$(9, 11, 13, 15)$	1	–	–	1
$(12, 13, 14, 15)$	1	1	–	–

does not contain a pair of rows that can be further combined.

Since each of the rows of the two other tables could be combined with at least one other row, only the rows of this final table correspond to the prime implicants. Hence, the disjunctive normal form consisting of the prime implicants is given by

$$L = \ (X_1^C \wedge X_2^C) \ \vee \ (X_2^C \wedge X_4) \ \vee \ (X_1 \wedge X_4) \ \vee \ (X_1 \wedge X_2).$$

2.2 Minimizing the Set of Prime Implicants

In the next step, the set of prime implicants is reduced by the following procedure:

1. Construct the prime implicant table.
2. Reduce the prime implicant table by
   (a) removing the essential prime implicants,
   (b) removing rows that dominate other rows,
   (c) removing columns dominated by other columns,
   (d) repeating Steps 2 (a)–(c) until no further reduction is possible.
3. If necessary, solve the remaining prime implicant table.

The goal of this reduction is that

– this set contains a minimum number of prime implicants,

– each minterm represented in **T** is covered by at least one prime implicant.

Construct the Prime Implicant Table. The *prime implicant table* is an $n_T \times p$ matrix indicating which of the n_T minterms represented in **T** are covered by which of the p prime implicants, i.e. which of the minterms have been combined to generate a particular prime implicant.

For Example 2.1, the prime implicant table is thus given by

	$X_1 \wedge X_2$	$X_1^C \wedge X_2^C$	$X_1 \wedge X_4$	$X_2^C \wedge X_4$
0	0	1	0	0
1	0	1	0	1
2	0	1	0	0
3	0	1	0	1
9	0	0	1	1
11	0	0	1	1
12	1	0	0	0
13	1	0	1	0
14	1	0	0	0
15	1	0	1	0

Essential Prime Implicants. If any of the minterms is covered by only one of the prime implicant, then this prime implicant will be *essential* for correctly evaluating the logic expression, and must hence be part of the minimum set. Both the columns corresponding to the essential prime implicants and the rows covered by these prime implicants are removed from the table.

In Example 2.1, $X_1 \wedge X_2$ and $X_1^C \wedge X_2^C$ are essential prime implicants, as they are the only prime implicants that cover the minterms named 12 and 14, and 0 and 2, respectively, in the above table. Removing the corresponding columns and the rows covered by $X_1 \wedge X_2$ or $X_1^C \wedge X_2^C$ from the prime

implicant table leads to

	$X_1 \wedge X_4$	$X_2^C \wedge X_4$
9	1	1
11	1	1

Row Dominance. A row *dominates* another row if it is covered by the same prime implicants as the dominated row as well as by other prime implicants. If, e.g., $\mathbf{r}_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$, $\mathbf{r}_2 = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}$, and $\mathbf{r}_3 = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}$ are rows of the prime implicant table, then $\mathbf{r}_1$ dominates both $\mathbf{r}_2$ and $\mathbf{r}_3$. Dominating rows such as $\mathbf{r}_1$ can be removed from the prime implicants table, since $\mathbf{r}_1$ is covered if, e.g., $\mathbf{r}_2$ (or $\mathbf{r}_3$) is covered. If two rows dominate each other – as in Example 2.1 – then only one of the rows is eliminated. Thus, removing row dominances from the above prime implicant table results, e.g., in

	$X_1 \wedge X_4$	$X_2^C \wedge X_4$
9	1	1

Column Dominance. Contrary to the row dominance step in which dominating rows are eliminated, columns dominated by other columns are removed in the column dominance step. If thus a prime implicant P_1 covers the same minterms as the prime implicant P_2 as well as additional minterms, the column in the prime implicant table corresponding to P_2 will be removed. Otherwise, i.e. if dominating rows would be removed, P_2 and additional prime implicants – instead of just P_1 – would be required to cover the same set of minterms.

In Example 2.1, the two remaining prime implicants dominate each other such that either of them can be removed from the table leading, e.g., to

	$X_1 \wedge X_4$
9	1

These three steps (removing essential prime implicants, column and row dominance) are repeated until no further reduction of the prime implicant table is possible anymore, either because this table is empty, or because there are no further essential prime implicants, or column or row dominances.

As $X_1 \wedge X_4$ is a (secondary) essential prime implicant, it is removed from the table in the second iteration of the reduction step. Since the prime implicant table is empty after this removal, the Quine-McCluskey algorithm stops, and a minimum disjunctive normal form of L is given by

$$L = \ (X_1^C \wedge X_2^C) \ \vee \ (X_1 \wedge X_4) \ \vee \ (X_1 \wedge X_2).$$

Solving the Prime Implicant Table. The prime implicant table is, however, not always empty after all essential prime implicants, all column and all row dominances have be eliminated.

Example 2.2 Consider the prime implicant table

P_1	P_2	P_3	P_4	P_5	P_6
0	1	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	1	0	0	1	0
1	0	1	0	0	0
1	0	0	1	0	0

Since none of the rows of this table is covered by only one of the prime implicants P_i, $i = 1, \ldots, 6$, or dominates another one, and none of the columns is dominated by another column, it is not possible to further reduce this prime

implicant table. A minimum DNF, however, would not be composed of all six prime implicants, as, e.g., $P_1 \vee P_5 \vee P_6$ cover all the minterms included in this table. But this is just one of the possible subsets that cover all minterms.

Instead of trying which subset of the remaining prime implicants, on the one hand, covers all the minterms, and on the other hand, contains the minimum number of prime implicants, this *cyclic covering problem* can be analytically solved by *Petrick's method* (Petrick, 1956).

Petrick's method. For $g = 1, \ldots, p$, set

$$C_g : \text{“Prime implicant } P_g \text{ is included in the covering subset.”}$$

and let C be a Boolean combination of the *propositions*, i.e. variables with outcome true or false, C_g, $g = 1, \ldots, p$, that is true if all rows of the prime implicants table are covered, and false otherwise.

In Example 2.2, the first row is covered if $C_2 \vee C_6$, the second row is covered if $C_4 \vee C_6$, and so on. The whole table is therefore covered if

$$\begin{aligned} C = \; & (C_2 \vee C_6) \;\wedge\; (C_4 \vee C_6) \;\wedge\; (C_3 \vee C_5) \;\wedge \\ & (C_2 \vee C_5) \;\wedge\; (C_1 \vee C_3) \;\wedge\; (C_1 \vee C_4) \\ = \; & (C_3 \wedge C_4 \wedge C_5 \wedge C_6) \;\vee\; (C_2 \wedge C_3 \wedge C_4) \;\vee \\ & (C_1 \wedge C_5 \wedge C_6) \;\vee\; (C_1 \wedge C_2 \wedge C_4 \wedge C_5) \;\vee \\ & (C_1 \wedge C_2 \wedge C_3 \wedge C_6) \end{aligned} \tag{2.1}$$

is true. If thus one of the five conjunction in the disjunctive normal form of (2.1) is true, then C will be true. Since we are looking for a minimum set of prime implicants, each of the conjunctions of order 3, i.e. either $C_2 \wedge C_3 \wedge C_4$ or $C_1 \wedge C_5 \wedge C_6$, is a solution to the cycling covering problem. The minimum DNF will hence contain either $\{P_2, P_3, P_4\}$ or $\{P_1, P_5, P_6\}$.

3 Logic Minimizing Based on Matrix Algebra

In this section, we present two new algorithms for minimizing a logic expressions. These algorithms make essential use of matrix computation to speed up the computation. As in the Quine-McCluskey procedure, we first identify the set of all prime implicants, and then reduce it to a minimum size. However, in particular the first step of our approach differs substantially from the Quine-McCluskey algorithm: Instead of successively combining minterms to generate the set of all prime implicants, we start with conjunctions of order 1, i.e. with binary variables or their complement, and check if any of these conjunctions is a prime implicant. Afterwards, we consider conjunctions of order 2, i.e. two-way interactions, to detect prime implicants of order 2. Then, conjunctions of order 3 are examined, and so on. This procedure stops when all minterms for which the logic expression of interest is true are covered by at least one of the identified prime implicants.

Since different types of the indicator function are used in the following, it is defined for these situations before the algorithms are described.

Definition 3.1 The *indicator function* I is a function that compares two object, where the structure of the outcome of I consisting only of ones and zeros depends on the structure of the two object.

(a) If $\mathbf{X}$ is an $n \times m$ matrix and y is a numeric value, then the outcome of the indicator function $I(\mathbf{X} = y)$ is an $n \times m$ matrix with elements

$$i_{k\ell} = \begin{cases} 1, & \text{if } x_{k\ell} = y \\ 0, & \text{if } x_{k\ell} \neq y \end{cases}, \; k = 1, \ldots, n, \; \ell = 1, \ldots, m.$$

(b) If $\mathbf{X}$ is an $n \times m$ matrix and $\mathbf{y}$ is a vector of length n, then the outcome of the indicator function $I(\mathbf{X} = \mathbf{y})$ is an $n \times m$ matrix with elements

$$i_{k\ell} = \begin{cases} 1, & \text{if } x_{k\ell} = y_k \\ 0, & \text{if } x_{k\ell} \neq y_k \end{cases}, \; k = 1, \ldots, n, \; \ell = 1, \ldots, m.$$

(c) If both $\mathbf{X}$ and $\mathbf{Y}$ are $n \times m$ matrices, then the outcome of the indicator function $I(\mathbf{X} = \mathbf{Y})$ is an $n \times m$ matrix with elements

$$i_{k\ell} = \begin{cases} 1, & \text{if } x_{k\ell} = y_{k\ell} \\ 0, & \text{if } x_{k\ell} \neq y_{k\ell} \end{cases}, \ k = 1, \ldots, n, \ \ell = 1, \ldots, m.$$

Analogously, define $I(\mathbf{X} > y)$ for any type of y by replacing each "=" by ">", and each "≠" by "≤" in Definition 3.1.

Note that the two cases $\mathbf{x}$ is a vector and $\mathbf{y}$ is either a numeric value or a vector of the same length as $\mathbf{x}$ are special cases of Definition 3.1 (a) and (b), respectively.

3.1 Identification of the Prime Implicants

The idea behind our procedure for the identification of prime implicants is based on the following theorem.

Theorem 3.1 Let $\mathbf{T}$ be an $n_T \times m$ matrix containing all the n_T minterms of length m for which the logic expression of interest is true. A conjunction of order i will be a prime implicant if

(a) 2^{m-i} rows of $\mathbf{T}$ contain this conjunction,

(b) no prime implicant of lower order exists that is part of this conjunction.

Proof. (b) follows directly from the fact that a conjunction has to be minimal, i.e. cannot be further combined with other conjunctions, to be considered as prime implicant.

For (a), note that if a prime implicant consists of i variables, then the corresponding row in the table containing the combined minterms (see Section 2.1) will consist of i values that are either 0 or 1, and $m - i$ entries being "−". Each of these "−" has been generated by successively combining two minterms. A row containing one "−" is thus obtained by combining

$2^{m-(m-1)} = 2$ rows with no "–", each row consisting of two "–" is generated by joining two rows each comprising one "–", i.e. by combining $2^{m-(m-2)} = 4$ rows with no "–", and so on. Consequently, a row with $m - i$ entries being "–", and hence the corresponding conjunction of order i are generated by combining 2^{m-i} rows containing no "–", i.e. by joining 2^{m-i} rows of $\mathbf{T}$.

And none of these 2^{m-i} rows is a duplicate of one of the other $2^{m-i} - 1$ rows. To see this, let's assume that two conjunctions should be combined, and that (at least) one minterm has been employed in the generation of both conjunctions. Two conjunctions can only be joined if exactly one element differs between them. This entry must thus be 0 in one of the conjunctions, and hence in all minterms used in the generation of it, and 1 in the other conjunction and in all its ancestral minterms. This, however, is a contradiction to the assumption that both conjunction have (at least) one common minterm. □

Thus, if a column of $\mathbf{T}$ contains either 2^{m-1} ones or 2^{m-1} zeros, then the corresponding variable or the complement of this variable, respectively, will be a prime implicant. After identifying such prime implicants of order 1, all possible Boolean combinations of two of the variables and their complements are considered. Each of these combinations that appears in 2^{m-2} rows of $\mathbf{T}$ corresponds to a prime implicant of order 2. Afterwards, all interactions of order $3, 4, \ldots$ are considered, until all rows of $\mathbf{T}$ are covered by at least one of the identified prime implicants. The procedure for the detection of all prime implicants is summarized in Algorithm 3.1.

Algorithm 3.1 (Identification of Prime Implicants)
Let $\mathbf{T}$ be an $n_\mathrm{T} \times m$ matrix in which each row corresponds to one of the n_T minterms for which the logic expression L of interest is true, and each column corresponds to one of the m variables $X_1, \ldots, X_m$ composing L.

1. Replace each zero in $\mathbf{T}$ by -1, and set $i = 1$.

2. Let $\mathbf{Q}^{(i)}$ be a $2^i \binom{m}{i} \times m$ with elements

$$q_{kj}^{(i)} = \begin{cases} 1, & \text{if } X_j \text{ is part of the } k^{\text{th}} \text{ conjunction of order } i \\ -1, & \text{if } X_j^C \text{ is part of the } k^{\text{th}} \text{ conjunction of order } i \\ 0 & \text{otherwise} \end{cases}. \quad (3.1)$$

3. Set $\mathbf{S}^{(i)} = \mathbf{Q}^{(i)}\mathbf{T}'$, and compute $\mathbf{h}^{(i)} = I\Big(\mathbf{S}^{(i)} = i\Big)\mathbf{1}_{n_\mathrm{T}}$.

4. Set $\mathcal{H}_i = \Big\{k : h_k^{(i)} = 2^{m-i}\Big\}$. If $\mathcal{H}_i = \emptyset$, set i to $i+1$, and repeat Steps 2-6. Otherwise, update $\mathbf{Q}^{(i)}$ by removing all rows $k \notin \mathcal{H}_i$ from $\mathbf{Q}^{(i)}$ such that $\mathbf{Q}^{(i)}$ becomes a $|\mathcal{H}_i| \times m$ matrix.

5. Denote the set of prime implicants $P_1, \ldots, P_{p_i}$ each of order i or lower by $\mathcal{L}_i$, where $\mathcal{L}_0 = \emptyset$. If $\mathcal{L}_{i-1} = \emptyset$, add the conjunctions represented by the (remaining) rows of $\mathbf{Q}^{(i)}$ to $\mathcal{L}_{i-1}$ to generate $\mathcal{L}_i$. Otherwise,

   (a) let $\mathbf{M}_{i-1}$ denote a $p_{i-1} \times m$ matrix in which each row represents – analogous to (3.1) – one of the p_{i-1} prime implicants of an order lower than i,

   (b) compute $\mathbf{v}^{(i-1)} = \mathrm{diag}\left(\mathbf{M}_{i-1}\mathbf{M}_{i-1}'\right)$, i.e. determine the vector $\mathbf{v}^{(i-1)}$ containing the p_{i-1} diagonal elements of $\mathbf{M}_{i-1}\mathbf{M}_{i-1}'$, and set $\mathbf{R}^{(i-1)} = \mathbf{M}_{i-1}\mathbf{Q}^{(i)\prime}$,

   (c) update $\mathbf{Q}^{(i)}$ by removing any row of $\mathbf{Q}^{(i)}$ corresponding to a non-zero entry in

   $$\mathbf{1}_{p_{i-1}}' I\Big(\mathbf{R}^{(i-1)} = \mathbf{v}^{(i-1)}\Big),$$

   (d) and add the conjunctions corresponding to the remaining rows of $\mathbf{Q}^{(i)}$ to $\mathcal{L}_{i-1}$ to generate $\mathcal{L}_i$.

6. Stop if all elements of $\mathbf{1}_{p_i}' I\Big(\mathbf{M}_i\mathbf{T}' = \mathbf{v}^{(i)}\Big)$ are non-zero. Otherwise, set i to $i+1$, and repeat Steps 2-6.

Instead of considering each of the $2^i \binom{m}{i}$ possible conjunctions of order i, $i = 1, \ldots, m$, separately, a $2^i \binom{m}{i} \times m$ matrix $\mathbf{Q}^{(i)}$ is constructed allowing to examine all conjunctions simultaneously. The second column of $\mathbf{Q}^{(2)}$ in (3.2), e.g., represents the conjunction $X_1 \wedge X_2^C$. In the following, it is explained how Algorithm 3.1 employs $\mathbf{Q}^{(2)}$ to identify the prime implicants of order 2 in Example 3.1.

Example 3.1 Suppose that the matrix $\mathbf{T}$ corresponding to a logic expression L composed of $m = 3$ variables X_1, X_2 and X_3 is given by

$$\mathbf{T}' = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix},$$

and that X_2 has already been identified as the only prime implicant of length 1 such that now all conjunctions of order $i = 2$ are of interest. In this case, the matrix $\mathbf{Q}^{(2)}$ is given by

$$\mathbf{Q}^{(2)} = \left[\begin{array}{rrr} 1 & 1 & 0 \\ 1 & -1 & 0 \\ -1 & 1 & 0 \\ -1 & -1 & 0 \\ \hdashline 1 & 0 & 1 \\ 1 & 0 & -1 \\ -1 & 0 & 1 \\ -1 & 0 & -1 \\ \hdashline 0 & 1 & 1 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \\ 0 & -1 & -1 \end{array}\right]. \tag{3.2}$$

For the detection of all prime implicants of order 2, the rows of $\mathbf{Q}^{(2)}$ are compared with the rows of $\mathbf{T}$ to compute for each of the conjunctions in $\mathbf{Q}^{(2)}$ the number of minterms (represented by the rows of $\mathbf{T}$) that are covered by the respective conjunction.

For this computation, it is necessary to replace each zero in $\mathbf{T}$ by -1 such that complements of variables (represented in $\mathbf{T}$ by zeros), and hence, prime implicants are, on the one hand, identifiable by matrix multiplication, and on the other hand, distinguishable from the variables themselves.

Having replaced the zeros in $\mathbf{T}$ by -1, the k^{th} conjunction of order i in $\mathbf{Q}^{(i)}$ covers, i.e. is part of, the conjunction of order m in the ℓ^{th} row of $\mathbf{T}$, if

$$\sum_{j=1}^{m} q_{kj}^{(i)} t_{\ell j} = i.$$

Therefore, if the $\left(k^{\text{th}}, \ell^{\text{th}}\right)$ element of

$$\mathbf{S}^{(2)} = \mathbf{Q}^{(2)}\mathbf{T}' = \mathbf{Q}^{(2)} \begin{bmatrix} -1 & -1 & 1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -2 & 0 & 0 & 0 & 2 & 2 \\ 0 & -2 & 2 & -2 & 0 & 0 \\ 0 & 2 & -2 & 2 & 0 & 0 \\ 2 & 0 & 0 & 0 & -2 & -2 \\ 0 & -2 & 0 & 0 & 0 & 2 \\ -2 & 0 & 2 & -2 & 2 & 0 \\ 2 & 0 & -2 & 2 & -2 & 0 \\ 0 & 2 & 0 & 0 & 0 & -2 \\ 0 & 0 & -2 & 2 & 0 & 2 \\ -2 & 2 & 0 & 0 & 2 & 0 \\ 2 & -2 & 0 & 0 & -2 & 0 \\ 0 & 0 & 2 & -2 & 0 & -2 \end{bmatrix}$$

is equal to 2, then the conjunction represented by the k^{th} row of $\mathbf{Q}^{(2)}$ covers the minterm corresponding to the ℓ^{th} row of $\mathbf{T}$.

Following Theorem 3.1, a conjunction of order i is a prime implicant if it covers 2^{m-i} rows of $\mathbf{T}$, and if this conjunction is not covered by a prime implicant of lower order. Thus, each conjunction represented by a row of $\mathbf{Q}^{(2)}$ that corresponds to an entry of

$$\mathbf{h}^{(2)} = I\big(\mathbf{S}^{(2)} = 2\big)\mathbf{1}_{n_T} = \begin{bmatrix} 2 & 1 & 2 & 1 & 1 & 2 & 2 & 1 & 2 & 2 & 1 & 1 \end{bmatrix}'$$

equal to $2^{3-2} = 2$, i.e. by a row of the reduced matrix

$$\mathbf{Q}^{(2)} = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 1 & 0 \\ 1 & 0 & -1 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & -1 \end{bmatrix}, \tag{3.3}$$

is a potential prime implicant.

However, the reduced matrix (3.3) still contains all conjunctions of order 2 that are composed of the prime implicant X_2 and another literal, i.e. another variable or complement of a variable. Such conjunctions are removed in Step 5 of Algorithm 3.1 by constructing a $p_{i-1} \times m$ matrix $\mathbf{M}_{i-1}$ containing the identified prime implicants of order $i-1$ or lower. For Example 3.1, e.g., $\mathbf{M}_1$ is given by

$$\mathbf{M}_1 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}.$$

If the prime implicant in the ℓ^{th} row of $\mathbf{M}_{i-1}$ is part of the conjunction represented by the k^{th} row of the (reduced) matrix $\mathbf{Q}^{(i)}$, then the $\big(\ell^{\text{th}}, k^{\text{th}}\big)$ element of $\mathbf{R}^{(i-1)} = \mathbf{M}_{i-1}\mathbf{Q}^{(i)\prime}$ will be equal to the order of the ℓ^{th} prime implicant, where the orders of the p_{i-1} prime implicants are comprised by the vector $\mathbf{v}^{(i-1)}$ containing the diagonal elements of $\mathbf{M}_{i-1}\mathbf{M}'_{i-1}$. Therefore,

each column k of $\mathbf{R}^{(i-1)}$ for which

$$\sum_{\ell=1}^{p_{i-1}} I\left(r_{\ell k}^{(i-1)} = v_{\ell}^{(i-1)}\right) > 0$$

corresponds to a row of $\mathbf{Q}^{(i)}$ consisting of a prime implicant of an order lower than i.

In Example 3.1, $\mathbf{v}^{(1)} = 1$ and

$$\mathbf{R}^{(1)} = \mathbf{M}^{(1)}\mathbf{A}^{(2)\prime} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

such that the first, second, fifth and sixth row of (3.3) can be removed leading to the matrix

$$\mathbf{Q}^{(2)} = \begin{bmatrix} 1 & 0 & -1 \\ -1 & 0 & 1 \end{bmatrix}$$

comprising the two prime implicants, $X_1 \wedge X_3^C$ and $X_1^C \wedge X_3$, of order 2.

Algorithm 3.1 will stop if each of the minterms in $\mathbf{T}$ is covered by at least one of the identified prime implicants. To check if Algorithm 3.1 can be stopped, the same idea as in Step 5 of Algorithm 3.1 is employed: If each of the columns of $\mathbf{M}_i\mathbf{T}'$ consist of at least one entry that is equal to the corresponding entry in $\mathbf{v}^{(i)}$, then each of the rows of $\mathbf{T}$ are covered by at least one of the rows of $\mathbf{M}_i$.

In Example 3.1,

$$\mathbf{M}_2\mathbf{T}' = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & -1 \\ -1 & 0 & 1 \end{bmatrix} \mathbf{T}' = \begin{bmatrix} -1 & 1 & -1 & 1 & 1 & 1 \\ -2 & 0 & 2 & -2 & 2 & 0 \\ 2 & 0 & -2 & 2 & -2 & 0 \end{bmatrix}$$

and

$$\mathbf{v}^{(2)} = \text{diag}\left(M_2 M_2'\right) = \text{diag}\left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & -2 \\ 0 & -2 & 2 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$

such that the first row of $\mathbf{T}$ is covered by $X_1^C \wedge X_3$, the second and the sixth row by X_2, the third row by $X_1 \wedge X_3^C$, the fourth row by both X_2 and

$X_1^C \wedge X_3$, and the fifth row by both X_2 and $X_1 \wedge X_3^C$. Since all minterms represented in $\mathbf{T}$, hence, contain at least one of the identified prime implicants, Algorithm 3.1 stops, and the disjunctive normal form of the logic expression corresponding to $\mathbf{T}$ is given by

$$L = \ X_2 \ \vee \ (X_1 \wedge X_3^C) \ \vee \ (X_1^C \wedge X_3).$$

3.2 Minimizing the Set of Prime Implicants

Contrary to Algorithm 3.1, Algorithm 3.2 follows the ideas of the second step of the Quine-McCluskey algorithm. We thus also reduce a prime implicant table successively by removing essential prime implicants, and column and row dominances, and if necessary, solve the remaining table. We, however, again employ matrix algebra to consider all prime implicants, columns or rows simultaneously in the respective reduction steps.

Algorithm 3.2 (Minimization of the Set of Prime Implicants)
Let $\mathbf{P}$ be the $n_T \times p$ prime implicants table, i.e. a matrix indicating which of the n_T minterms for which the logic expression of interest is true is covered by which of the p prime implicants.

1. Successively reduce $\mathbf{P}$, where the dimensions of $\mathbf{P}$ at the respective stage of reduction are denoted by n_{now} and p_{now}, by

   (a) setting $n_{\text{red}} = n_{\text{now}}$ and $p_{\text{red}} = n_{\text{now}}$,

   (b) removing each column of $\mathbf{P}$ corresponding to a non-zero entry of

   $$\mathbf{e} = \mathbf{P}' \cdot I\big(\mathbf{P}\mathbf{1}_{p_{\text{now}}} = 1\big),$$

   and each row of $\mathbf{P}$ corresponding to a non-zero entry of

   $$\mathbf{P} \cdot I\big(\mathbf{e} > 0\big),$$

   where the prime implicants represented by the removed columns are added to $\mathcal{P}$, the set of the essential prime implicants,

(c) eliminating each row of $\mathbf{P}$ that

(i) is a duplicate of a row with a smaller subscript,

(ii) corresponds to an entry of

$$\mathbf{r}'_{\text{dom}} = \mathbf{1}'_{n_{\text{now}}} \cdot I\big(\mathbf{P}\mathbf{P}' = \mathbf{P}\mathbf{1}_{p_{\text{now}}}\big)$$

not equal to 1,

(d) removing each column of $\mathbf{P}$ that

(i) is a duplicate of a column with a smaller subscript,

(ii) corresponds to an entry of

$$\mathbf{c}_{\text{dom}} = I\big(\mathbf{P}'\mathbf{P} = \mathbf{P}'\mathbf{1}_{n_{\text{now}}}\big) \cdot \mathbf{1}_{p_{\text{now}}}$$

not equal to 1,

(e) repeating 1 (a)–(d), if $n_{\text{now}} < n_{\text{red}}$, or $p_{\text{now}} < p_{\text{red}}$.

2. If $p_{\text{now}} > 0$, solve $\mathbf{P}$ by

(a) setting C_g: "The prime implicant represented by the g^{th} column of $\mathbf{P}$ is included in the subset that should cover the rows of $\mathbf{P}$", $g = 1, \ldots, p_{\text{now}}$,

(b) constructing a $2^{p_{\text{now}}} \times p_{\text{now}}$ matrix $\mathbf{A}$ consisting of zeros and ones, where each row indicates one of the $2^{p_{\text{now}}}$ possible conjunctions of the Boolean propositions C_g, $g = 1, \ldots, p_{\text{now}}$,

(c) computing

$$\mathbf{v} = I\big(\mathbf{A}\mathbf{P}' = 0\big) \cdot \mathbf{1}_{n_{\text{now}}},$$

(d) removing all rows of $\mathbf{A}$ corresponding to a non-zero entry of $\mathbf{v}$ to generate the $n_v \times p_{\text{now}}$ matrix $\mathbf{A}_{\text{red}}$,

(e) computing

$$a_{\min} = \min_{k=1,\ldots,n_v} \{a_k : \ \mathbf{a} = \mathbf{A}_{\text{red}}\mathbf{1}_{p_{\text{now}}}\},$$

(f) adding the prime implicants indicated by one of the rows of $\mathbf{A}_{\text{red}}$ that add up to $a_{\min}$ to $\mathcal{P}$.

The minimum set of prime implicants is given by $\mathcal{P}$.

The entries of each row of $\mathbf{P}$ covered by only one prime implicant sum up to 1. Essential prime implicants can thus be identified by multiplying $\mathbf{P}'$ by a vector indicating which of the rows sum up to one, since only the entries of the vector $\mathbf{e}$ resulting from this matrix calculation will be non-zero that correspond to an essential prime implicant.

Not only the columns representing essential prime implicants, but also rows covered by these conjunctions should be removed from $\mathbf{P}$. Such rows can be identified by multiplying $\mathbf{P}$ with a vector indicating if the entries of $\mathbf{e}$ are non-zero or not, as the non-zero entries of the resulting vector correspond to the rows covered by the essential prime implicants.

Example 3.2 Suppose that the conjunctions P_p, $p = 1, \ldots, 4$, have been identified as prime implicants, and that the corresponding prime implicant table $\mathbf{P}$ is given by

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}. \tag{3.4}$$

Since only the second row of $\mathbf{P}$ sums up to 1, $\mathbf{e}$ is computed by

$$\mathbf{e} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The first column of $\mathbf{P}$ thus contains an essential prime implicant that in turn covers the second and the fifth row of $\mathbf{P}$ as indicated by

$$\mathbf{P} \cdot I(\mathbf{e} > 0) = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \end{bmatrix}'.$$

Hence, P_1 is an essential prime implicant, and $\mathbf{P}$ is reduced to

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

If two or more rows are duplicates of each other, i.e. if rows dominate each other, all but one of these rows are eliminated first, since otherwise all these rows would be removed by the matrix computation described in the next paragraph. Instead of setting

$$\mathbf{U} = I\Big(\mathbf{P}\mathbf{P}' = \mathbf{P}\mathbf{1}_{p_{\text{now}}}\Big) \tag{3.5}$$

and

$$\mathbf{V} = \mathbf{U} * \mathbf{U}' * \mathbf{W},$$

where $\mathbf{W}$ is the $p_{\text{now}} \times p_{\text{now}}$ indicator matrix of the lower triangle of $\mathbf{U}$, i.e.

$$\mathbf{W} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & 0 & 0 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix},$$

and "$*$" denotes the element-wise multiplication of matrices, i.e.

$$v_{k\ell} = u_{k\ell} \cdot u_{\ell k} \cdot w_{k\ell}\,, \quad k, \ell = 1, \ldots, p_{\text{now}},$$

and identifying the rows that are duplicates of other rows with a lower subscript by the rows of $\mathbf{V}$ having at least one non-zero entry, the `R` (Ihaka and Gentleman, 1996) function `duplicated` is employed to detect such duplicated rows.

If the k^{th} row of $\mathbf{P}$ is dominated by no other row, $k = 1, \ldots, n_{\text{now}}$, then only the k^{th} entry in the k^{th} row of $\mathbf{PP}'$ will be equal to the corresponding element of $\mathbf{P1}_{p_{\text{now}}}$, i.e. the vector of the row-wise sums of $\mathbf{P}$. If this row is, however, dominated by the ℓ^{th} row of $\mathbf{P}$, then also the element (k, ℓ) of $\mathbf{PP}'$ will be equal to the ℓ^{th} entry in $\mathbf{P1}_{p_{\text{now}}}$. Therefore, each row of $\mathbf{P}$ corresponding to a column of $\mathbf{U}$, see (3.2) with more than one entry equal to 1 dominates at least one other row, and can therefore be removed from $\mathbf{P}$.

Column dominances can be eliminated similarly by removing columns that dominate each other and that are dominated by another column. Again, the former has to be done first such that the latter columns can be identified by the corresponding rows of

$$I\left(\mathbf{P}'\mathbf{P} = \mathbf{P}'\mathbf{1}_{n_{\text{now}}}\right)$$

with more than one entry equal to 1.

In Example 3.2, there are no rows dominating each other. So

$$\mathbf{U} = I\left(\mathbf{PP}' = \mathbf{P1}_{p_{\text{now}}}\right) = I\left(\left[\begin{array}{ccc} 2 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 3 \end{array}\right] = \left[\begin{array}{c} 2 \\ 2 \\ 3 \end{array}\right]\right) = \left[\begin{array}{ccc} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{array}\right] \tag{3.6}$$

can be computed, and the third row dominates other rows, since the sum over the third column of (3.6) is larger than 1. Removing this row from $\mathbf{P}$ leads to

$$\mathbf{P} = \left[\begin{array}{ccc} 0 & 1 & 1 \\ 1 & 1 & 0 \end{array}\right]. \tag{3.7}$$

Since there are no duplicated columns in (3.7),

$$I\left(\mathbf{P}'\mathbf{P} = \mathbf{1}_{n_{\text{now}}}\mathbf{P}\right) = I\left(\left[\begin{array}{ccc} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{array}\right] = \left[\begin{array}{c} 1 \\ 2 \\ 1 \end{array}\right]\right) = \left[\begin{array}{ccc} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{array}\right] \tag{3.8}$$

can be computed directly. Since the row-wise sums of the first and the third column of (3.8) are larger than 1, the first and the third column of $\mathbf{P}$ are

dominated by another column, and can hence be removed leading to

$$\mathbf{P} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \tag{3.9}$$

where the remaining column of $\mathbf{P}$ represents the prime implicant P_3.

In the second iteration of Step 1 of Algorithm 3.2, the second row of $\mathbf{P}$ in (3.9) is removed because of row dominance, and in the third iteration, P_3 is identified as essential prime implicant such that a minimum disjunctive normal form of (3.4) is given by

$$L = P_1 \vee P_3.$$

Example 3.3 Assume that after Step 1 of Algorithm 3.2, the prime implicant table $\mathbf{P}$ is given by

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix},$$

where the g^{th} column represents the prime implicant P_g, $g = 1, \ldots, 4$.

Since neither essential prime implicants, nor row or column dominances can be removed from $\mathbf{P}$, this table has to be solved.

To solve a prime implicant table $\mathbf{P}$, a $2^p \times p$ matrix $\mathbf{A}$ indicating each of the 2^p possible covering subsets of the p prime implicants is constructed. For example, the second column of $\mathbf{A}'$ in (3.10) represents the subset containing P_1, P_2, and P_3. If the k^{th} row of $\mathbf{P}$ is covered by any of the prime implicants represented by the ℓ^{th} row of $\mathbf{A}$, then element (k, ℓ) of $\mathbf{AP'}$ will be larger than zero. Since all the rows of $\mathbf{P}$ should be covered, we are only interested in rows of $\mathbf{A}$ resulting in no non-zero entries in the respective rows of $\mathbf{AP'}$. Denoting the matrix containing such rows by $\mathbf{A}_{\text{red}}$, each row of $\mathbf{A}_{\text{red}}$ having a minimum row-wise sum contains a minimal solution to the cyclic covering problem.

In Example 3.3, $\mathbf{A}_{\text{red}}$ is composed by the rows of $\mathbf{A}$ that correspond to columns of

$$\begin{aligned}\mathbf{PA}' = \mathbf{P}&\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \\ = &\begin{bmatrix} 2 & 1 & 2 & 1 & 1 & 0 & 1 & 0 & 2 & 1 & 2 & 1 & 1 & 0 & 1 & 0 \\ 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 0 & 2 & 1 & 1 & 0 & 2 & 1 & 1 & 0 & 2 & 1 & 1 & 0 \\ 2 & 2 & 1 & 1 & 2 & 2 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}\end{aligned} \tag{3.10}$$

with only non-zero entries, and the row-wise sum of $\mathbf{A}_{\text{red}}$ is computed by

$$\mathbf{A}_{\text{red}}\mathbf{1}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 3 \\ 3 \\ 2 \\ 3 \\ 2 \end{bmatrix}.$$

Since both the fifth and the seventh row of $\mathbf{A}_{\text{red}}$ show the minimum row-wise sum, both the subset $\{P_1, P_4\}$ and the subset $\{P_2, P_3\}$ are minimal solutions to the cyclic covering problem.

In Example 2.2, $\mathbf{A}$ consists of 64 rows, and $\mathbf{A}_{\text{red}}$ of 18 rows. After removing row dominances from $\mathbf{A}_{\text{red}}$,

$$\mathbf{A}_{\text{red}} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

contains exactly the same combinations of prime implicants represented by the conjunctions of the Boolean propositions C_g, $g = 1, \ldots, 6$, in (2.1).

4 Discussion

In this paper, we have presented two algorithms based on matrix algebra for the identification of the prime implicants composed by a logic expression, and the minimization of the set containing these conjunctions.

The former algorithm has been in particularly developed for a situation in which hundreds of logic expressions containing up to 16 binary variables should be converted into a disjunctive normal form (as, e.g., in Schwender and Ickstadt, 2006) consisting of prime implicants, where the prime implicants exhibit typically a maximum order of 5.

This algorithm is limited by m, the number of variables in the logic expression of interest, and the maximum order i of the prime implicants, since a matrix consisting of $2^i \binom{m}{i}$ rows is constructed. A solution to this problem would be to split this matrix in several matrices of smaller sizes, and consider each of these matrices when searching for prime implicants of order i. If the logic expression, however, will contain a large number of variables, and the prime implicants are conjunctions of a very high order, then this solution will also fail and heuristic algorithms for logic minimization such as Espresso (McGeer et al., 1996) have to be used for logic minimization.

A drawback of the second algorithm for the minimization of the set of prime implicants is that it just delivers one minimum disjunctive normal form, even though more than one solution might exist when, e.g., two columns dominate each other (see Section 2.2), or a prime implicant table has to be solved (see, e.g., Section 3.2). This problem can be solved by recording which prime implicant(s) can be replaced by which of the other prime implicants.

Since our main focus is on the identification of all prime implicants this feature has not been implemented yet in our software for the matrix based

logic minimization that is available in the `R` package `logicFS` which can be downloaded from `http://www.bioconductor.org`, the web page of the Bioconductor project (Gentleman et al., 2004).

Acknowledgement

The financial support of the Deutsche Forschungsgemeinschaft (SFB 475, "Reduction of Complexity in Multivariate Data Structures") is gratefully acknowledged.

References

Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K., Hothorn, T., Huber, W., Iacus, S., Irizarry, R., Leisch, F., Li, C., Maechler, M., Rossini, A. J., Sawitzki, G., Smith, C., Smyth, G., Tierney, L., Yang, J. Y .H. and Zhang, J. (2004). Bioconductor: Open Software Development for Computational Biology and Bioinformatics. *Genome Biology*, 5, R80.

Ihaka, R. and Gentleman, R. (1996). R: A Language for Data Analysis and Graphics. *Journal of Computational and Graphical Statistics*, 5, 299–314.

Karnaugh, M. (1953). The Map Method for Synthesis of Combinatorial Logic Circuits. *Transactions of the American Institute of Electrical Engineers, Communications and Electronics*, 72, 1, 593–598.

McCluskey, E. (1956) . Minimization of Boolean Functions. *Bell System Technical Journal*, 35, 1417–1444.

McGeer, P. C., Sanghavi J. V., Brayton, R. K. and Sangiovanni-Vincentelli, A. L. (1993). Espresso-Signature: A new Exact Minimizer for Logic Functions. In: *Proceedings of the* 30^{th} *International Conference on Design Automation*, ACM Press, New York, NY, USA, 618–624.

Petrick, S. R. (1956). A Direct Determination of the Irredundant Forms of a Boolean Function from the Set of Prime Implicants. *Technical Report AFCRC-TR-56-110*, Air Force Cambridge Research Center, Cambridge, MA, USA.

Quine, W. V. (1952). The Problem of Simplify Truth Functions. *American Mathematical Monthly*, 59, 8, 521–531.

Schwender, H. and Ickstadt, K. (2006). Identification of SNP Interactions Using Logic Regression. *Technical Report*, SFB 475, University of Dortmund, Germany.