Int. J. Appl. Math. Comput. Sci., 2020, Vol. 30, No. 1, 161–171
DOI: 10.34768/amcs-2020-0013

HEURISTIC SEARCH OF EXACT BICLUSTERS IN BINARY DATA

MARCIN MICHALAK [a,*], ROMAN JAKSIK [b], DOMINIK ŚLĘZAK [c]

[a]Institute of Informatics
Silesian University of Technology
ul. Akademicka 16, 44-100 Gliwice, Poland
e-mail: Marcin.Michalak@polsl.pl

[b]Institute of Automatic Control
Silesian University of Technology
ul. Akademicka 16, 44-100 Gliwice, Poland
e-mail: Roman.Jaksik@polsl.pl

[c]Institute of Informatics
University of Warsaw
ul. Banacha, 02-097 Warsaw, Poland
e-mail: slezak@mimuw.edu.pl

The biclustering of two-dimensional homogeneous data consists in finding a subset of rows and a subset of columns whose intersection provides a set of cells whose values fulfil a specified condition. Usually it is defined as equality or comparability. One of the presented approaches is based on the model of Boolean reasoning, in which finding biclusters in binary or discrete data comes down to the problem of finding prime implicants of some Boolean function. Due to the high computational complexity of this task, the application of some heuristics should be considered. In the paper, a modification of the well-known Johnson strategy for prime implicant approximation induction is presented, which is necessary for the biclustering problem. The new method is applied to artificial and biomedical datasets.

Keywords: biclustering, Boolean reasoning, prime implicant approximation, biomedical data analysis, Johnson heuristic.

1. Introduction

Finding biclusters in the twodimensional matrix of values is the problem of finding some subsets of rows and subsets of columns, whose intersection defines the region of similar cells, called the bicluster. It depends on the nature of the input data and the goal of the analysis whether all cells from the bicluster should have the same value (exact biclusters) or some level of differences between them is allowed. It is usually expected that the bicluster should be inclusion-maximal, which means that adding any new row (or column) to the bicluster violates the constraint mentioned above.

Amongst many approaches to find biclusters, a new one has been developed, with strong mathematical background and involving the Boolean reasoning paradigm. In our previous research it was defined and proved that Boolean representation of discrete (or binary) data may be successfully used to find inclusion-maximal exact biclusters (Michalak and Ślęzak, 2018). This approach was further generalized for matrices with continuous values (Michalak and Ślęzak, 2019) for finding biclusters of similarity (the difference between any two cells does not exceed the assumed level of similarity: σ) and even biclusters of dissimilarity (the difference between any two different cells is not smaller than the assumed level of dissimilarity: χ). The high computational complexity of such an approach to finding all and inclusion-maximal biclusters implies the development of some faster heuristics for finding a smaller number of biclusters but having the same properties.

*Corresponding author

In the paper, a new heuristic approach to find exact biclusters in binary data is presented. It is based on two assumptions: firstly, binary data are coded with a Boolean function and biclusters are coded with its prime implicants, and secondly, covering all data with biclusters is performed with the sequential covering strategy: the bicluster found is removed from the data and a new bicluster in the remaining data is the second of the search.

The paper is organized as follows: it starts with a short description of the biclustering problem; then the background of Boolean reasoning in biclustering is presented; afterwards, the Johnson heuristic of prime implicant search is described, together with its new modification (for the problem of searching for prime implicants that code exact biclusters in binary data); finally, the experiments and their results for artificial and biomedical data are presented and discussed; the paper ends with some conclusions and perspectives of future works in this area.

2. Related works

Biclustering is a technique of two-dimensional data analysis. It was started by Hartigan in the 1970s (Hartigan, 1972). The goal of the analysis of scalar data in two dimensions was continuously presented on the example of voting results from American presidential elections in different states.

Since then biclustering has been successfully applied in many areas, such as text mining (Chagoyen *et al.*, 2006; Orzechowski and Boryczko, 2016), gene expression analysis (Tanay *et al.*, 2005), data explorations (Latkowski, 2003) and many more (Busygin *et al.*, 2008). It has also received several equivalent names like two-mode clustering, co-clustering or two-dimensional clustering.

Biclustering is commonly mistaken for the clustering of two-dimensional data. But in such a case, the clustering algorithm (regardless of clustering rows or clustering columns) introduces the partition (of a set of rows or a set of columns accordingly) in the sense of its mathematical definition. The domains of rows and columns may differ (the data are heterogeneous). In the case of biclustering we are looking for some regions in homogeneous data that are more similar to each other than to the other elements in the data. The difference between clustering and biclustering is explained more precisely in Figs. 1 and 2.

In Fig. 1 the results of clustering objects (left) and clustering features (right) are presented. In both cases the operation resulted in introducing a partition in the mathematical sense. In the centre of Fig. 2 a sample matrix of discrete data is presented, whose data were biclustered in two ways: exact biclustering means finding inclusion-maximal biclusters of cells with the same value while similarity biclustering means finding an inclusion-maximal bicluster of cells of such values, whose maximal difference does not exceed an assumed level (here: level $= 5$). The biclusters found are presented below and also marked with different shades in the matrix.

All biclusters presented above fulfilled the condition of being non-extendable (inclusion-maximal). That means that any row or any column could be added to the bicluster without breaking the restriction set to the bicluster cells values. If we define the bicluster as an ordered pair of a subset of rows and a subset of columns, the nonextendability can be defined with the inclusion relation. A non-extendable bicluster is defined with a subset of rows and a subset of columns that are maximal in the sense of inclusion. Accordingly, this kind of biclusters will be named maximal in the sense of inclusion (in short: inclusion-maximal).

Many approaches to bicluster induction have been successfully developed. In the paper of Cheng and Church (2000) a bicluster is searched by minimizing the squares of residuals between the average value of bicluster cells. In the original approach the bicluster found is later replaced with random data and the procedure is repeated for the modified data. In the modified version of this algorithm FLOC Yang *et al.* (2003) provide a solution that allows biclusters to overlap each other, and when a bicluster is found it is not replaced by random values.

For discretized data, Murali and Kasif (2003) developed the xMotif algorithm. In this approach the set of genes that is simultaneously conserved across a subset of conditions is the goal of the analysis. This means that this approach requires the initial discretization of continuous data. The algorithm starts with random initial sets of rows and columns and proceeds in an iterative way. To assure finding several biclusters, the method should be invoked several times with different initial sets of rows and columns.

One of specific approaches for finding biclusters is represented, for example, by the BiMax algorithm (Prelić *et al.*, 2006), where data to be biclustered are binary. In this approach the separate-and-conquer strategy is applied for finding inclusion-maximal biclusters.

A more detailed description of the algorithms mentioned and a presentation of many more biclustering algorithms (such as FABIA (Hochreiter *et al.*, 2010)) can be found in the work of Kasim *et al.* (2016).

However, the issue of finding biclusters in data can be also defined 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 (Ignatov and Watson, 2016); in the domain of the basket analysis, the exact bicluster will correspond to the frequent itemset (Serin and Vingron, 2011); if the binary matrix represents some graph contingency matrix

	f_1	f_2	f_3	f_4
o_1	1	2	30	40
o_2	2	3	31	41
o_3	10	11	-5	-8
o_4	11	12	-4	-7
o_5	101	102	103	104

	f_1	f_2	f_3	f_4
o_1	31	32	88	91
o_2	1	4	87	95
o_3	2	4	90	93
o_4	50	52	91	94
o_5	50	53	90	94

Fig. 1. Possible results of object (left) and feature (right) clustering.

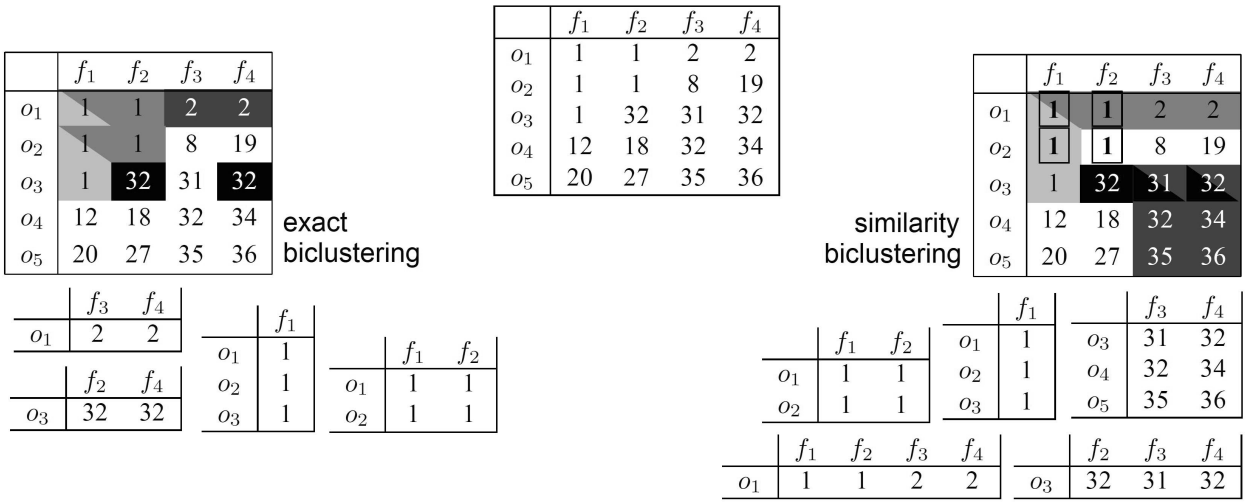

Fig. 2. Possible results of discrete data biclustering: the original data are presented in the centre, results of exact biclustering are presented on the left-hand side, while results of similarity based biclustering (here: the maximal difference between bicluster elements is not higher than 5) are presented on the right-hand side; in both the cases only non-trivial (not single cell) biclusters are mentioned.

the inclusion-maximal bicluster may refer to the clique in the graph or 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.

3. Context of Boolean reasoning based biclustering

Similarly to many other data analysis tasks, such as reducts finding, bireducts finding and many more, the issue of searching biclusters in binary data can be described in terms of Boolean reasoning. This means that the problem can be coded as a Boolean function and the goal may be interpreted from prime implicants of this formula.

Let us consider the binary matrix presented in Table 1. Let us also define the problem of finding all inclusion-maximal exact biclusters, covering ones in the presented matrix. As the bicluster, the ordered pair of a subset of rows and a subset of columns is considered.

The solution of the presented problem can be found with coding all zero-cells of the matrix as the Boolean formula in such a way that it is the conjunction of

Table 1. Sample binary matrix Mb.

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

two-literal clauses, whose variables correspond to the following zero-cells indexes (row and column).

In the presented example we have three cells with zero (with coordinates $(1, b), (2, b)$ and $(2, c)$), which implies the following Boolean formula $f_{Mb(0)}$:

$$f_{Mb(0)} = (1 + b)(2 + b)(2 + c).$$

(From the clarity viewpoint we do not introduce separate notation for Boolean variables: the meaning of "1" depends whether it is used in the formula (literal) or in the bicluster (row index)). The same formula presented in the form of prime implicants would be as follows:

$$f_{Mb(0)} = 12 + 2b + bc,$$

but we remember that 1 and 2 are literals, not digits, so the first prime implicant is a two-literal one.

The interpretation of the obtained prime implicants is as follows: each prime implicant represents one exact and inclusion-maximal bicluster of ones in the original data in such a way that the prime implicant consists of only those literals whose corresponding rows and columns are not elements of the bicluster.

Let us take a closer look at prime implicants of the function $f_{Mb(0)}$: the first implicant consists of literals 1 and 2 while the matrix Mb has three rows $\{1, 2, 3\}$ and three columns $\{a, b, c\}$, so the bicluster coded with the implicant 12 has the form $(\{3\}, \{a, b, c\})$. The graphical visualization of all three biclusters coded by three prime implicants of the formula $f_{Mb(0)}$ is presented in Fig. 3.

The same steps may be performed to find biclusters of zeros in the data. The only difference is in the Boolean function construction: instead of coding cells of zeros, cells of ones are coded in the formula $f_{Mb(1)}$, whose prime implicants will then correspond to inclusion-maximal exact biclusters of zeros in the data.

The example described above joins prime implicants of a Boolean function and exact, inclusion-maximal biclusters in the data, from which the Boolean function was derived. The correctness of such an approach to bicluster finding was proved by Michalak and Ślęzak (2018) in two aspects: correctness and maximality. In the mentioned paper two theorems of binary biclustering were presented. The weaker one says that for each implicant of the Boolean formula there exists an exact bicluster in the data as the complement of this implicant (correctness). The stronger one says that for each prime implicant of the Boolean formula there exists an inclusion-maximal exact bicluster in the data (maximality).

The exhaustive strategy that assumes finding all inclusion-maximal exact biclusters to cover the data implies high computational complexity of the problem. Even if it is the problem of just 2-CNF formulas satisfiability, which is P-complete, the goal is still time consuming in terms of calculations. These theorems imply that the application of the heuristics of prime implicants approximations should become interesting from the point of view of exact bicluster searching: as long the heuristics find implicants of the Boolean function, the covering strategy of finding biclusters will cover all required cells in the data with exact non-overlapping biclusters.

4. Heuristics of bicluster searching

The issue of finding biclusters in the binary matrix boils down to the problem of finding all prime implicants in the Boolean formula derived from the data. Due to the high computational complexity of this problem, the climbing strategy of sequential data covering may be proposed. Let us consider the following algorithm: as long as there are uncovered cells in the data to be covered by the bicluster, find the one implicant of the Boolean function and move the corresponding bicluster to the result set; then mark all cells of this bicluster as covered and generate the Boolean function for the modified data. The presented approach requires the algorithm of finding one prime implicant of the Boolean function.

Instead of the one prime implicant we may take into consideration some heuristics for finding the logical statement that would be implicant but not the prime one. As proved by Michalak and Ślęzak (2018), each prime implicant of such a defined Boolean function for the binary data biclustering corresponds to the exact bicluster.

In this section the classic problem of finding prime implicants of the Boolean formula with the Johnson heuristic will be presented, with a short discussion why it cannot be directly applied for the purpose of exact bicluster searching. Afterwards, a suitable modification will be presented.

4.1. Classic Johnson approach. One of the most popular and simple heuristics of the prime implicant searching is Johnson's approach (Johnson, 1974). For a given formula f in the CNF, the set of literals that is the approximation of the prime implicant is found on the basis of the literals in the clauses occurrence frequency.

Consider the following Boolean formula in the CNF:

$$f = (a+b+c+e)(b+c+d)(e+f+g)(d+e+f)(h+i).$$

When we bring it to the DNF, we will obtain a formula built from 20 prime implicants. The Johnson heuristic searches for the most frequent literal in formula clauses and moves it to the result set of variables. Then, all clauses containing the literal found are removed from the formula and the next step of the search is performed. Iterations stop when the formula is empty (all clauses are covered by at least one literal from the result set).

The frequency of literals for the given formula is presented in Table 2.

The most frequent literal is e. Accordingly, the new formula, after deleting clauses containing e, has the following form (the set $\{\ldots\}$ in the lower index of f points literals in the current result set—the implicant approximation):

$$f'_{\{e\}} = \cancel{(a+b+c+e)}(b+c+d)\cancel{(e+f+g)}\cancel{(d+e+f)} \times (h+i) = (b+c+d)(h+i).$$

Table 2. Frequency of literals in the formula f.

literals	a	b	c	d	**e**	f	g	h	i
occurrences	1	2	2	2	**3**	2	1	1	1

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

$12 : (\{3\}, \{a, b, c\})$

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

$2b : (\{1, 3\}, \{a, c\})$

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

$bc : (\{1, 2, 3\}, \{a\})$

Fig. 3. Visualization of three exact and inclusion-maximal biclusters in matrix Mb, corresponding to prime implicants of the function $f_{Mb(0)}$.

With the next iteration of finding the most frequent literal in the clauses, we may find b, which leads to the formula

$$f''_{\{e,b\}} = \cancel{(b + e + d)}(h + i) = (h + i),$$

and at last the h as the most frequent one:

$$f'''_{\{e,b,h\}} = \emptyset.$$

Thus, finally, it appears that the set $\{e, b, f\}$ is the Johnson approximation of the prime implicant of the formula f. This means that the set is the superset for at least one of the real prime implicants.

4.2. Modified Johnson approach. One may suspect that the heuristic presented above would be sufficient to be applied to the problem of finding approximate prime implicants for the Boolean function coding biclusters in the binary data. Just a simple example will show that the original Johnson algorithm requires modifications.

Consider the binary matrix M_b, presented in Table 3, with the goal of finding biclusters of ones in the data. Remembering that the Boolean function for biclusters of ones is built from the clauses that code cells with zero, the proper formula looks as follows:

$$\begin{aligned} f &= (1 + a)(1 + b)(1 + e)(1 + f)(1 + g)(1 + h) \\ &\quad \times (2 + a)(2 + d)(2 + e)(2 + f)(2 + g)(2 + h) \\ &\quad \times (3 + a)(3 + c)(3 + d)(3 + e)(3 + f)(3 + g)(3 + h). \end{aligned}$$

Frequencies of literals of the formula f are presented in Table 4.

The most frequent literal is 3 (it appears seven times in the formula). Removing all clauses covered by the literal 3 leads to the following form of the formula:

$$\begin{aligned} f' &= (1 + a)(1 + b)(1 + e)(1 + f)(1 + g)(1 + h) \\ &\quad \times (2 + a)(2 + d)(2 + e)(2 + f)(2 + g)(2 + h). \end{aligned}$$

Now it is easy to observe that the most frequent literal for the formula f' would be 1 (or 2: this would depend on the algorithm of finding the most frequent item in the collection), and after removing the first (second) row of the matrix the most frequent literal in the new Boolean function would be 2 (respectively, 1). Thus, finally, the original Johnson strategy will point the set of variables $\{1, 2, 3\}$ as the approximation of the prime implicant of the Boolean function for the matrix M_b. Despite the fact that the result fulfils the criterion of being the approximation of some prime implicant of the Boolean function for the matrix M_b, it is a very disappointing result in the context of binary biclustering, as it defines an empty bicluster—the one corresponding to no rows and some columns or some rows but no columns. This simple observation leads us to a modification of the Johnson strategy of finding the prime implicant approximation in the context of binary biclustering. The main problem of the original solution is that the prime implicant approximation is built without any distinction between Boolean variables corresponding to rows and Boolean variables corresponding to columns.

An intuitive modification of the original Johnson approach consists in building the prime implicant approximation on the rows and columns simultaneously. In each iteration the pair of most frequent literals coding rows and columns should be taken into consideration.

Let us go back to the matrix M_b. The most frequent row-based literal is still 3, while the most frequent column-based one is b, so the pair $\{3, b\}$ is moved to the final prime implicant approximation set. The matrix corresponding to the formula in the next iteration is presented in Table 5.

Table 3. Binary matrix M_b.

	a	b	c	d	e	f	g	h
1	0	0	1	1	0	0	0	0
2	0	1	1	0	0	0	0	0
3	0	1	0	0	0	0	0	0

Table 4. Frequency of literals in the formula f.

literals	a	b, c	d	e, f, g, h	1, 2	**3**
occurrences	3	1	2	3	6	**7**

Table 5. Binary matrix M'_b.

	a	c	d	e	f	g	h
1	0	1	1	0	0	0	0
2	0	1	0	0	0	0	0

After this step the pair of the most frequent rows and columns literals is $(2, c)$. As the last pair of frequent literals $(1, c)$ should be taken into consideration. But adding this pair of variables would also affect the empty bicluster, as all rows are represented in the prime implicant approximation. But at this moment we should note that this addition will be effective in generating another empty bicluster, as literals coding all rows would be present in the resulting set. Therefore, the final strategy of finding the prime implicant approximation in the formula coding the background cells should be defined as in Algorithm 1.

The pseudocode of three methods used in Algorithm 1 is not supplied, but their names explain most of their meaning: TheMostFrequentColumnLiteral() returns the column coding literal that appears most often in the formula, which is the function argument; TheMostFrequentRowLiteral() returns the most frequent row coding literal analogously; RemoveCoveredClauses() removes clauses from the formula, covered by literals from the list of literals coding rows (tRows) and literals from the list of literals coding columns (tColumns).

4.3. Sequential biclustering heuristic. Solving the problem of finding the approximate prime implicant of the Boolean function does not solve the general problem of Boolean reasoning about the bicluster, which consists in finding all prime implicants of the formula to cover all the data in the matrix. Having the modified Johnson strategy of finding one prime implicant for the formula, we can provide the heuristic of sequential covering of all the data in the matrix.

Let us start with the original formula that codes the problem of finding biclusters of ones (with coding all cells with zero in the formula). As proved by Michalak and Ślęzak (2018), each implicant of this formula (not necessarily the prime one) codes one exact bicluster. The bicluster found with the presented strategy describes a part of the data in the matrix. Then we may remove the data covered by this bicluster in such a way that these cells will be changed from 1 to 0. The new matrix contains only the ones that are not covered yet. Therefore, in another iteration, a new formula is built and its implicant approximation is found. The procedure is applied as long as there are any ones in the matrix. The pseudocode of this procedure is presented as Algorithm 2.

The meanings of three methods used in the Algorithm 2 are as follows: BuildFormula() codes all background cells (zeros in the case of searching for biclusters of ones and vice versa) as presented in Section 3; in the fifth line Algorithm 1 is invoked; RemoveOnesCoveredByBicluster() replaces ones (or zeros) in the matrix with zeros (ones) covered by a newly found bicluster.

4.4. Complexity study. Consider a matrix with r rows, c columns and w ones. Let us also define the following variables:

- $l = \min(r, c)$: the lower bound of matrix dimensions;
- $L = \max(r, c)$: the upper bound of matrix dimensions;
- $|f| = r \times c - w$: the length (number of clauses) of formula f.

The loop in the 4th line of Algorithm 1 may be executed up to $|f|$ times. Each iteration consists of two sortings (row literals and column literals separately), which gives the $r \log r + c \log c$ addend and Boolean formula shortening, with the complexity of its length: $r \times c - w$. In the most pessimistic case, Algorithm 2 may invoke Algorithm 1 in the 5th line up to $r \times c - w$ times. This finally gives us the complexity of the algorithm as follows:

$$O((r \times c - w)(r \log r + c \log c)).$$

If n means the final size of the data $n = r \times c$, the approximation of the method can be expressed as

$$O((n - w)(n \log n)).$$

This suggests that in the most pessimistic case the method has the complexity of $O(n^2 \log n)$, but it decreases significantly as the matrix has more cells to be biclustered (up to $n \log n$, approximately).

4.5. Discussion of results quality. In Section 3 the theorem from the paper by Michalak and Ślęzak (2018) was recalled. The theorem says that each implicant of such an earlier defined Boolean formula corresponds to the exact bicluster in the data. This means that each bicluster found by the FindSingleBicluster() function must be an exact bicluster, because this function builds the implicant of the Boolean function.

Considering Algorithm 2, it must be stated that it provides a set of biclusters generated by Algorithm 1 and all of them are exact (as already proved). The loop in line 3 of this algorithm also must finish as each iteration covers at least one cell in the data, so in each iteration the number of ones in the matrix decreases (ones covered by the newly generated bicluster are replaced by zeros).

Algorithm 1. Modified Johnson algorithm of prime implicant approximation searching for formulas coding biclusters.

```
1: function FindSingleBicluster(formula)
2: columnLiterals := ∅ {the set of column literals (initially empty)}
3: rowLiterals := ∅ {the set of row literals (initially empty)}
4: while formula ≠ ∅ do
5:    {find the most frequent literal coding row and column in the formula}
6:    mostFrequentColumn := TheMostFrequentColumnLiteral(formula)
7:    mostFrequentRow := TheMostFrequentRowLiteral(formula)
8:    tColumns := columnLiterals ∪ mostFrequentColumn
9:    tRows := rowLiterals ∪ mostFrequentRow
10:   if (tColumns == allColumns) or (tRows == allRows) then
11:      {adding the pair will lead to the empty bicluster}
12:      tRows := all but one last rows not covered by the rowLiterals
13:      tColumns := literals coding clauses not covered by literals from tRows
14:      formula := ∅
15:   else
16:      {adding the pair is safe}
17:      rowLiterals := rowLiterals ∪ tRows
18:      columnLiterals := columnLiterals ∪ tColumns
19:      {remove clauses from the formula, covered by the pair of mostFrequentRow and mostFrequentColumn literals}
20:      formula := RemoveCoveredClauses(formula, tRows, tColumns)
21:   end if
22: end while
23: return columnLiterals ∪ rowLiterals
```

Algorithm 2. Heuristic of sequential biclustering.

```
1: biclusters := ∅ {the set of generated biclusters (initially empty)}
2: {count the number of ones in matrix}
3: while numberOfOnesInMatrix > 0 do
4:    formula := BuildFormula(matrix)
5:    bicluster := FindSingleBicluster(formula)
6:    matrix := RemoveOnesCoveredByBicluster(matrix, bicluster)
7:    biclusters := biclusters ∪ { bicluster }
8:    {update the number of ones in matrix}
9: end while
10: return biclusters
```

Concluding, it is not possible to obtain wrong results such as a non-exact bicluster (zeros in the bicluster of ones or vice versa) or having some cells of ones in the data non-covered by any bicluster.

5. Experiments

Experiments were performed on artificial and real data, using self-developed software. The first experiment was planned to show the ability of reducing the computational complexity of the problem with the heuristic approach. The goal of the second experiment was to show the real application of the methodology.

5.1. Artificial data. The artificial data contain 100 rows and 100 columns (presented in Fig. 4), and four discrete values are observed in the data: 0, 77, 237 and 255. The 255 value is considered the background while the remaining three values are intended to be covered by biclusters. To apply the strategy of binary biclustering, the binarization of the original data should be done. Therefore, for the final experiments three datasets were used: #0, #77 and #237 (also presented in Fig. 4).

Table 6 presents the results of biclustering. For each of the three datasets, simple information of the input data is shown (the number of ones in the data and the number of clauses in the original formula). The last two columns provide the number of biclusters found in an exhaustive way and with the application of the modified Johnson strategy.

As expected, the total number of exact biclusters, covering ones in each of the matrices considered,

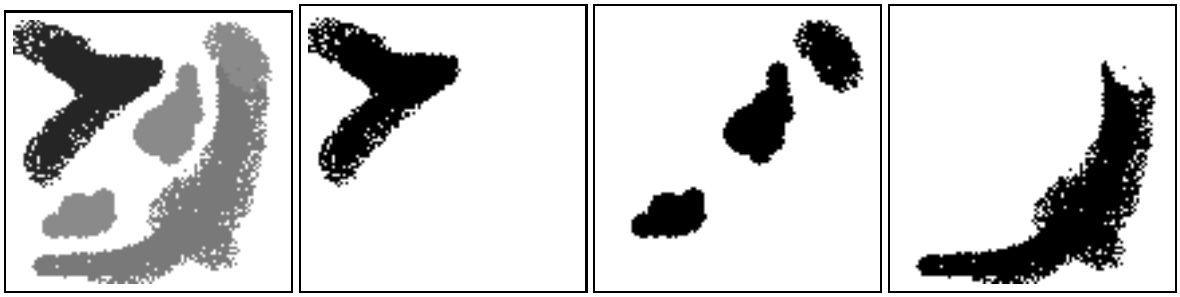

Fig. 4. Artificial data of 100 rows and columns (left) and the following binarized data matrices: # 0 (left center), # 77 (right center) and #237 (right).

Table 6. Comparison of the exhaustive and heuristic approach to biclustering.

data	number of			
	ones	clauses	exhaustive biclusters	Johnson biclusters
#0	1 415	2 256	5 463 (+2 empty)	224
#77	1 327	4 560	503 (+2 empty)	129
#237	2 148	5 267	30 194 (+2 empty)	201

decreased compared with the total number of inclusion-maximal exact biclusters generated in an exhaustive way. The total computation time also decreased significantly: the exhaustive computation time was in weeks (up to 8 weeks for the #237 matrix), while computations for the modified Johnson strategy took minutes (hours). However, the quality of the obtained results cannot be directly compared. In the first case the result contains all exact and inclusion-maximal biclusters (which may overlap themselves) covering all the data while in the second case the result contains only exact biclusters (which do not overlap) covering all the data.

5.2. Comparison with other methods. The new heuristics was compared with several methods of binary biclustering: BiMax (Prelić *et al.*, 2006), Boolean reasoning (Michalak and Ślęzak, 2018), whose results are identical when empty biclusters are removed from BR results, iBBiG (Gusenleitner *et al.*, 2012), and BiBit (Rodriguez-Baena *et al.*, 2011). The experiments with the latter two methods were performed in the R (R Studio) environment. The results of comparison are presented in Table 7. The first column denotes the label of the dataset, the next one contains the name of the method. In the third column the total number of biclusters found is provided. The last two columns describe the complexity and redundancy of data description. The total coverage means the fraction of ones in the binary matrix covered by at least one bicluster from the resulting set. The maximal possible value is one and any lower value reflects the situation when there is at least one 1 in the matrix that is not covered by any bicluster. The measure called the overlapping level reflects how many ones in the data are covered more than once by biclusters. For the matrix with 25 ones and the set of biclusters of a total area 80, the overlapping level equals 2.2, which comes from $(80 - 25)/25$. The interpretation says that each cell is covered by 2.2 additional (redundant) biclusters. The overlapping level equals 0, which means that there is no cell covered by two biclusters.

The first two methods generate the highest number of biclusters because they find all inclusion-maximal bilusters. This implies also the maximal total coverage (equal to one) and the highest overlapping level. The next two methods (iBBiG and BiBit) provided smaller sets of biclusters, which covered the data almost entirely (up to ten ones in the data were not covered). However, the results of iBBiG are not exact biclusters (the accuracy of biclusters was from the range $[.75, 1]$), while the results of BiBit are characterized by quite high redundancy (overlapping level).

The above comparison shows that the new presented approach has properties that are not provided by other comparable methods. It keeps the maximal total coverage like BiMax or the exhaustive Boolean reasoning approach without the effect of overlapped biclusters (like BiBit). The next subsection presents the method application on real data, whose results are significant from the biomedical point of view.

5.3. Biomedical data. Data obtained using three various methods were downloaded from The Cancer Genome Atlas (TCGA) database for 2,088 patients diagnosed with either thyroid, breast or prostate cancer. For each patient we obtained the copy number variation, methylation and gene expression data, which were

Table 7. Comparison of the results of different biclustering approaches.

data	method	# biclusters	total coverage	overlapping level
#0	BiMax / Boolean reasoning	5 463	1.0000	453.81
	iBBiG	45	0.9964	0.61
	BiBit	849	0.9985	78.61
	modif. Johnson strat.	224	1.0000	0.00
#77	BiMax / Boolean reasoning	503	1.0000	34.80
	iBBiG	45	0.9925	0.28
	BiBit	302	0.9977	23.11
	modif. Johnson strat.	129	1.0000	0.00
#237	BiMax / Boolean reasoning	30 194	1.0000	3246.28
	iBBiG	45	0.9995	0.56
	BiBit	1 287	0.9990	94.20
	modif. Johnson strat.	201	1.0000	0.00

further converted into the binary format, for each method separately, where zero represents normal state and one—abnormalities associated with an individual gene. Only 7,018 genes associated with known cancer processes were retained for further study, resulting in three 7,018×2,088 binary tables. In the preprocessing step these binary data were sorted horizontally and vertically, decreasingly according to the number of ones in rows and columns. Going further, their size was limited into 1,000 rows and 1,000 columns to select only the most important biclusters in the data. These three datasets are presented in Fig. 5.

For the given data only the heuristic strategy of finding all covering biclusters was applied. The raw results for each dataset are presented in Table 8.

However, the number of biclusters found is not as important as the meaning of the largest ones. The main goal of this biomedical datasets analysis was to find interesting, interpretable and wide biclusters. Let us consider patients from the largest bicluster found in the CNV data: it contained 24 patients and 24 genes. It appears that the patients from the bicluster are characterized with a completely different survival curve than the remaining ones. The comparison of these two survival curves is presented in Fig. 6.

The gray line represents 24 patients that belong to the selected cluster, which is associated with abnormalities in the copy number of 24 specific genes. The survival times of the patients from this cluster are much shorter (with p-value < 0.0001), emphasizing the significance of those genes for cancer-related processes.

A low p-value indicates that it is very unlikely to select a random set of 24 patients that would show such significantly different survival time, compared with the remaining cases used in the study. Additionally, some of the genes which form the identified cluster are already known to be associated with cancer, including SOX7, ESCO2 and IL24, recognized for their tumor suppressor properties, which further increases the credibility of this finding.

Table 8. Number of biclusters found in each dataset and their subset which shows significant association with patients survival data (adjusted p-value< 0.01).

data	number of biclusters	
	all	$p < 0.01$
CNV	2 300	21
EXP	8 411	138
METH	1 738	18

6. Conclusions and further work

In the paper, a novel approach to exact biclustering in binary matrices was proposed. It is based on the paradigm of Boolean reasoning. The high computational complexity of finding all exact and inclusion-maximal biclusters implied the application of some heuristic to find biclusters covering all the required cells in the matrix. It appeared that one of the most popular heuristics for finding the approximation of the prime implicant of the Boolean formula cannot be applied in this case directly. The paper presents the necessary modification of the procedure on some boundary conditions.

The new biclustering algorithm was successfully applied to artificial and biomedical data. It was shown on the artificial data that total computational time of covering all the data decreases by several orders of magnitude and the total number of resulting biclusters also decreases. It was also shown on the biomedical data that large biclusters can be interpreted from the point of view of survival time analysis of patients.

The presented results give a wide variety of further algorithm development possibilities, starting with other heuristics analysis and modifications, through further modifications of the Johnson approach (e.g., for finding

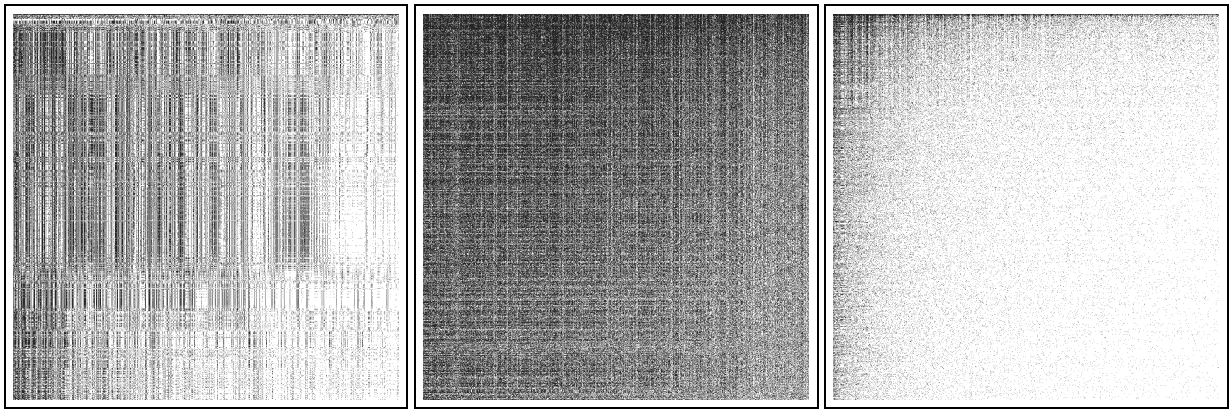

Fig. 5. Three biomedical datasets: CNV (left), EXP (center) and METH (right).

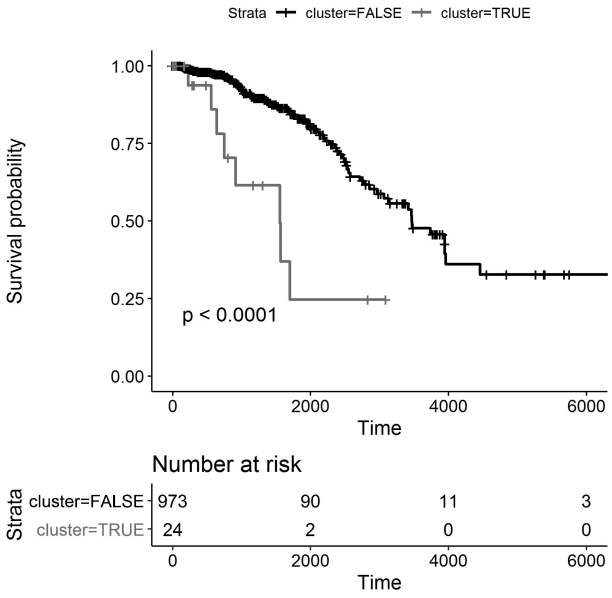

Fig. 6. Comparison of survival curves of two groups of patients: from the bicluster (gray) and the remaining ones (black).

just the top of the largest biclusters in the data), ending with considering the possibility of controlling the proportions between the number of rows and columns in the bicluster.

The promising results of prime implicant approximation for exact biclustering of binary data suggest that such an approach should also be applied to biclustering continuous data.

Acknowledgment

The work was partially carried out within the statutory research project of the Institute of Informatics (Silesian University of Technology) BK-204/RAU2/2019 (Marcin Michalak) and the Polish National Science Center grant 2016/23/D/ST7/03665 (Roman Jaksik).

References

Busygin, S., Prokopyev, O. and Pardalos, P.M. (2008). Biclustering in data mining, *Computers & Operations Research* **35**(9): 2964–2987.

Chagoyen, M., Carmona-Saez, P., Shatkay, H., Carazo, J.M. and Pascual-Montano, A. (2006). Discovering semantic features in the literature: A foundation for building functional associations, *BMC Bioinformatics* **7**, Article no. 41.

Cheng, Y. and Church, G. (2000). Biclustering of expression data, *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology, San Diego, CA, USA*, Vol. 8, pp. 93–103.

Gusenleitner, D., Howe, E.A., Bentink, S., Quackenbush, J. and Culhane, A.C. (2012). iBBiG: Iterative binary bi-clustering of gene sets, *Bioinformatics* **28**(19): 2484–2492.

Hartigan, J.A. (1972). Direct clustering of a data matrix, *Journal of the American Statistical Association* **67**(337): 123–129.

Hochreiter, S., Bodenhofer, U., Heusel, M., Mayr, A., Mitterecker, A., Kasim, A., Khamiakova, T., Van Sanden, S., Lin, D., Talloen, W., Bijnens, L., Goehlmann, H. W.H., Shkedy, Z. and Clevert, D.-A. (2010). FABIA: Factor analysis for bicluster acquisition, *Bioinformatics* **26**(12): 1520–1527.

Ignatov, D.I. and Watson, B.W. (2016). Towards a unified taxonomy of biclustering methods, *Russian and South African Workshop on Knowledge Discovery Techniques Based on Formal Concept Analysis, Stellenbosch, South Africa*, Vol. 1522, pp. 23–39.

Johnson, D. (1974). Approximation algorithms for combinational problems, *Journal of Computer and System Sciences* **9**(3): 256–278.

Kasim, A., Shkedy, Z., Kaiser, S., Hochreiter, S. and Talloen, W. (2016). *Applied Biclustering Methods for Big and High Dimensional Data Using R*, CRC Press, Taylor & Francis Group, New York, NY.

Latkowski, R. (2003). On decomposition for incomplete data, *Fundamenta Informaticae* **54**(1): 1–16.

Michalak, M. and Ślęzak, D. (2018). Boolean representation for exact biclustering, *Fundamenta Informaticae* **161**(3): 275–297.

Michalak, M. and Ślęzak, D. (2019). On Boolean representation of continuous data biclustering, *Fundamenta Informaticae* **167**(3): 193–217.

Murali, T.M. and Kasif, S. (2003). Extracting conserved gene expression motifs from gene expression data, *Proceedings of the Pacific Symposium on Biocomputing, Kauai, HI, USA*, pp. 77–88.

Orzechowski, P. and Boryczko, K. (2016). Text mining with hybrid biclustering algorithms, *Proceedings of the 15th International Conference on Artificial Intelligence and Soft Computing, Zakopane, Poland*, pp. 102–113.

Prelić, A., Bleuler, S., Zimmermann, P., Wille, A., Bühlmann, P., Gruissem, W., Hennig, L., Thiele, L. and Zitzler, E. (2006). A systematic comparison and evaluation of biclustering methods for gene expression data, *Bioinformatics* **22**(9): 1122–1129.

Rodriguez-Baena, D.S., Perez-Pulido, A.J. and Aguilar-Ruiz, J.S. (2011). A biclustering algorithm for extracting bit-patterns from binary datasets, *Bioinformatics* **27**(19): 2738–2745.

Serin, A. and Vingron, M. (2011). DeBi: Discovering differentially expressed biclusters using a frequent itemset approach, *Algorithms for Molecular Biology* **6**, Article no. 1.

Tanay, A., Sharan, R. and Shamir, R. (2005). *Handbook of Computational Molecular Biology*, Chapman & Hall/CRC Press, New York, NY.

Yang, J., Wang, H., Wang, W. and Yu, P. (2003). Enhanced biclustering on expression data, *Proceedings of the 3rd IEEE Symposium on Bioinformatics and Bioengineering, Bethesda, MD, USA*, pp. 321–327.

Marcin Michalak received a PhD in computer science in 2009 from the Institute of Informatics at the Silesian University of Technology. Since then he has been an active scientist in the institute, now as an assistant professor. His scientific areas of interest include rough sets, data mining, machine learning, image processing and, recently, biclustering as the area of Boolean reasoning application.

Roman Jaksik received a PhD in biocybernetics and biomedical engineering from the Silesian University of Technology in 2013. He is currently an assistant professor at the Department of Automatic Control there. He is an author or a co-author of more than 50 journal articles, book chapters and conference papers. His research interests concern the development of novel bioinformatic methods for the study of data obtained using high throughput measurement methods as well as the study of intracellular processes related to the development and progression of cancer.

Dominik Ślęzak received his MSc degree in mathematics (1996) and his PhD degree in computer science (2002) from the University of Warsaw, and his DSc degree from the Institute of Computer Science of the Polish Academy of Sciences (2011). He has worked at the Polish-Japanese Institute of Information Technology, the University of Regina and York University. Currently he works at the Institute of Informatics of the University of Warsaw. He is continually engaged in both academic and commercial R&D initiatives in the fields of artificial intelligence, machine learning and big data.

Received: 11 December 2018
Revised: 30 April 2019
Re-revised: 10 June 2019
Accepted: 2 September 2019