Acta Informatica (2022) 59:673–685
https://doi.org/10.1007/s00236-021-00415-9

ORIGINAL ARTICLE

Hierarchical heuristics for Boolean-reasoning-based binary bicluster induction

Marcin Michalak[1]

Received: 5 October 2021 / Accepted: 30 December 2021 / Published online: 21 January 2022
© The Author(s) 2022

Abstract
Biclustering is a two-dimensional data analysis technique that, applied to a matrix, searches for a subset of rows and columns that intersect to produce a submatrix with given, expected features. Such an approach requires different methods to those of typical classification or regression tasks. In recent years it has become possible to express biclustering goals in the form of Boolean reasoning. This paper presents a new, heuristic approach to bicluster induction in binary data.

1 Introduction

Biclustering is a data analysis technique that searches for interesting submatrices of a given matrix. The resultant submatrix, referred to as a bicluster, can be defined as an ordered pair, consisting of a subset of rows and a subset of columns of the given matrix. This approach was first used in the 1970s by Hartigan [1].

Boolean reasoning [2] is a paradigm computational task solving. Typically, the original issue becomes encoded into a Boolean formula and results of its transformation may be decoded into solutions of the original problem. Such an approach is widely applied in Rough Set Theory [2,3]; however, it is also used for decision tree induction [4].

Among many others [2,5–7], a new approach to biclustering based on Boolean reasoning was presented [8] in 2018. Promising results with discrete and binary data took effect also with continuous data biclustering methods development [9].

The primary disadvantage of methods based on Boolean reasoning is high computational complexity, due to Boolean function satisfiability checking. This problem has given rise to the use of heuristics to accelerate the computations. In the paper [10] the proposition of a simple, sequential covering strategy is presented. The approach searches for the set of biclusters that together contain all ones in the binary data and requires the modified Johnson's strategy of prime implicant approximation [11]. Some scenarios, however, may require wider biclusters, potentially including those which overlap one another, that are more general and do not necessarily contain all ones in the binary data.

✉ Marcin Michalak
Marcin.Michalak@polsl.pl

1 Department of Computer Networks and Systems, Silesian University of Technology, ul. Akademicka 16, 44-100 Gliwice, Poland

The need for wide bicluster searches has engendered new development of bicluster induction heuristics. The above-mentioned modified version of Johnson's strategy uses single prime implicant approximation in order to produce an iterative, sequential coverage of ones in a binary matrix. In general, searching only among the uncovered ones of a generated bicluster provides smaller biclusters (i.e., biclusters with fewer rows or columns) as the number of iterations increases.

This paper presents a new hierarchical, heuristic strategy for binary data biclustering. The heuristics used is that of the modified version of Johnson's strategy of prime implicant approximation [10]. The use of the term hierarchical refers to the "tabu search" [12] paradigm: following the discovery of a solution (node), solutions that are similar but not equivalent are found (subnodes). This process then iterates. The similarity condition is satisfied by random data modification: several subnodes are invoked, and for each of them a different element of the input data is altered from one to zero (only ones in a bicluster discovered by the given node are altered in this manner).

This paper is organized as follows: it begins with a brief review of existing approaches to biclustering; following this, the essential notions of Boolean-reasoning-based biclustering are defined and presented; the subsequent section develops the central concept of the hierarchical, heuristic strategy for binary biclustering induction, and provides abstract examples and pseudo-code; the penultimate section reports the results of experimental tasks using artificial data; and the final section offers conclusions and a perspective on possible further work in this area.

2 Related works

Biclustering was first used in the 1970's [1]. Following this, the technique has been applied to many disciplines, including biomedical data analysis [13] and text mining [14], leading to the development of multiple approaches to biclustering.

Tanay et al. [5] describe several biclustering methods including Cheng and Churs's algorithm [15], coupled two-way clustering [16], iterative approaches [17,18], SAMBA [19], spectral approaches [20], and plaid models [21]. Pontes et al. [22] provide a complex classification of biclustering methods, divided into:

- greedy strategies [1,15,23],
- stochastic approaches [24,25],
- meta-heuristics [26,27],
- clustering-based approaches [28,29],
- graph-based approaches [19,30],
- one-way clustering-based approaches [16,31],
- probabilistic models [21,32],
- linear-algebra-based models [17,20], and
- row and column reordering approaches [33].

Beyond methods dedicated strictly to biclustering, other data analysis paradigms share similar characteristics. For example, the search for an inclusion-maximal bicluster of ones in a binary matrix is comparable to the extraction of the concept lattice for a given context [34]. In the domain of basket analysis, the generation of a frequent itemset corresponds to the generation of an exact bicluster [35].

3 Boolean-reasoning-based biclustering

This section defines objects and concepts that will be used throughout the paper as well as provides backgrounds of Boolean-Reasoning-Based biclustering.

3.1 Definitions

Definition 1 (Bicluster) Let M be a matrix with rows R and columns C. The bicluster $\mathcal{RC} \equiv (\mathcal{R}, \mathcal{C})$ is an ordered pair of a susbset of rows $\mathcal{R} \subseteq R$ and a subset of columns $\mathcal{C} \subseteq C$.

Definition 2 (Exact bicluster) Let $\mathcal{RC}$ be a bicluster. $\mathcal{RC}$ is exact iff

$$\forall_{r_i, r_j \in \mathcal{R}} \forall_{c_u, c_v \in \mathcal{C}} \; M(r_i, c_u) = M(r_j, c_v),$$

where $M(r, c)$ is the element (cell) of matrix M in row r and column c.

Definition 3 (Inclusion–maximality of exact bicluster) Let M be a binary matrix and let $\mathcal{RC}$ be an exact bicluster derived from M. The bicluster is inclusion-maximal if and only if there exists no row $r \in R \setminus \mathcal{R}$ or column $c \in C \setminus \mathcal{C}$ such that any of the extended biclusters

$$(\mathcal{R} \cup r)\mathcal{C} \text{ or } \mathcal{R}(\mathcal{C} \cup c) \text{ or } (\mathcal{R} \cup r)(\mathcal{C} \cup c)$$

are also exact biclusters.

Definition 4 (Implicant) Let $f(a_1, \ldots, a_n)$ be a Boolean function of n Boolean variables. The expression

$$P_f(A = \{a_m, \ldots, a_p\}) = a_m \wedge \ldots \wedge a_p, \quad A \subseteq \{a_1, \ldots, a_n\},$$

such that

$$P_f(A) = 1 \Rightarrow f(a_1, \ldots, a_n) = 1,$$

is an implicant of the function f.

Definition 5 (Prime implicant) Let $f(a_1, \ldots, a_n)$ be a Boolean function of n Boolean variables and let $P_f(A)$ be an implicant of f. $P_f(A)$ is a prime implicant if and only if for all $A' \subset A$ $P_f(A')$ is not an implicant.

Definition 6 (Row/column corresponding variable) Let M be a matrix with rows R and columns C. Each row r (column c) has a corresponding Boolean variable r' (c').

Definition 7 (Implicant and bicluster correspondence) Let M be a matrix with rows R and columns C. Bicluster $\mathcal{RC}$ and implicant $P_f(A)$ correspond to one another if and only if

$$A = \{a' : a \in (R \cup C) \setminus (\mathcal{R} \cup \mathcal{C})\}.$$

That is to say, the implicant and bicluster correspond if and only if the implicant contains Boolean variables that correspond to rows and columns that are not elements of the bicluster. Such a corresponding implicant is denoted as

$$P_f(A) = \mathcal{R}'\mathcal{C}'.$$

3.2 Boolean reasoning in binary biclustering

Michalak and Ślęzak [8] provide the mathematical background for bicluster induction with discrete and binary data in the context of Boolean reasoning. There exist two theorems that bind biclusters of binary matrices and implicants of precisely defined (and data dependent) Boolean formulas. As written here the definition and theorems are used to find exact biclusters of ones among a background of zeros in a matrix. Replacing zero with one in the text, and vice versa, generates a definition and theorems for finding exact biclusters of zeros in binary data.

Definition 8 (Zero-encoding Boolean function) Let M be a binary matrix with rows R and columns C. The zero-encoding Boolean function is the conjunction of disjunctions of the corresponding variables of row $r \in R$ and column $c \in C$, such that $M(r, c) = 0$:

$$f(M) = \bigwedge \left(r' \vee c'\right), \quad M(r, c) = 0.$$

Following the above function definition Michalak and Ślęzak [8] prove two theorems. Here, the theorems are stated. The first theorem details the correspondence between implicants of $f(M)$ and exact biclusters of M.

Theorem 1 (Exact bicluster and implicant correspondence theorem) *Let M be a binary matrix with rows R and columns C. Bicluster $\mathcal{RC}$ is an exact bicluster of ones in M if and only if $\mathcal{R}'\mathcal{C}'$ is an implicant of $f(M)$.*

The second theorem demonstrates the correspondence between exact, inclusion-maximal biclusters of ones in M and prime implicants of $f(M)$.

Theorem 2 (Exact, inclusion-maximal bicluster and prime implicant correspondence theorem) *Let M be a binary matrix with rows R and columns C. Bicluster $\mathcal{RC}$ is an exact, inclusion-maximal bicluster of ones in M if and only if $\mathcal{R}'\mathcal{C}'$ is a prime implicant of $f(M)$.*

Consider the binary matrix M presented in Table 1. The goal is to find all exact, inclusion-maximal biclusters of ones. The formula $f(M)$ can be expressed at the logical multiplication of two-literal clauses. A given two-literal clause consists of the Boolean variables that correspond to the row and column indices of a zero value element of M (Table 1). Consider the matrix element $M(1, b) = 0$. The corresponding two-literal clause has the form:

$$(1 \vee b).$$

Note that the same notation is used for both rows and columns and the Boolean variables that correspond to those rows and columns. However, the meaning is context-dependent. For example, b can represent either an index, as in $M(1, b)$, or a Boolean variable, as in $1 \vee b$.

The formula that encodes all zeros in the matrix M has the following form:

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

Transforming this into a function that consists only of prime implicants gives:

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

The result is a function of three prime implicants, each of which corresponds (via Theorem 2) to an exact, inclusion-maximal bicluster of ones. A visualization of the biclusters corresponding to the $f(M)$ prime implicants is presented in Table 2. Note that none of the biclusters contain a zero, and neither may they be extended by row or column without subsequently containing a zero.

Table 1 An example binary matrix M

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

Table 2 Prime implicants of the $f(M)$ function and their corresponding biclusters

$1 \wedge 2 : (\{3\}, \{a, b, c\})$				$2 \wedge b : (\{1, 3\}, \{a, c\})$				$b \wedge c : (\{a\}, \{1, 2, 3\})$			
	a	b	c		a	b	c		a	b	c
1	1	0	1	1	**1**	0	**1**	1	**1**	0	1
2	1	0	0	2	1	0	0	2	**1**	0	0
3	**1**	**1**	**1**	3	**1**	1	**1**	3	**1**	1	1

4 Heuristic and hierarchical search of wide biclusters in binary data

The above approach to binary data biclustering has a high degree of computational complexity due to the satisfiability problem of Boolean formulas. From Theorem 1, each implicant of a Boolean formula $f(M)$ encodes an exact bicluster of the matrix M. By exploiting this relationship heuristic strategies can be applied to find implicants of Boolean formulas.

A popular approach to prime implicant approximation searching, based on the frequency with which literals occur, is Johnson's strategy [11]. However, Michalak et al. [10] prove that this strategy may induct implicants for which the corresponding biclusters are empty (i.e., biclusters with rows but no columns, or vice versa). To avoid such situations, Michalak et al. [10] propose a new heuristic for implicant induction. In addition, they present an approach for sequential coverage of bicluster induction that covers all ones in the data.

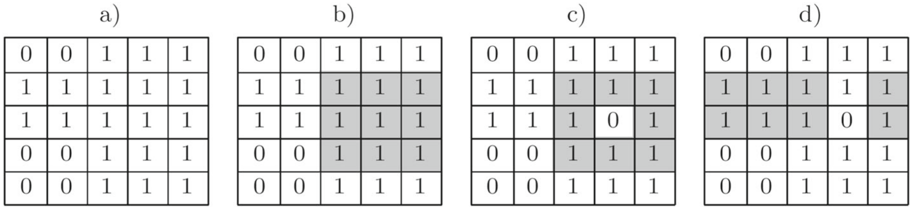
The sequential coverage strategy ensures that all ones in the data are eventually covered by a bicluster. However, as the process progresses, the size of newly found biclusters decreases. As a result, only biclusters found in the initial phase of the process may be generalizable. The new heuristic presented in this work adopts a different approach to finding biclusters.

Consider searching a binary matrix for biclusters of ones that are as wide as possible in both directions. The heuristics of Michalak et al. [10] provide a Boolean function implicant that encodes one exact bicluster of the data. This is the widest possible bicluster that the heuristics can find. Now consider the effect of replacing a one in the bicluster with a zero, and invoking the heuristics again. The result would not be the same bicluster as was originally found; the zero inside the original bicluster would violate the exactness condition. The visualization of two iterations of such an approach is presented in Table 3. It assumes a given heuristic to search for an exact bicluster of ones (not necessarily the heuristic above).

The example shown in Table 3 proceeds as follows. From the binary data (a) a bicluster is found using a given heuristic (b). An arbitrarily chosen one (third row, fourth column) is replaced by a zero (c). From the modified data the same heuristic is used to find another bicluster (d). The newly found bicluster can not be the same as was previously found.

For a bicluster consisting of r rows and c columns up to $r \cdot c$ different modifications to the original data can be made, and up to $r \cdot c$ new biclusters can be found heuristically. Moreover, each of the biclusters found in the modified data can be used as an input for further processing. This forms the general hierarchical strategy of bicluster induction.

Table 3 Modifying data to invoke another iteration of bicluster searching: **a** original binary data; **b** bicluster found by a given heuristic; **c** a single one in the bicluster is replaced by a zero; **d** from the altered data a new bicluster is found



a)

0	0	1	1	1
1	1	1	1	1
1	1	1	1	1
0	0	1	1	1
0	0	1	1	1

b)

0	0	1	1	1
1	1	1	1	1
1	1	1	1	1
0	0	1	1	1
0	0	1	1	1

c)

0	0	1	1	1
1	1	1	1	1
1	1	1	0	1
0	0	1	1	1
0	0	1	1	1

d)

0	0	1	1	1
1	1	1	1	1
1	1	1	0	1
0	0	1	1	1
0	0	1	1	1

Intuitively, such a recursive strategy can be executed until a given stop criterion is fulfilled. We can consider no fewer than four stop criteria:

- a maximum number of iterations (recursive invocations),
- a maximum number of found biclusters,
- a maximum total coverage of the data,
- a minimum assumed coverage of the data.

The pseudocode for this heuristic is presented in Algorithm 1.

Algorithm 1 Heuristics pseudocode.

```
1:  function biclusters := HeuristicSearch(data)
2:  biclusters := ∅                                    ▷ the set of generated biclusters (initially empty)
3:  run = true                                         ▷ set the loop condition
4:  queue = new Queue()                                ▷ create the empty queue of tasks
5:  queue.Add(data)                                    ▷ add data as the first task
6:  while run do
7:      whileData := queue.GetFirst()                  ▷ take first data from the queue
8:      queue.RemoveFirst()                            ▷ remove first task from the queue
9:      bicluster := FindSingleBicluster(whileData)    ▷ heuristic search of one bicluster in the data
10:     if not AlreadyCovered(biclusters, bicluster) then   ▷ checking whether bicluster is a subset of any
    previously found bicluster
11:         for all cell in bicluster do
12:             forallData = whileData.SetZero(element)    ▷ set the element value to zero
13:             queue.Add(forallData)                      ▷ add new data into the queue
14:         end for
15:         run := Update()                            ▷ update continue value due to stop conditions
16:     end if
17: end while
```

The *queue* (Algorithm 1, line 4) is used as part of the breadth–search strategy: each iteration of the *while* loop adds new data based on the found biclusters. The stop condition *continue* can take the form of one of the above criteria. The *FindSingleBicluster* method is an implementation of the heuristics from [10]. To avoid the need for postprocessing of the biclustering results (i.e., the removal of biclusters that are fully covered by others) the *AlreadyCovered* method (line 11) tests each newly found bicluster to determine if it is a subset (by rows and columns) of the union of a bicluster in the list of *biclusters*. If this is not the case, the newly found bicluster is appended to the list of *biclusters*, and it is inserted into the *queue* as new data.

In practice, in addition to stop conditions, limitations on tree generation are required. The first limitation deals with non-exhaustive subtree induction: this limits the percentage of bicluster ones that are replaced by zeros and processed further. The second limitation handles maximal tree depth. The pseudocode for this limited heuristics is presented in Algorithm 2.

Algorithm 2 Heuristics pseudocode with random queue generation and limited tree depth.

```
 1: function biclusters := HeuristicSearch(data, pct, maxdepth)
 2: biclusters := Ø                    ▷ the set of generated biclusters (initially empty)
 3: run = true                                               ▷ set the loop condition
 4: queue = new Queue()                                ▷ create the empty queue of tasks
 5: queue.Add(data, 0)                  ▷ add data as the first task with depth equal to zero
 6: while run do
 7:    whileData := queue.GetFirst()                        ▷ take first data from the queue
 8:    queue.RemoveFirst()                               ▷ remove first task from the queue
 9:    if whileData.depth > maxdepth then
10:       if queue.length > 0 then
11:          continue;                                          ▷ make another while loop run
12:       else
13:          queue.Add(uncoveredData, 0) ▷ prepare new task from uncovered data and put in into the queue
14:          continue;                                          ▷ make another while loop run
15:       end if
16:    end if
17:    bicluster := FindSingleBicluster(whileData)      ▷ heuristic search of one bicluster in the data
18:    if not AlreadyCovered(biclusters, bicluster) then    ▷ checking whether bicluster is a subset of any
   previously found bicluster
19:       subcells := ChooseCells(bicluster, pct)           ▷ select pct % of cells from the bicluster
20:       for all cell in subcells do
21:          forallData = whileData.SetZero(element)              ▷ set the element value to zero
22:          queue.Add(forallData, whileData.depth + 1)   ▷ add new data into the queue, incrementing the
   task depth value
23:       end for
24:       run := Update()                                  ▷ update value due to stop conditions
25:    end if
26: end while
```

Following the preparation of a new task in the main *while* loop, if the queue is not empty the depth of the task is checked. Tasks with a depth greater than a given threshold are omitted. (line 11). If the queue is empty a new root with zero depth is built from the remaining, uncovered original data (line 13) and appended to the queue.

5 Experiments

Experiments were performed on an example data set. The data set took the form of three binary matrices, presented in Fig. 1. The matrices are those used by Michalak and Ślęzak [8]. The three binary matrices were derived from a single data matrix (Fig. 1, left) which contained data of three discrete values. Each binary matrix was assigned one of the discrete values. If the value of a given element in the original data matrix was equal to one of the three discrete values, the corresponding element in the relevant binary matrix was set to one. All other elements of the binary matrices were set to zero.

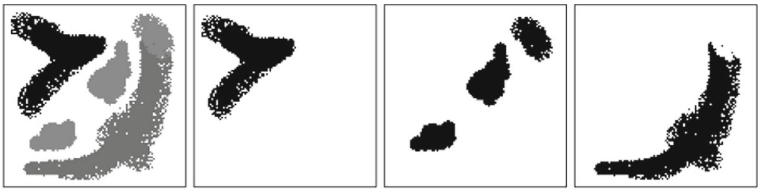


Fig. 1 The discrete data matrix (left) and the three binary matrices derived from it. The binary matrices were created using three discrete values: #0 (left center), #77 (right center) and #237 (right)

The set of all exact, inclusion-maximal biclusters in each of three binary matrices can be found by using the BiMax algorithm [2] or an exhaustive Boolean reasoning strategy [8]. The results of applying these methods are presented in Table 4.

The total coverage is the ratio of the number of ones that are covered by at least one bicluster and the total number of ones. This value must be equal to unity as both methods used are exhaustive: they find all inclusion-maximal biclusters, which necessitates that every one in the data is covered by at least one bicluster. The overlap is the ratio of the summed size of all biclusters (the number of rows multiplied by the number of columns) and the total number of ones in the data. It represents the average number of biclusters that cover a single one in the data.

To provide a comparison to the exhaustive search and hierarchical heuristics strategies, Table 5 presents the results of sequential covering using a modified version of Johnson's strategy.

The application of a hierarchical heuristics to the data was carried out with the following assumptions and parameter settings:

- the total coverage of ones in the data should be approximately 90%,
- the maximal depth of the search tree is set as three, to enforce search outside of the root bicluster, and
- up to 1% of matrix elements are selected from a newly found bicluster (Algorithm 2, line 20), but no fewer than three subiterations are invoked.

Originally, 10 experiments were performed on each binary matrix, with a generated bicluster value between 100 and 1000, in steps of 100. However, due to the experiments with 1000 biclusters providing unsatisfactory results, an additional three experiments per matrix were performed, with a generated bicluster value between 1,100 and 1300, in steps of 100.

Table 4 The results of applying an exhaustive strategy for binary bicluster induction to an example data set

Data	# Biclusters	Total coverage	Overlapping level
#0	5463	1.0000	453.81
#77	503	1.0000	34.80
#237	30,194	1.0000	3246.28

Table 5 The results of applying a modified version of Johnson's strategy for binary bicluster induction to an example data set

Data	# Biclusters	Total Coverage	Overlapping level
#0	224	1.0000	1.00
#77	129	1.0000	1.00
#237	201	1.0000	1.00

The results of using the exhaustive strategy (Table 4) provide a reference for subsequent results. The sequential coverage strategy (Table 5) generated a set of biclusters covering all ones; however, the size of the newly found biclusters decreased as the coverage increased. The results of using the hierarchical heuristics strategy, presented in Table 6, show a compromise between high generalization (biclusters that are wide in both directions) and computation period.

Figures 2, 3 and 4 present histograms of bicluster area for each of the three binary matrices, when using each of the three bicluster induction strategies. The histograms show the comprise between high generalization and computation period more clearly.

For each set of data the same observations can be made:

- the exhaustive strategy finds biclusters with a wider range of areas,
- the sequential coverage strategy finds biclusters with smaller areas,

Table 6 The results of applying a hierarchical heuristics for binary bicluster induction

Data	# Biclusters	Total coverage	Overlapping level
#0	1000	0.9456	19.55
#77	1000	0.9088	29.91
#237	1000	0.8240	16.00
#237	1300	0.9111	15.92

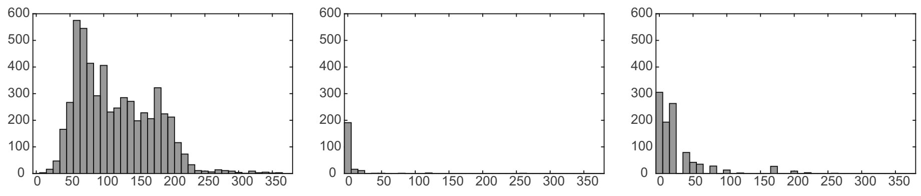


Fig. 2 Histograms of bicluster area for the #0 data: exhaustive strategy (left), modified version of Johnson's strategy (center) and hierarchical heuristics (right)

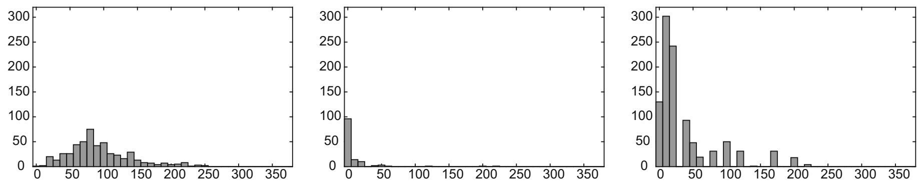


Fig. 3 Histograms of bicluster area for the #77 data: exhaustive strategy (left), modified version of Johnson's strategy (center) and hierarchical heuristics (right)

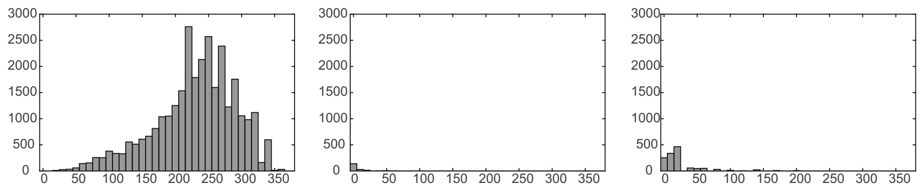


Fig. 4 Histograms of bicluster area for the #237 data: exhaustive strategy (left), modified version of Johnson's strategy (center) and hierarchical heuristics (right)

- the hierarchical heuristics strategy finds bicluster with a wider range of areas than those found by the sequential coverage strategy.

These observations demonstrate that the hierarchical heuristics strategy fulfills its expectations—the strategy can find more general biclusters in less time, compared to the modified version of Johnson's strategy.

Figure 5 presents the relationship between total coverage and number of biclusters generated, when using the hierarchical heuristics strategy.

As the hierarchical heuristics strategy implements sequential coverage (only newly found biclusters that cover previously uncovered data are added to the final set), coverage increases as the number of generated biclusters increases.

Figure 6 presents the relationship between bicluster area and iteration number, when using the hierarchical heuristics strategy. The observed relationship validates the central concept of the strategy. When using the modified version of Johnson's strategy, all covered ones are replaced with zeros, and the updated data is used as an input for further analysis. When using the hierarchical heuristics strategy, only a small number of ones are replaced with zeros. Invoking the process recurrently allows for the induction of biclusters that can cover both previously covered and previously uncovered ones, increasing bicluster generalization.

The results of applying the hierarchical heuristics strategy to the #237 data (Fig. 6, bottom) provides further insights. The strategy generates biclusters with small areas between iterations 200 and approximately 400. At approximately iteration 400 a steep increase in the area of the generated biclusters can be observed. A similar situation occurs at approximately iteration

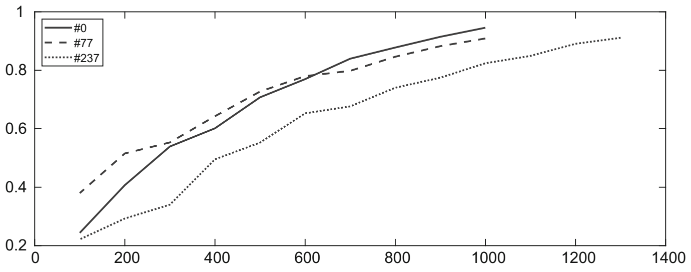


Fig. 5 Total coverage as a function of the number of biclusters generated by the hierarchical heuristics strategy

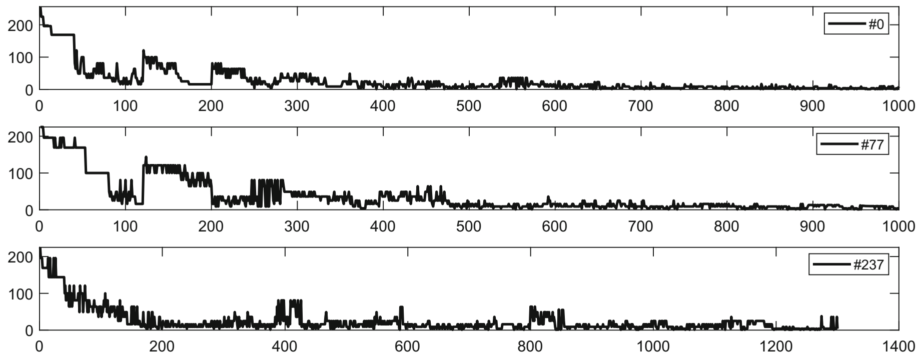


Fig. 6 The area of the bicluster as the function of the iteration number

Table 7 The results of hierarchical heuristics strategy for binary bicluster induction with the complete coverage of ones

Data	# Biclusters	Total coverage	Overlapping level
#0	1248	1.000	21.54
#77	1434	1.000	28.46
#237	2130	1.000	17.55

800. This is caused by the strategy "jumping" to unexploited regions of the matrix (as a new root bicluster is generated) and inducting matrices from those uncovered areas.

The hierarchical strategy is capable of a total coverage value of unity. Table 7 presents the number of iterations required for this, in addition to the overlap.

6 Conclusions and further works

Exhaustive search generates the widest possible exact biclusters; however, such an approach has high computational complexity (with regards to both processing time and memory). In the paper [10] it was attempted to decrease the computation time while retaining the theoretical background of the approach by modifying Johnson's strategy of prime implicants approximation. Although this was successful, the sequential coverage approach limited the area of the generated biclusters. Based on the modified version of Johnson's strategy, the hierarchical heuristics strategy, introduced in this work, is capable of more general bicluster induction. Experimental results when using the hierarchical heuristics strategy are promising, demonstrating an ability to find widespread biclusters covering a substantial section of the binary data.

Further modifications to the strategy could be considered. The use of a task queue provides a straightforward method to further decrease computation time. As all tasks are independent (with regards to single task analysis), they can be executed concurrently on different processor cores or computing nodes. Moreover, the initial results when using the hierarchical heuristics strategy could be postprocessed in order to remove small and insignificant biclusters. This confirms that the application of the Boolean reasoning paradigm to binary data biclustering continues to provide challenges to be solved.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Hartigan, J.A.: Direct clustering of a data matrix. J. Am. Stat. Assoc. **67**(337), 123–129 (1972). https://doi.org/10.1080/01621459.1972.10481214
2. Prelić, A., Bleuler, S., Zimmermann, P., Wille, A., Bühlmann, P., Gruissem, W., Hennig, L., Thiele, L., Zitzler, E.: A systematic comparison and evaluation of biclustering methods for gene expression data. Bioinformatics **22**(9), 1122–1129 (2006)

3. Pawlak, Z., Skowron, A.: Rough sets and Boolean reasoning. Inf. Sci. **177**(1), 41–73 (2007)
4. Nguyen, H.S., Nguyen, S.H.: From optimal hyperplanes to optimal decision trees. In: Tsumoto S., Kobayashi S., Yokomori T., Tanaka H. , Nakamura A. (ed.) Proceedings of the Fourth International Workshop on Rough Sets, Fuzzy Sets, and Machine Discovery (RSFD '96), pp. 82–88 (1996)
5. Tanay, A., Sharan, R., Shamir, R.: Discovering statistically significant biclusters in gene expression data. Bioinformatics **18**(suppl 1), 136–144 (2002)
6. Kluger, Y., Basri, R., Chang, J.T., Gerstein, M.: Spectral biclustering of microarray data: coclustering genes and conditions. Genome Res. **13**(4), 703–716 (2003)
7. Aguilar-Ruiz, J.S., Divina, F.: Evolutionary biclustering of microarray data. Lect. Notes Comput. Sci. **3449**, 1–10 (2005)
8. Michalak, M., Ślęzak, D.: Boolean representation for exact biclustering. Fund. Inform. **161**(3), 275–297 (2018). https://doi.org/10.3233/FI-2018-1703
9. Michalak, M., Ślęzak, D.: On Boolean representation of continuous data biclustering. Fund. Inform. **167**(3), 193–217 (2019). https://doi.org/10.3233/FI-2019-1814
10. Michalak, M., Jaksik, P., Ślęzak, D.: Heuristic search of exact biclusters in binary data. Int. J. Appl. Math. Comput. Sci. **30**(1), 161–171 (2020). https://doi.org/10.34768/amcs-2020-0013 https://doi.org/10.34768/amcs-2020-0013 https://doi.org/10.34768/amcs-2020-0013 https://doi.org/10.34768/amcs-2020-0013
11. Johnson, D.: Approximation algorithms for combinational problems. J. Comput. Syst. Sci. **9**, 256–278 (1974). https://doi.org/10.1016/S0022-0000(74)80044-9
12. Glover, F.: Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. **13**(5), 533–549 (1986). https://doi.org/10.1016/0305-0548(86)90048-1. Applications of Integer Programming
13. Henriques, R., Madeira, S.: Bicpam: pattern-based biclustering for biomedical data analysis. Algorithms Mol. Biol. **9**, 27 (2014). https://doi.org/10.1186/s13015-014-0027-z
14. de Castro, P.A.D., de França, F.O., Ferreira, H.M., Von Zuben, F.J.: Applying biclustering to text mining: an immune-inspired approach. In: de Castro, L.N., Von Zuben, F.J., Knidel, H. (eds.) Artificial Immune Systems, pp. 83–94. Springer, Heidelberg (2007)
15. Cheng, Y., Church, G.M.: Biclustering of Expression Data. In: Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology vol. 8, pp. 93–103 (2000)
16. Getz, G., Levine, E., Domany, E.: Coupled two-way clustering analysis of gene microarray data. Proc. Natl. Acad. Sci. **97**(22), 12079–12084 (2000) https://www.pnas.org/content/97/22/12079.full.pdf. https://doi.org/10.1073/pnas.210134797
17. Bergmann, S., Ihmels, J., Barkai, N.: Iterative signature algorithm for the analysis of large-scale gene expression data. Phys. Rev. E **67**, 031902 (2003). https://doi.org/10.1103/PhysRevE.67.031902
18. Ihmels, J., Friedlander, G., Bergmann, S., et al.: Biclustering of Expression Data. In: Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology vol. 8, pp. 93–103 (2000)
19. Tanay, A., Sharan, R., Kupiec, M., Shamir, R.: Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data. Proc. Natl. Acad. Sci. **101**(9), 2981–2986 (2004). https://doi.org/10.1073/pnas.0308661100
20. Kluger, Y., Ronen, B., Chang, J., Gerstein, M.: Spectral biclustering of microarray data: coclustering genes and conditions. Genome Res. **13**, 703–716 (2003)
21. Lazzeroni, L., Owen, A.: Plaid models for gene expression data. Stat. Sin. **12**(1), 61–86 (2002)
22. Pontes, B., Giráldez, R., Aguilar-Ruiz, J.S.: Biclustering on expression data: a review. J. Biomed. Inform. **57**, 163–180 (2015). https://doi.org/10.1016/j.jbi.2015.06.028
23. Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S.: A novel coherence measure for discovering scaling biclusters from gene expression data. J. Bioinform. Comput. Biol. **07**(05), 853–868 (2009). https://doi.org/10.1142/S0219720009004370
24. Yang, J., Wang, H., Wang, W., Yu, P.S.: An improved biclustering method for analyzing gene expression profiles. Int. J. Artif. Intell. Tools **14**(05), 771–789 (2005). https://doi.org/10.1142/S0218213005002387
25. Angiulli, F., Cesario, E., Pizzuti, C.: Random walk biclustering for microarray data. Inf. Sci. **178**(6), 1479–1497 (2008). https://doi.org/10.1016/j.ins.2007.11.007
26. Bryan, K., Cunningham, P., Bolshakova, N.: Application of simulated annealing to the biclustering of gene expression data. IEEE Trans. Inf Technol. Biomed. **10**(3), 519–525 (2006). https://doi.org/10.1109/TITB.2006.872073
27. Liu, J., Li, Z., Hu, X., et al.: Biclustering of microarray data with mospo based on crowding distance. BMC Bioinformatics **10**, 9 (2009). https://doi.org/10.1186/1471-2105-10-S4-S9
28. Cano, C., Adarve, L., Lopez, J., Blanco, A.: Possibilistic approach for biclustering microarray data. Comput. Biol. Med. **37**(10), 1426–1436 (2007). https://doi.org/10.1016/j.compbiomed.2007.01.005. QT Variability & Heart Rate Variability

29. Yan, D., Wang, J.: Biclustering of gene expression data based on related genes and conditions extraction. Pattern Recogn. **46**(4), 1170–1182 (2013). https://doi.org/10.1016/j.patcog.2012.09.028
30. Zhao, L., Zaki, M.J.: Microcluster: efficient deterministic Biclustering of microarray data. IEEE Intell. Syst. **20**(6), 40–49 (2005). https://doi.org/10.1109/MIS.2005.112
31. Tang, C., Zhang, A.: Interrelated two-way clustering and its application on gene expression data. Int. J. Artif. Intell. Tools **14**(04), 577–597 (2005). https://doi.org/10.1142/S0218213005002272
32. Segal, E., Taskar, B., Gasch, A., Friedman, N., Koller, D.: Rich probabilistic models for gene expression. Bioinformatics **17**(S1), 243 (2001). https://doi.org/10.1093/bioinformatics/17.suppl_1.s243
33. Ben-Dor, A., Chor, B., Karp, R., Yakhini, Z.: Discovering local structure in gene expression data: the order-preserving submatrix problem. J. Comput. Biol. **10**(3–4), 373–384 (2003). https://doi.org/10.1089/10665270360688075
34. Ignatov, D.I., Watson, B.W.: Towards a unified taxonomy of Biclustering methods. In: Russian and South African Workshop on Knowledge Discovery Techniques Based on Formal Concept Analysis, vol. 1522, pp. 23–39 (2016)
35. Serin, A., Vingron, M.: DeBi: discovering differentially expressed biclusters using a frequent itemset approach. Algorithms for Molecular Biology **6**(1), 1–12 (2011)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.