


DOI: <https://dx.doi.org/10.21123/bsj.2022.6373>

A Heuristic Approach to the Consecutive Ones Submatrix Problem

Rewayda Abo-Alsabeh^{1*} 

Hajem Ati Daham² 

Abdellah Salhi³ 

¹Department of Mathematical Sciences, Faculty of Computer Science and Mathematics, University of Kufa, Iraq

²Department of Mathematics, College of Education for Pure Science, Al Muthanna University, Iraq

³Department of Mathematical Sciences, Faculty of Science, University of Essex, UK

*Corresponding author: ruwaida.mohsin@uokufa.edu.iq

E-mail addresses: hajem.daham@mu.edu.iq, as@essex.ac.uk

Received 26/5/2021, Revised 2/4/2022, Accepted 3/4/2022, Published Online First 20/7/2022,
Published 1/2/2023



This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

Abstract:

Given a $(0, 1)$ -matrix, the Consecutive Ones Submatrix (C1S) problem which aims to find the permutation of columns that maximizes the number of columns having together only one block of consecutive ones in each row is considered here. A heuristic approach will be suggested to solve the problem. Also, the Consecutive Blocks Minimization (CBM) problem which is related to the consecutive ones submatrix will be considered. The new procedure is proposed to improve the column insertion approach. Then real world and random matrices from the set covering problem will be evaluated and computational results will be highlighted.

Keywords: Consecutive Block Minimization, Consecutive Ones Property, Consecutive Ones Submatrix, Column insertion, Heuristic Approach.

Introduction:

The C1S problem on a $(0, 1)$ -matrix is a generalization of the Consecutive Ones Property (C1P). The later has been proposed many decades ago. Fulkerson and Gross¹ suggested it as follows. Given an incidence matrix A , is it possible to rearrange the columns so that all the 1's in each row are together?

In combinatorial optimization, the C1P property is important since it indicates that the problem utilizes a matrix with this property simpler to solve than the original model. Indeed, such a matrix is totally unimodular. It appears in plenty of applications including computational biology, railway optimization, file organization, and scheduling. The C1P property is also used for ancestral genome reconstruction^{2,3}. In graph theory, it helps detecting interval and circle graphs^{4,5,6}.

C1P has been extensively investigated. Kendall^{7,8} indicated that the first study of the property was introduced by an archaeologist "Flinders Petrie" in the 19th century. Some heuristic approaches were established for this problem before the first polynomial complexity solution that was proposed by Fulkerson and Gross¹. Tucker⁹ showed a substructure characterization of the problem. He

used a graph theoretic method to characterize matrices by using forbidden consecutive ones submatrices. In 1976, Booth and Lueker¹⁰ provided a linear-time algorithm for it. They found a permutation that transforms a $(0, 1)$ -matrix into one with C1P. Their linear-time sequential algorithm is based on the PQ-tree data structure. Binary matrices can have the property if and only if their PQ-tree exists.

Let (α, β) -matrices be the $(0, 1)$ -matrices that are having at most α 1's and β 1's in per column and per row, respectively. For the problem with C1S, Hajiaghayi and Ganjali¹¹ solved it for the $(2, 2)$ -matrices in polynomial time and found that the problem for $(2, 4)$ -matrices is NP-Hard. These results post the issue of whether the C1S problem is NP-complete for the $(2, 3)$ and $(3, 2)$ matrices. Tan and Zhang¹² answered the question and showed that the two decision versions are NP-complete. They proved the problem is 0.8-approximable for $(2, 3)$ -matrices that have no two similar columns. In addition, they illustrated that it can be 0.5-approximable for $(3, 2)$ and $(2, \infty)$ matrices. Finally, they showed that the problem's

approximation for matrices of type $(\infty, 2)$ is NP-Hard within a factor of n^ϵ when $\epsilon > 0$.

Preliminaries

Definition 1. Given a $(0, 1)$ -matrix A , a set of consecutive 1 elements (0 elements) in a row of A is known as a block of ones (block of zeros) respectively.

Definition 2. A $(0, 1)$ -matrix A is said to have the consecutive ones property if the columns are permuted such that a consecutive block of 1's occurs per row¹³.

The C1P is satisfied for the columns by matrix transposition.

Definition 3. Let A be a $(0, 1)$ -matrix, finding largest set of columns in A that construct a submatrix has the C1P is called the C1S problem¹².

Definition 4. Given a $(0, 1)$ -matrix A , a columns' permutation of A which leaves the 1 entries consecutive in all the rows is called a valid permutation. If A is rearranged by this permutation then it contains the consecutive ones property.

The C1P is demanded as it often provides effective algorithms. Large attention for modifying and transforming a $(0, 1)$ -matrix into a matrix satisfying the C1P has been presented recently. These transformations can be delivered as the following problems¹⁴.

1. The problem of finding a maximal number of columns in a $(0, 1)$ -matrix A which induces a submatrix contains C1P is called the Max-C1S-C (Consecutive Ones Submatrix by Column).
2. The problem of finding a maximal number of rows in a $(1, 0)$ -matrix A which induces a submatrix contains C1P is called the Max-C1S-R (Consecutive Ones Submatrix by Row).
3. The problem of deleting the minimal number of columns which produces a matrix contains C1P is called the Min-C1S-C (Consecutive Ones Submatrix by Deleting Columns).
4. The problem of deleting the minimal number of rows that produces a matrix contains C1P is called the Min-C1S-R (Consecutive Ones Submatrix by Deleting Rows).
5. The problem of finding the minimal set of 1's in a matrix that can be changed into 0 results in a matrix with C1P is called the Min-C1-1E (Consecutive Ones by Flipping 1-Entries).

The third and fourth problems are equivalent to the first two. Generally, all the cases are also NP-hard for quite sparse matrices. Traditional methods rely on finding the Tucker forbidden submatrices^{9,14}. This paper introduces an evolutionary method to solve the C1S problem.

A related problem is the so-called Consecutive Block Minimization (CBM). The goal is to reduce

the blocks' number of ones per row by reordering the matrix's columns¹⁵. Haddadi et al.¹³, proposed a polynomial time heuristic to solving the problem. Leonardo CR and others¹⁶ proposed the most recent heuristic work on the problem of CBM. They introduced a heuristic relied on a traditional algorithm in graph theory. They designed a graphical representation to address the CBM problem and detect the reduction of the CBM in the traveling salesman problem. Abo Alsabeh¹⁷ suggested a metaheuristic approach for the C1S and CBM problems.

Another related problem is the Simultaneous Consecutive Ones Property (SC1P) where a $(0, 1)$ -matrix contains the C1P for rows and columns simultaneously. Subashini et al.^{18,19}, suggested the classical complexity and fixed parameter tractability of Simultaneous Consecutive Ones Submatrix (SC1S) and Simultaneous Consecutive Ones Editing (SC1E). They proved that the decision cases for the two problems are NP-complete.

Heuristic methods have been applied to enormous problems such as^{20,21}. The paper is arranged as follow. Studying some previous solution methods in Section 2. Illustrating the suggested column insertion algorithm to handle the problem in Section 3. Computational experience in Section 4. Conclusion in Section 5.

Previous Solution Procedures¹³

Polynomial Time Local Improvement Heuristic for CBM

The decision problem of the CBM is NP-complete when restricted to $(0, 1)$ -matrices including two ones per row. Nevertheless, Haddadi provided a polynomial time algorithm that finds a permutation where the number of consecutive blocks and the optimum do not vary further than 50%. Haddadi et al.¹³ provided a polynomial time local search algorithm for the problem. For a binary $m \times n$ -matrix A , they proposed two $O(n^2)$ -sized local neighbourhoods search such that the blocks' number of a neighbour is found in $O(m)$ time.

- *Improvement by two columns interchange:* Let A_ρ be a matrix associated with a permutation ρ in which the entire number of blocks is μ . Suggest the $O(n^2)$ -sized neighborhood $N(\rho)$ to be all the permutations that are produced from ρ by swapping two columns. Exploring the $N(\rho)$ for a permutation to give less blocks; if such permutation is not found, the algorithm stops. If an improvement δ is reached by interchanging two columns $\rho(i)$ and $\rho(j)$, $i \neq j$, (the new permutation is called ρ'), then update $\mu \leftarrow \mu - \delta$, $\rho'(i) \leftarrow \rho(j)$, $\rho'(j) \leftarrow \rho(i)$, and iterate the process with ρ' .

- *Improvement by column-shifting*: Let $N'(\rho)$ be the permutation set that results from ρ by inserting one column. In this method, the column is moved from its location and placed between two other columns. $N'(\rho)$ is searched for a permutation to produce fewer blocks and the search stops when no such permutation exists. Consider that an improvement δ is reached by inserting a column $\rho(i)$, and let the new permutation be $\rho'(i)$. If all the required updates are carried out, the algorithm is repeated for $\rho'(i)$. The two procedures are shown below in Algorithms 1 and 2.

Algorithm 1: Interchange procedure¹³

```

1  Input Positive integers  $m, n$ , a total number
    of blocks  $\mu$ , binary  $m \times n$  -matrix  $A$ ,
    permutation  $\rho$ ;
2  for  $i = 1, \dots, n - 2$ 
3  for  $j = i + 2, \dots, n$ 
4    Interchange two
5    non-adjacent columns  $\rho(i)$  and  $\rho(j)$ ;
6  Compute  $\delta$ ;
7  if  $\delta > 0$ 
8     $\mu \leftarrow \mu - \delta$ ;
9    Swap  $\rho(i)$  and  $\rho(j)$ ;
10 end if
11 end for
12 end for
13 Return Permuted matrix.

```

New Solution Approaches

An Alternative Column Insertion Procedure

A columns' permutation of a matrix that improves the size of the submatrix with C1S is wanted here. So a local improvement heuristic is suggested, which is polynomial in time. Suppose a $m \times n$ -matrix A with two submatrices, $m \times p$ -submatrix A_1 (it will be called the remainder submatrix) and $m \times (n - p)$ -submatrix A_2 which has the C1P. The columns of A_1 will be inserted one by one between the columns of A_2 . Consider investigating the matrix A_ρ associated with a permutation ρ where the entire number of C1S columns is β . Suppose the neighbourhood $N''(\rho)$ to be all the permutations that are produced from ρ by moving one column, where the column is moved from its location in A_1 and placed between two columns in A_2 . $N''(\rho)$ is explored seeking a permutation to increase the number of columns with C1S property. This procedure ends if there is no such permutation.

Algorithm 2: Column shifting procedure

```

1  Input Positive integers  $m, n$ , a total number
    of blocks  $\mu$ , binary  $m \times n$  -matrix  $A$ ,
    permutation  $\rho$ ;
2  for  $i = 1, \dots, n$ 
3  for  $j = 1, \dots, n$  ( $j \neq i - 1, j \neq i, j \neq i + 1$ )
4    Insert column  $\rho(i)$ 
5    between columns  $\rho(j - 1)$  and  $\rho(j)$ ;
6    Compute  $\delta$ ;
7    if  $\delta > 0$ 
8       $\mu \leftarrow \mu - \delta$ ;
9      if  $i > j$  move  $\rho(i)$  to position  $j$  in  $\rho$ 
10     else move  $\rho(i)$  to position  $j - 1$ ;
11    end if
12  end if
13 end for
14 end for
15 Return Permuted matrix.

```

The process starts by choosing the first left column from A_1 , say $\rho(j)$ for $1 \leq j \leq p$. This column is inserted between two adjacent columns of A_2 say $\rho(i)$, and $\rho(j + 1)$ for $i = 1, \dots, n - p$ and checked. If the column $\rho(j)$ improves the C1S columns in A_2 , update A_2 and choose the second column $\rho(j + 1)$ to insert in the new matrix. If $\rho(j)$ fails to provide an improvement then it is returned to the right-hand side of A_1 and $\rho(j + 1)$ is chosen for insertion.

Illustration:

Suppose A_2 is a submatrix with only two columns. The 0 and 1 in this submatrix have four arrangements (0 0, 0 1, 1 0, 1 1). Inserting column $\rho(j)$ in A_2 produces eight cases that are suggested by^{7,13}, (0 1 0, 0 0 0, 1 0 1, 1 1 1, 1 1 0, 0 1 1, 1 0 0, 0 0 1). A_2 is destroyed in one case, when the sequence of columns $\rho(i) \rho(j) \rho(i + 1)$ is 1 0 1, otherwise, the matrix still has the C1P.

If number of columns of A_2 is greater than 2, then another destructive case appears. That is when the sequence of columns $\rho(i) \rho(j) \rho(i + 1)$ is 0 1 0 and there is another block of ones in the same row (say 1 0 1 0), this row will lose the C1P property. Table 1 shows the checking of the different cases. The destructive cases are referred to as false and other cases that do not destroy A_2 as true. Column $\rho(j)$ fits between $\rho(i)$ and $\rho(i + 1)$ if the result of checking all the rows is true.

Table 1. Cases of checking a column inserted between two columns.

$\rho(i)$	$\rho(j)$	$\rho(i + 1)$		Result
1	0	1		false
0	1	0		false
0	0	0	and there is another	true
1	1	1	block of ones in this	true
1	1	0	row	true
0	1	1		true
1	0	0		true
0	0	1		true

If an improvement γ is achieved by inserting a column $\rho(j)$ in A_2 , the new permutation is called ρ . The matrix A_ρ will be updated to $A_{\rho'}$ and the process repeated with ρ' . The number of blocks which result from inserting an arbitrary column between two other columns can be evaluated in $O(m)$ operations, see¹³. The size of searching $N''(\rho)$ costs time $O(\sigma)$, where σ is clarified below. That is because there is one possibility to insert $\rho(j)$ between $\rho(i)$ and $\rho(i + 1)$. If it fits, then $\rho(j + 1)$ needs two possibilities, and so on. It can say that the size of possible columns for insertion is $\leq (n - 2)(n - 1)/2$.

Lemma 1. The complexity of the column insertion algorithm is $O(m\sigma p)$ where σ is the size of the neighborhood $N''(\rho)$ to explore. Proof. Searching the neighborhood $N''(\rho)$ costs $O(\sigma)$ operation where

$$\sigma = \sum_{N=n-p}^{n-1} N - 1, \quad 1$$

The integer number N starts with the $n - p$ columns of A_2 . For every neighbor, the calculations of checking the cases for all the rows cost $O(m)$. As long as the number of C1S columns β cannot be greater than n , and no less than $n - p$ columns, the finiteness of the inserting procedure is guaranteed and the number of improvements is at most $n - (n - p) = p$ in the worst case. ■

Algorithm 3: Column insertion procedure

```

1 Input Positive integers  $m, n$ , a total number of
   blocks  $\beta$ , binary  $m \times p$  -submatrix  $A_1$ ,
    $m \times (n - p)$  -submatrix  $A_2$ , with the C1P,
   and permutation  $\rho$ ;
2 for  $j = 1, \dots, p$ 
3 for  $i = 1, \dots, (n - p) - 1$ 
4 Insert column  $\rho(j)$  between  $\rho(i)$  and  $\rho(i + 1)$ ;
5 Check  $\rho(i) \rho(j) \rho(i + 1)$ ;
6 if the result of the check is true for all  $m$ 
7  $\beta \leftarrow \beta + \gamma$ ;
8 Move  $\rho(j)$  between  $\rho(i)$  and  $\rho(i + 1)$  and
  update  $A_2$ ;
9 end if
10 end for
11 end for
12 Return Permuted matrix.

```

Improving the number of blocks The procedure of column insertion that is used to search the neighbourhood $N''(\rho)$ for a permutation to maximize the C1S submatrix also minimize the number of blocks. Mentioning analogous arguments as in Lemma 1, the complexity of inserting the columns to find the minimum number of blocks is $O(m\sigma(g - m))$, where g is the number of 1's in A . Also, since the number of blocks μ cannot be greater than g , and no less than m , the finiteness of the insertion method is ensured and the number of improvements is at most $g - m$ in the worst case. This algorithm can be implemented directly on the matrix A by extracting the C1S matrix then fill the reminder columns.

Computational Experience

Implementing Column Insertion Algorithm

The matrix is separated into two submatrices, one having C1S (columns ≥ 2), then the column insertion algorithm is applied. From our experiments, it is found that the column insertion algorithm gives better results on larger C1S matrices. The results are promising: there are fewer blocks, and more columns having the C1P obtained in a reasonable time.

Overall, the heuristic is applied to many various matrices from real world data or generated by random. The generated square nonsymmetric matrices of the Set Covering Problem (SCP) are not tested for the C1P, then their optimums are not recognized. Therefore, the quality of the outcomes cannot be discussed. Randomly 10 matrices for different sizes are generated, the algorithm is performed then the average of the results of each size is taken. The remaining data, the real world

matrices, $(R_{1km} - R_{10km})$ which are shown in Table 2, come from the stop location problem, supplied by a German railway company²². This problem is written as SCP problem. The instances provide binary matrices which are supposed to contain almost C1P. The matrices B and C of small size that are generated by Ruf and Schöbel²², are sparse and almost contain the consecutive ones property with density 3% and 5%, respectively, the results are shown in Table 4. The heuristic is tested by the following ways:

Directly extract the C1S submatrix of a given matrix, and then apply Algorithm 3. Concerning the five real-world matrices, only two columns are separated as a C1S submatrix then the algorithm is applied. Results are in columns 5 to 8 of Table 3 and columns 4 to 7 of Tables 4 and 5.

Note that Tables 3, 4, and 5 show the average number of blocks and columns. The results of Table

3 show that Algorithm 3 gives a larger number of columns with C1P and fewer blocks than the original matrix. The results of Table 4 show that almost (30-50)% of the columns of the B and C matrices have C1P. Table 5 includes the outcomes on the real world matrices. Matrix R_{1km} has the C1P where final blocks' number is equal to m and the columns' number is equal to n , hence optimal. Concerning the remainder matrices of real world data, the last numbers of blocks and columns are close to the lower bound m and upper bound n respectively. One can say that the column insertion algorithm is fast with respect to computation time. In comparison with the results of the CBM problem from¹³, with respect to the number of blocks, it is found that the results do not differ a lot from theirs. The last column shows

Table 2. Test problem statistics.

Real-world matrices			Randomly generated instances					
Mat.	Dens. (%)	Size	Mat.	Dens. (%)	Size	Mat.	Dens. (%)	Size
R _{1km}	0.002	757 × 707	B1	0.032	100 × 96	C1	0.048	100 × 100
R _{2km}	0.014	1196 × 889	B2	0.036	100 × 95	C2	0.054	100 × 100
R _{3km}	0.022	1419 × 886	B3	0.034	100 × 92	C3	0.510	100 × 99
R _{5km}	0.043	1123 × 593	B4	0.031	100 × 92	C4	0.050	100 × 100
R _{10km}	0.203	275 × 165	B5	0.029	100 × 92	C5	0.051	100 × 100

Table 3. The column insertion performance for non-symmetric matrices.

First information				Column insertion			
Mat.	Dens.	Initial blocks	Initial(%) C1P	Final Blocks	Blocks improve.	Nbcols C1P	Time(s)
100	2	209	17	148	60	83	0.13
100	4	444	10	331	67	47	0.14
100	10	863	8	788	172	26	0.15
200	2	781	16	651	140	93	0.46
200	4	1379	8	1191	189	60	0.40
200	10	3248	8	896	351	32	0.45
500	2	5189	11	4538	651	118	5.19
500	4	9612	4	8856	766	67	4.82
500	10	22021	6	20826	893	34	5.96

Table 4. The column insertion performance for matrices having almost C1P.

First information			Column insertion			
Mat.	Initial blocks	Initial C1P	Final Blocks	Blocks improve.	Nbcols C1P	Time(s)
Matrices of sparsity 3%						
B1	296	12	267	29	47	0.15
B2	325	7	295	30	39	0.14
B3	304	10	279	25	38	0.13
B4	276	11	255	21	42	0.13
B5	270	13	242	28	48	0.13
Matrices of sparsity 5%						
C1	456	10	437	19	29	0.14
C2	516	7	496	20	22	0.15
C3	479	8	463	16	23	0.15
C4	482	7	468	24	23	0.13
C5	483	7	468	15	27	0.14

Their results. Also, the numbers of columns with C1P are not far from the columns' sizes of the matrices.

The three tables illustrate that:

1. The outcomes of Algorithm 3 do not entirely depend on the size of the C1S; they as well rely on the matrices' structures.

2. Computation time relies on the size of the matrix and the density. The algorithm performs well on matrices with more sparsity.
3. Reducing the blocks does not lead to producing a sizeable C1S submatrix. It is noticed that developing the C1S enhances the CBM. However, the reverse may not be correct since the submatrix size depends on the location of the destructive column.

Table 5. The column insertion performance for real-world instance matrices.

Mat.	First information		Column insertion			Time(s)	Algo. (1, 2) Final blocks
	Initial blocks	Initial C1P	Final blocks	Blocks improve.	Nbcols C1P		
Real-world matrices of sparsity (1 - 2)%							
R_{1km}	764	243	757	7	707	2.565	757
R_{2km}	1359	52	1210	157	882	6.940	1206
R_{3km}	1813	38	1487	326	859	8.566	1461
R_{5km}	1597	34	1185	412	568	3.152	1143
R_{10km}	389	32	281	108	159	0.081	280

Conclusion:

A heuristic method for solving the C1S problem is represented. A column insertion procedure was suggested to address the problem. The CBM problem is solved by Algorithm 3 and the maximum C1S submatrix is improved by using a polynomial time local algorithm with a complexity of $O(m\sigma p)$. The same algorithm is used for solving the minimum consecutive blocks problem in which the complexity for finding only the CBM is $O(m\sigma(g - m))$. The algorithm is applied to a set of real world matrices and randomly generated matrices from set covering. The outcomes present that large submatrices having the C1P can be detected. However, as the optimums are unknown for these matrices, it is impossible to say how far the solutions got by our procedure are actually resulted from them.

Acknowledgments:

The authors would like to thank Schöbel, one of the authors of that provided us with real-world data. Also, to Omar Kirikcki for supplying us with the random nonsymmetric matrices.

Authors' declaration:

- Conflicts of Interest: None.
- We hereby confirm that all the Figures and Tables in the manuscript are mine ours. Besides, the Figures and images, which are not mine ours, have been given the permission for re-publication attached with the manuscript.

- Ethical Clearance: The project was approved by the local ethical committee in University of Kufa.

Authors' contributions statement:

R. A. A. worked on the experiments and got the results above. A. S. helped with his experience to make the work straightforward and clear. He also revised the paper. H. A. D. searched for the literature on the problem and helped us write the paper.

References:

1. Fulkerson D, Gross O. Incidence matrices and interval graphs. *Pac J Math.* 1965; 15(3): 835-855.
2. Haddadi S, Chenche S, Cheraitia M, Guessoum F. Polynomial time local-improvement algorithm for consecutive block minimization. *Inf Process Lett.* 2015; 115(6): 612-617.
3. Tan J, Zhang L. The consecutive ones submatrix problem for sparse matrices. *Algorithmica.* 2007; 48(3): 287-299.
4. Dom M, Guo J, Niedermeier R. Approximation and fixed-parameter algorithms for consecutive ones submatrix problems. *J Comput Syst Sci.* 2010; 76(3): 204-221.
5. Tucker A. A structure theorem for the consecutive 1's property. *J Comb Theory Ser B.* 1972; 12(2): 153-162.
6. Safe MD. Circularly compatible ones, d-circularity, and proper circular-arc bigraphs. *arXiv preprint arXiv. 1906.00321.* 2019.
7. Pardal N. Structural characterization of some problems on circle and interval graphs. *arXiv preprint arXiv. 2006.00099,* 2020.
8. Safe MD. Characterization and linear-time detection of minimal obstructions to concave-round graphs and

- the circular-ones property. J Graph Theory. 2020; 93(2): 268-298.
9. Pardal N, Durán GA, Grippo LN, Safe MD, On nested and 2-nested graphs: two subclasses of graphs between threshold and split graphs. arXiv preprint arXiv. 1906.11970, 2019.
10. Cao Y, Grippo LN, Safe MD. Forbidden induced subgraphs of normal helly circular-arc graphs: Characterization and detection. Discret Appl Math. 2017;216:67-83.
11. De Luca F, Hossain MI, Kobourov S, Lubiw A, Mondal D. Recognition and drawing of stick graphs. Theor Comput Sci. 2019; 796:22-33.
12. Meidanis J, Porto O, Telles GP. On the consecutive ones property. Discrete Appl Math. 1998; 88(1): 325-354.
13. Booth KS, Lueker GS. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. J Comput Syst Sci. 1976; 13(3): 335-379.
14. Hajiaghayi MT, Ganjali Y. A note on the consecutive ones submatrix problem. Inf Process Lett. 2002; 83(3): 163-166.
15. Haddadi S. Benders decomposition for set covering problems. J Comb Optim. 2017; 33(1): 60-80.
16. Soares LC, Reinsma JA, Nascimento LH, Carvalho MA. Heuristic methods to consecutive block minimization. Comput Oper Res. 2020; 120: 104948.
17. Abo-Alsabeh R, Salhi A. A Metaheuristic approach to the C1S problem. Iraqi J Sci. 2021; 62 (1): 218-227.
18. Subashini R, Rani MR, Jagalmohan M. Simultaneous consecutive ones submatrix and editing problems: Classical complexity and fixed-parameter tractable results. Theor Comput Sci. 2020; 812: 13-38.
19. Kendall D. Incidence matrices, interval graphs and seriation in archaeology. Pac J Math. 1969; 28(3): 565-570.
20. Abduljabbar IA, Abdullah SM. An Evolutionary Algorithm for Solving Academic Courses Timetable Scheduling Problem. Baghdad Sci J. 2022; 19(2): 0399-0399.
21. Iqbal Z, Ilyas R, Chan HY, Ahmed N. Effective Solution of University Course Timetabling using Particle Swarm Optimizer based Hyper Heuristic approach. Baghdad Sci J. 2021; 18(4): 1465-1465.
22. Ruf N, Schöbel AA. Set covering with almost consecutive ones property. Discrete Optim. 2004; 1(2): 215-228.

نهج إرشادي لمشكلة C1S

عبد الله صالح³

حاجم حات دحام²

رويدة رزاق محسن¹

¹قسم الرياضيات, كلية علوم الحاسوب والرياضيات, جامعة الكوفة, العراق.
²قسم الرياضيات, كلية التربية للعلوم الصرفة, جامعة المثنى, العراق.
³قسم الرياضيات, كلية العلوم, جامعة اسكس, المملكة المتحدة.

الخلاصة:

أعطيت مصفوفة $(0,1)$ ، تم اقتراح مسألة المصفوفة الجزئية ذات الواحدات المتعاقبة والتي تهدف إلى إيجاد تبديل للأعمدة التي تزيد من عدد الأعمدة التي تحتوي معاً على قالب واحد فقط من الواحدات المتعاقبة في كل صف. سيتم اقتراح أسلوب الاستدلال لحل المسألة. كما سيتم دراسة مسألة تقليل القوالب المتتالية ذات الصلة بمسألة المصفوفة الجزئية ذات الواحدات المتعاقبة. تم اقتراح إجراء جديد لتحسين طريقة إدراج العمود. يتم بعد ذلك تقييم مصفوفات العالم الحقيقي ومصفوفات متولدة عشوائياً من مسألة غطاء المجموعة و تعرض النتائج الحسابية.

الكلمات المفتاحية: تصغير الكتل المتتالية, خاصية الواحدات المتتالية, مصفوفة الواحدات المتتالية الفرعية, إدراج العمود, استدلال.