World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:14, No:1, 2020

Problems of Boolean Reasoning Based Biclustering Parallelization

Marcin Michalak

Abstract—Biclustering is the way of two-dimensional data analysis. For several years it became possible to express such issue in terms of Boolean reasoning, for processing continuous, discrete and binary data. The mathematical backgrounds of such approach — proved ability of induction of exact and inclusion–maximal biclusters fulfilling assumed criteria — are strong advantages of the method. Unfortunately, the core of the method has quite high computational complexity. In the paper the basics of Boolean reasoning approach for biclustering are presented. In such context the problems of computation parallelization are risen.

Keywords—Boolean reasoning, biclustering, parallelization, prime implicant.

I. Introduction

BICLUSTERING is the technique of two–dimensional homogeneous data analysis. It was started by Hartigan in 1970's [1]. The goal of biclustering is to find submatrix (or submatirices) of a given one, whose elements fulfill a sure defined criterion: equality, similarity or even dissimilarity.

The bicluster will be defined as an ordered pair of the subset of rows and the subset of columns. The intersection of mentioned columns and sets points cells that are elements of the bicluster.

One of the novel approaches to biclustering is the Boolean based biclustering. It was originally proposed in [2] for finding exact biclusters in the binary and discrete data and it is successively developed for continuous data [3], [4]. The mentioned approach requires to code the data to be biclustered in the form of Boolean formula, which is in CNF (Conjunction Normal Form), and the formula is later transformed to DNF (Disjuntion Normal Form). Prime implicants of DNF are finally interpreted as inclusion–maximal biclusters fulfilling the assumed criterion of (dis)similarity.

What is important to be mentioned, in general the issue of transforming the CNF to DNF has quite high computational complexity. It is comparable to 3–SAT problem (for finding inclusion–maximal exact biclusters in discrete data) or 2–SAT problem (for finding icnlusion–maximal exact biclusters in binary data).

In this paper the most basic issues that deal with the Boolean transforming are presented. It occurs that simple propositions of calculations parallelization do not bring satisfactory results in such an area.

The paper is organized as follows: it starts with short description biclustering problem, afterwards the Boolean reasoning based approach for biclustering id presented; finally the problems of Boolean reasoning based biclustering parallelization are risen; the paper ends with conclusions and perspectives of further works.

M. Michalak is with the Institute of Informatics, Silesian University of Technology, ul. Akademicka 16, 44-100 Gliwice, Poland (e-mail: Marcin.Michalak@polsl.pl).

II. Related Works

It is very common to mistake the biclustering and clustering of two–dimensional data. The most important difference between such two approaches is the nature of the data: biclustering requires homogeneous data in the whole input data while clustering requires just the homogeneity of rows (or columns). The another difference is the nature of the method results: clustering result is the input set partition in terms of its mathematical definition:

- different elements of partition are disjoint (they have no common elements),
- there is no empty element of partition,
- sum of all elements of partition gives the initial set;

while biclustering result does not have to fulfill such criterion: elements may have common intersection and they do not have to cover all of the input data (it was proved in [2] that an empty bicluster — containing no rows or no columns — still fulfills criteria of exactness and inclusion–maximality).

The difference between clustering and biclustering is explained more precisely in Figs. 1 and 3.

In Fig. 1 we see two possible applications of clustering: we may be interested in object (a)) or feature (b)) clustering. These both results fulfill obviously the criteria of division from the mathematical point of view (presented above). Nevertheless, we can also see that possible results of biclustering of the data brings completely different solutions. Considering the data in of Fig. 2 we may be interested in finding only the exact biclusters (all elements of bicluster have the same value – se Fig. 3 a)) but it is possible to require that the bicluster elements are similar to each other: e.g. their values can not be different by the value 5 (see Fig. 3 b)).

Since Hartigan paper [1] many approaches of bicluster induction were successfully developed. In [5] bicluster induction is performed by minimizing the squares of residuals between the average value of bicluster cells. Founded bicluster is then replaced with random values and the procedure is repeated for the modified data. In [6] such an approach is modified: biclusters may overlap each other and the found bicluster is not replaced by random values.

The other approach was proposed for discretized data [7]. Here, the set of genes that is simultaneously conserved across subset of conditions is the goal of the analysis (that is why the method works only on discretized data). The algorithm starts with random initial sets of rows and columns and proceeds

a)

	f1	f2	f3	f4
o1	1	2	30	40
o2	2	3	31	41
o3	10	11	-5	-8
o4	11	12	-4	-7
o5	101	102	103	104

b)

	f1	f2	f3	f4
o1	31	32	88	91
o2	1	4	87	95
o3	2	4	90	93
o4	50	52	91	94
o5	50	53	90	94

Fig. 1 Possible results of clustering: (a) object clustering; (b) feature clustering

	f_1	f_2	f_3	f_4
o_1	1	1	2	2
o_2	1	1	8	19
o_3	1	32	31	32
o_4	12	18	32	34
o_5	20	27	35	36

Fig. 2 A sample continuous matrix

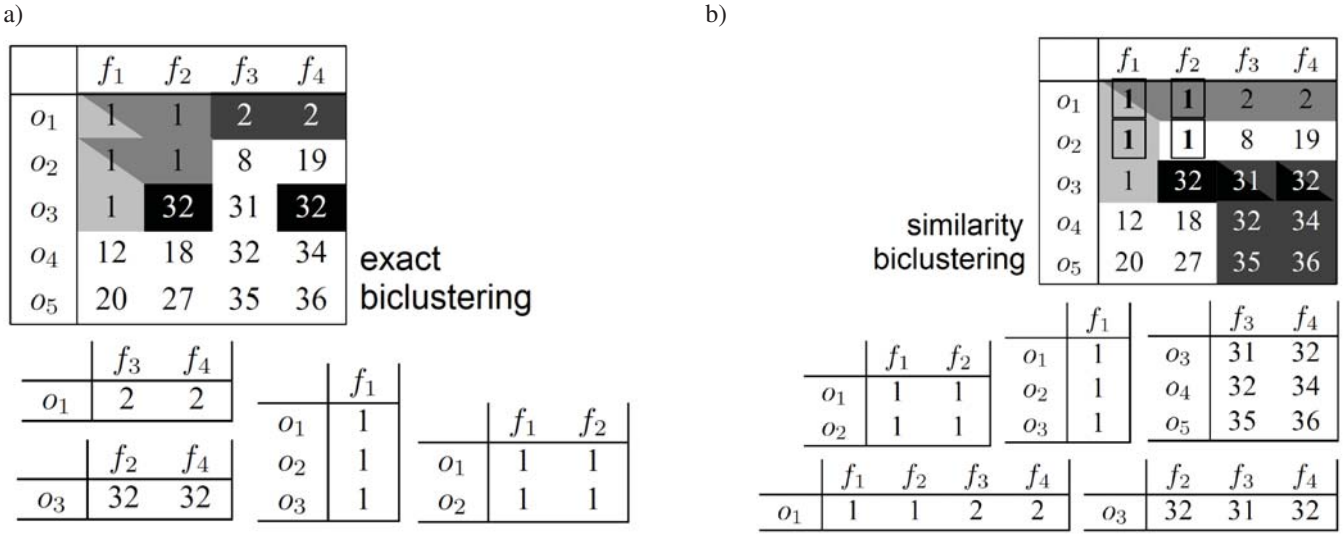

Fig. 3 Possible results of discrete data biclustering: (a) results of exact biclustering; (b) results of similarity biclustering, here: the maximal difference between bicluster elements is not higher than 5; in both cases only non–trivial (not single cell) biclusters are marked

in an iterative way. To assure finding several biclusters, the method should be invoked several times with different initial sets of rows and columns.

BiMax [8] is the method of binary data biclustering. In this approach the separate-and-conquere strategy is applied for finding inclusion–maximal biclusters.

A more detailed description of mentioned algorithms and a presentation of many more biclustering algorithms (such as FABIA [9]) can be found in [10].

It is worth to notice the idea of finding biclusters in the matrix can be also expressed in different data analysis paradigms: in terms of the formal concept analysis, extracting a concept lattice for a given context is equivalent to the problem of finding all inclusion–maximal exact biclusters in a binary matrix [11]; in the domain of the basket analysis, the exact bicluster will correspond to the frequent itemset [12]; if the binary matrix represents some graph contingency matrix the inclusion–maximal bicluster may refer to the clique in the graph or may refer to the two subsets of its vertexes which are directly accessible: each vertex from the first subset is directly accessible from each vertex in the second subset and vice versa.

III. Context of Boolean Reasoning in Biclustering

In this section the short introduction to Boolean reasoning biclustering methods will be presented for the better understanding of a problem of its parallelization. This requires to bring some useful definitions. The set of matrix rows will be denoted as B while the set of columns of matrix will be denoted as X. Then the bicluster may be defined as a such ordered pair $(\{a_1, a_2, \ldots, a_m\}, \{x_1, x_2, \ldots, x_m\})$ such that $\{a_1, a_2, \ldots, a_m\} \subseteq B$ and $\{x_1, x_2, \ldots, x_m\} \subseteq X$.

The Conjunction Normal Form of the formula is the

conjunction of alternatives, e.g.:

$$(a \vee b \vee \ldots x) \wedge (b \vee d \vee \ldots y) \wedge \ldots \wedge (a \vee f \vee \ldots y)$$

while the Disjunction Normal Form of the formula is the disjunction of conjuntions, e.g:

$$c \wedge b \wedge \ldots x \vee c \wedge r \wedge \ldots \wedge y \vee \ldots \vee b \wedge c \wedge \ldots \wedge z$$

A. Discrete Data Biclustering

Let us consider a discrete value matrix as presented in Table I and we are looking for inclusion–maximal exact bicluster. The exactness means that all cells of bicluster have the same value. The inclusion–maximality means that no row or no column can be added to the bicluster without violating the condition of exactness.

TABLE I
A SAMPLE DISCRETE MATRIX M_d

	a	b	c
1	1	0	2
2	1	1	0
3	1	1	1

It is very easy to observe that there is only one bicluster of 2: $(\{1\},\{c\})$, two biclusters of 0: $(\{1\},\{b\}),(\{2\},\{c\})$, and three biclusters of 1: $(\{1,2,3\},\{a\}),(\{2,3\},\{a,b\}),(\{3\},\{a,b,c\})$.

It was proved in [2] that the Boolean function of the form:

$$f = \bigwedge (a \vee b \vee x) \wedge \bigwedge (c \vee y \vee z)$$

where

$$a,b,c \in B \quad x,y,z \in X$$

such that[1]

$$\forall_{a,b,c,x,y,z} \ (a(x) \neq b(x) \wedge a \neq b) \vee (c(y) \neq c(z) \wedge y \neq z)$$

has the following property: each prime implicant of the function codes one inclusion–maximal exact bicluster. Also the opposite implication is true: each inclusion–maximal exact bicluster can be coded by a prime implicant of the formula.

The appropriate formula (in CNF form) for the data from the Table I looks as follows:

$$\begin{aligned} f = \ & (1 \vee a \vee b) \wedge (1 \vee a \vee c) \wedge (1 \vee b \vee c) \wedge \\ & \wedge (2 \vee a \vee c) \wedge (2 \vee b \vee c) \wedge (b \vee 1 \vee 2) \wedge \\ & \wedge (b \vee 1 \vee 3) \wedge (c \vee 1 \vee 2) \wedge (c \vee 1 \vee 3) \wedge (c \vee 2 \vee 3) \end{aligned}$$

When converted to DNF it looks as follows:

$$\begin{aligned} f = \ & (1 \wedge 2) \vee (1 \wedge c) \vee (b \wedge c) \vee (1 \wedge 3 \wedge a \wedge b) \vee \\ & \vee (2 \wedge 3 \wedge a \wedge c) \vee (2 \wedge 3 \wedge a \wedge b) \end{aligned}$$

But how the prime implicants and biclusters correspond to each other? Bicluster contains these rows and columns, whose corresponding literals are not present in the prime implicant (and vice verse). The Table II shows how to decode biclusters from prime implicants.

As we can see all biclusters from the Table II are the same as mentioned in the text in the beginning of the subsection.

[1] $a(x) = x(a) =$ element of a^{th} row and x^{th} column.

TABLE II
DECODING BICLUSTERS FROM THE PRIME IMPLICANTS

prime implicant	missing literals	bicluster
$1 \wedge 2$	$3, a, b, c$	$(\{3\},\{a,b,c\})$
$1 \wedge c$	$2, 3, a, b$	$(\{2,3\},\{a,b\})$
$b \wedge c$	$1, 2, 3, a$	$(\{1,2,3\},\{a\})$
$1 \wedge 3 \wedge a \wedge b$	$2, c$	$(\{2\},\{c\})$
$2 \wedge 3 \wedge a \wedge c$	$1, b$	$(\{1\},\{b\})$
$2 \wedge 3 \wedge a \wedge b$	$1, c$	$(\{1\},\{c\})$

B. Binary Data Biclustering

Due to the high computational complexity of direct search of all biclusters in discrete data the binary simplification was proposed [2]. It is easy to bring the problem of biclustering the n-value discrete matrix to n tasks of binary data biclustering.

Such simplification of the data — only zeros and ones in the matrix — changes the goal of biclustering: we can be interested in finding zeros on the background of ones or ones on the background of zeros.

Let us consider the following binary matrix M_b as in Table III and let us be interested in finding biclusters of ones.

TABLE III
A SAMPLE BINARY MATRIX M_b

	a	b	c
1	1	0	1
2	1	0	0
3	1	1	1

It was also proved in [2] that it is possible with the Boolean formula that codes all cells with value zero. Such a formula looks as follows:

$$f = (1 \vee b) \wedge (2 \vee b) \wedge (2 \vee c)$$

and transformed to DNF:

$$f = (1 \wedge 2) \vee (2 \wedge b) \vee (b \wedge c)$$

The interpretation of prime implicants as inlusion–maximal exact biclusters of ones remains the same as it was in case of discrete biclustering.

In general, let $\square \in \{0,1\}$, then if we are looking for biclusters of $\neg\square \in \{0,1\} \setminus \{\square\}$, the appropriate Boolean function is defined as follows:

$$f_{Mb(\square)} = \bigwedge (a \vee x), \ a \in B, x \in X, a(x) = \square \qquad (1)$$

C. Continuous Data Biclustering

Success of applying Boolean reasoning for biclustering discrete and binary data mobilized to extend such an approach for continuous data. In paper [3] an analogous attempt was defined.

Let us define the $\sigma-$bicluster as the inclusion–maximal submatrix whose elements has the maximal absolute different not higher than σ:

$$\max_{\substack{i,j \in X \\ m,n \in B}} |i(m) - j(n)| \leq \sigma$$

The Boolean formula in this case is defined as follows:

$$f = \bigvee (a \vee b \vee x \vee y)$$

where:

$$a, b \in B; x, y \in X; (a \neq b) \vee (x \neq y)$$

such that:

$$|a(x) - b(y)| > \sigma$$

Let us consider the following matrix of continuous data as presented in Table IV.

TABLE IV
A SAMPLE CONTINUOUS MATRIX M_d

	a	b	c
1	1	2	3
2	2	3	5

The highest difference between cells in the given matrix is 4 and the data have only integer numbers. So the highest sensible range of a bicluster that could be interested is $\sigma = 3$. Only one pair of cells fulfills the above condition: the first pair is in the row 1 and column a while the second one is in the row 2 and column c.. That leads to the boolean formula of the following form:

$$f_3 = (1 \vee a \vee 2 \vee c)$$

That means that we have four biclusters with the maximal difference not exceeding 3: $(\{2\}, \{a, b, c\})$, $(\{1, 2\}, \{b, c\})$, $(\{1\}, \{a, b, c\})$ and $(\{1, 2\}, \{a, b\})$.

IV. PARALLELIZATION ISSUES

The naive algorithm of transforming CNF to DNF may be described with the Algorithm 1.

Such simple strategy provides that in following iterations it is possible to multiply two shortest clauses as multiplication results are stored in the end of the clause.

The multiplication of two clauses is performed in two steps: a proper multplication (line 3) and application of absorption laws (line 4) to simplify the new clause (to remove all clause implicants that are not prime ones). As the number of clauses in CNF decreases the number of conjunctions in clauses increases.

Let us consider the binary matrix presented in Fig. 4.

Fig. 4 A sample binary matrix

It contains only 900 elements (biomedical data dimensions are up to thousands rows and columns), where 592 of them are ones.

The total number of inclusion–maximal exact biclusters of ones is 1618. But it is not the merit of the problem. Let us take a closer look for the Boolean formula and how does it change during converting from CNF to DNF. The original CNF consists of 223 two-literal clauses.

In Fig. 5 the number of implicants in newly created clause is presented as the function of the total number of clauses in the formula.

As we may observe as long as the number of clauses is higher than 112 the newly created clause contains only four implicants and is easy to be simplified. As the number of clauses decreases the number of implicants varies from 4 up to 8 (exceeding 10 several times). When the length of the clause decreases below 30 the rapid grow of the implicants number. Finally, the last clause to be simplified consists of 35 000 of implicants.

When the size of the input data increases the scale of the problem becomes more visible. For the input matrix of 40 rows and four columns a similar chart may be generated. It is presented in Fig. 6.

As previously, most of newly created clauses are rather short and easy to be reduced. But the size of the last one clause to proceed reaches almost 2 000 000.

A. CNF Parallelization

Let us assume that we have n computing threads. One of the possible ways of CNF to DNF transformation parallelization may relay on sending pairs of clauses to threads. This intuitive approach will surely accelerate calculation in the initial part of computation. As long as the number of clauses in formula will be higher than $2n$ all threads will be load with computation. Situation will change when the number of clauses will decrease and will be smaller than $2n$. Gradually, more and more threads will be inactive, because there will be no pair of clauses to be multiplied. Moreover, the remaining threads will be loaded with more and more complex task (as it was presented in Figs. 5 and 6). Finally, in the last two clauses multiplication only the one thread will be loaded while the remaining $n - 1$ threads will be inactive.

B. Multiplication Parallelization

The clause shortening procedure has the square computational complexity (from the number of implicants point of view) as each implicant must be compared with other ones to check, whether first or second of them may be exclude form the clause. This observation brings the suggestion, that it may be helpful to parallel the shortening phase with separate–and–conquare stategy: clause is divided into more or less $\sqrt{n}$ parts and each thread performs one of n multiplications and shortenings and their results merged and shortened again.

Such approach is inappropriate in the initial phase of calculations: as it was defined the original formula consists of two–literal clauses and there is no need of multiplication partitioning.

Algorithm 1 Simple algorithm of CNF to DNF transformation.

```
1: RemoveCoveredClauses(formula) {formula in CNF}
2: while formula.NumberOfClauses > 1 do
3:     newClause = RemoveCoveredClauses(formula.Clauses[0], formula.Clauses[1]); {multiply first two clauses}
4:     newClause = ClauseSimplify(newClause); {absorption laws application}
5:     formula.RemoveClauseAt(0);
6:     formula.RemoveClauseAt(0); {removing first two clauses}
7:     formula.AtClauseAtEnd(newClause); {adding new clause to the end of formula}
8: end while
9: return  formula.Clauses[0]; {returning the first clause that is in DNF}
```


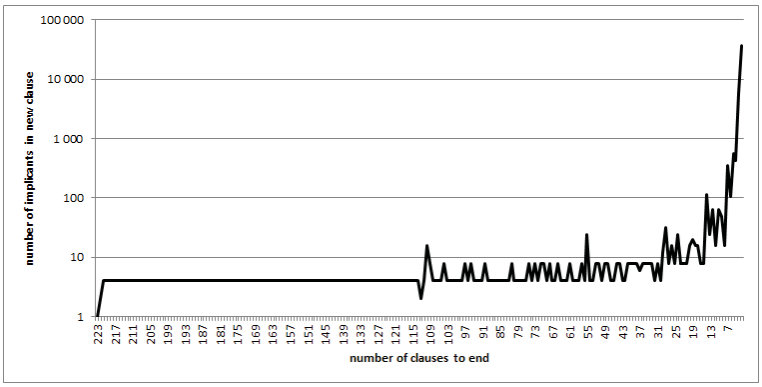

Fig. 5 Number of implicants in newly created clause as the function of total number of clauses for the 900 element matrix

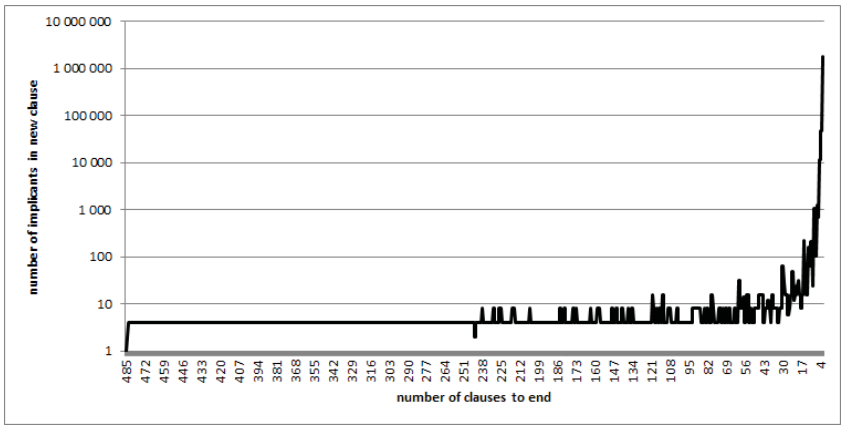

Fig. 6 Number of implicants in newly created clause as the function of total number of clauses for the 1 600 element matrix

C. Perspectives of Data–driven Approaches

The presented two approaches (CNF parallelization anhd multiplication parallelization) solve only one (but different) aspect of CNF to DNF transformation. Te first approach accelerates the initial part of calculation while the second one accelerates the multiplication of long clauses, which appears at the end of calculation.

The intelligent parallelization algorithm should take into consideration the both of mentioned approaches. Moreover, it should also decide whether the frst approach should be stopped and replaced by the second approach, to avoid the situation, where some threads become inactive.

V. Conclusions and Further Works

The promising results of application of Boolean reasoning in the area of biclustering [2], [3] with their theoretical backgrounds and consequences should be considered as the interesting mathematical approach to process the two-dimensional data. From the calculation point of view the problem of bicluster induction is brought to the transforming the formula from its CNF to DNF. Of course it may be also simplified to the problem of finding prime implicants approximations of the original formula. Such works were also performed [13].

However, it still very important to provide the concurrent version of CNF to DNF transformation. As it was explained in the previous section, naive approach of calculation parallelization do not assure the whole computational power usage. The future CNF to DNF transformation algorithms should switch between two mentioned approach intuitively and data and threads dependently. The important element of smart task scheduling should take into consideration the real measures time of computations performed by all threads.

Future works will focus on developing such methods. Moreover, new tasks of biclustering induction in terms of Boolean reasoning will also become the goal of the analysis.

Acknowledgment

The work was carried out within the statutory research project of the Institute of Informatics, BK-204/RAU2/2019.

References

[1] J. A. Hartigan, "Direct Clustering of a Data Matrix," *Journal of the American Statistical Association*, vol. 67, no. 337, pp. 123–129, 1972.

[2] M. Michalak and D. Ślęzak, "Boolean Representation for Exact Biclustering," *Fundamenta Informaticae*, vol. 161, no. 3, pp. 275–297, 2018.

[3] ——, "On Boolean Representation of Continuous Data Biclustering," *Fundamenta Informaticae*, vol. 167, no. 3, pp. 193–217, 2019.

[4] M. Michalak, "Induction of Centre–Based Biclusters in Terms of Boolean Reasoning," *Advances in Intelligent Systems and Computing*, 2019 (to appear).

[5] Y. Cheng and G. Church, "Biclustering of Expression Data," *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, vol. 8, pp. 93–103, 2000.

[6] J. Yang, H. Wang, W. Wang, and P. Yu, "Enhanced Biclustering on Expression Data," in *Proceedings of the Third IEEE Symposium on Bioinformatics and Bioengineering*, 2003, pp. 321–327.

[7] T. M. Murali and S. Kasif, "Extracting Conserved Gene Expression Motifs from Gene Expression Data," in *Proceedings of Pacific Symposium on Biocomputing*, 2003, pp. 77–88.

[8] A. Prelić, S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler, "A Systematic Comparison and Evaluation of Biclustering Methods for Gene Expression Data," *Bioinformatics*, vol. 22, no. 9, pp. 1122–1129, 2006.

[9] S. Hochreiter, U. Bodenhofer, M. Heusel, A. Mayr, A. Mitterecker, A. Kasim, T. Khamiakova, S. Van Sanden, D. Lin, W. Talloen, L. Bijnens, H. W. H. Goehlmann, Z. Shkedy, and D.-A. Clevert, "FABIA: Factor Analysis for Bicluster Acquisition," *Bioinformatics*, vol. 26, no. 12, pp. 1520–1527, 2010.

[10] A. Kasim, Z. Shkedy, S. Kaiser, S. Hochreiter, and W. Talloen, *Applied Biclustering Methods for Big and High Dimensional Data using R*. CRC Press, Taylor & Francis Group, 2016.

[11] D. I. Ignatov and B. W. Watson, "Towards a Unified Taxonomy of Biclustering Methods," in *Russian and South African Workshop on Knowledge Discovery Techniques Based on Formal Concept Analysis*, vol. 1522, 2016, pp. 23–39.

[12] A. Serin and M. Vingron, "DeBi: Discovering Differentially Expressed Biclusters using a Frequent Itemset Approach," *Algorithms for Molecular Biology*, vol. 6, no. 1, 2011.

[13] M. Michalak, R. Jaksik, and D. Ślęzak, "Heuristic Search of Exact Biclusters in Binary Data," *International Journal of Applied Mathematics and Computer Science*, 2019 (to appear).