UDC 004.021
DOI https://doi.org/10.32838/2663-5941/2020.1-1/21

Paulin O.M.
Odessa National Polytechnic University

Komleva N.O.
Odessa National Polytechnic University

Sinegub M.I.
Odessa National Polytechnic University

Sarafaniuk D.E.
Odessa National Polytechnic University

ABOUT MODIFICATION THE COVERAGE ALGORITHM USING THE "MINIMUM COLUMN – MAXIMUM ROW" METHOD

The paper considers the search algorithm for coverage by the method of "minimum column – maximum row". Unlike exact algorithms based on the methods of "full search", "boundary search", "using the properties of the coverage table", this algorithm is approximate – there is no guarantee of obtaining the shortest coverage. However, instead of some deterioration in the quality of the computational process (CP) of the search for coverage, its significant acceleration is achieved.

In order to further accelerate the search coverage CP, we propose to use an additional data structure, namely: an integer characteristic vector. The characteristic vector is intended for the initial preparation of data on the coverage table (CT) in the form of the sum of the individual elements of the columns/rows of the CT and the reflection of the current state of the CT when simulating its simplification (in reality, the CT does not change). This approach takes the solution of the coverage problem to a higher level of generalization.

For additional acceleration, a mask (binary vector) is introduced, obtained from the row vector by replacing the integer values of its elements with "1" values. Using the mask in the minimum column, those elements that have a value "0" in the converted row vector are zeroed, which reduces the number of rows to be searched when finding the maximum row.

From the CP of finding the coverage macrooperations that are additional to those obtained in previous works are distinguished. New macrooperations are associated with characteristic vector.

An experiment to determine the time for solving the problem for standard and modified algorithms for finding coverage is conducted. To clarify the factors due to which a decrease in the processing time of CT is achieved, the number of iterations and assignments in the computing process is calculated. A comparison of data on time and operations shows that the gain in time is determined by a decrease in the number of operations.

*Based on the results of the experiment, graphs are constructed from which the advantage of those proposed for solving the problem of finding the coverage of the entered data structures follows. The gain in time and complexity is greater, the larger the dimension l of the coverage table, determined by the product of the number of columns n by the number of rows m, l = m * n. From the same data, an estimate of the complexity of an algorithm of the form O (l) follows.*

Key words: *coverage table, algorithm, computational process, minimum column, maximum row, characteristic vector, binary vector, mask, macro operation, experiment, problem solving time, number of operations, algorithm complexity rating.*

Problem statement. The coverage problem is very common. For example, in the manufacture of a complex product, you may need some parts from various suppliers with different sets of parts and with different prices. Moreover, the manufacturer, solving the problem of coverage, can minimize either the total price of supplies (minimum coverage) or the number of suppliers (shortest coverage). In the general case, the coverage problem is considered as a variant of the search problem in some finite set of defined subsets with given properties.

There are [1] various methods and algorithms for solving the coverage problem, which are distinguished by the accuracy and complexity of the computing process. The following exact algorithms for finding coverages were considered in [2, p. 385, 3, p. 333]:

− method and algorithm for exhaustive search. Here, in a special way, enumeration of subsets of many rows of the coverage table is organized;

− concave set boundary-value algorithm. The algorithm is based on the method of generating subsets and their targeted selection;

− algorithm for processing a table of coverages based on its properties. The table of coverages may possess (not possess) the following properties: a row is nuclear or anti-nuclear, a column is absorbing, a row is absorbed. The method consists in sequentially considering the listed properties and, if available, in reducing the coverage table by deleting certain rows and/or columns.

Here, from the computational processes (CPs) under consideration, macrooperations were identified (MOs are functionally complete elementary CPs) and a consolidated list of MOs for the listed problems of finding coverage was compiled.

In [4, p. 260], the importance and relevance of constructing a high-quality CP, which is understood as the absence of various errors and proximity to the optimum, is noted. For this purpose, it was proposed to extract elementary processes from the CP and to put them in correspondence with the MOs, to compose their models in the form of *fragments* of the Petri net, and to conduct modeling of these fragments. If successful, MOs and their models are placed in the library. Then, from the fragments of the Petri net, a *complete* Petri net is compiled and its modeling is carried out; if successful, the CP and the corresponding Petri net are also placed in the library. In case of failure, the initial CP is corrected.

In this paper, we continue to consider the next method and algorithm for the allocation of MOs from the CP that implements this method. This time, an approximate method and an algorithm called "minimum column – maximum row" are considered. However, the simplest analysis showed that in this CP, although it is accelerated relative to other CPs of finding coverages by reducing the requirements for its optimality, not all reserves for acceleration used.

Latest research and publications analysis. The range of applications of coverage algorithms has been and remains quite wide. Consider some of these areas and show the features of applying coverage algorithms in them.

One area of application of coverage algorithms is software testing. As shown in [5, p. 27], functional testing provides for complete (most often the shortest) coverage with test cases of software functions. Testing by the "white box" method is carried out according to standard methods and consists in fully covering with tests all the structural elements of the program [6, p. 5]. Another current area of application for coverage algorithms is the use of heterogeneous communication technologies for smart cities and smaller infrastructures that allow them to communicate with each other through a network connection. At the same time, special techniques are being developed for data collection and providing an ubiquitous communication network, including, in addition to standard coverage algorithms, system scenarios for real-time operation [7, p. 785]. A number of works are devoted to the study of the efficiency and increase the speed of coverage algorithms in tasks requiring large computational resources. So, in [8, p. 166], an approach based on parallelizing the processing of initial data is presented. In this case, the global goal is divided into subgoals, which are achieved by different parallel processes, with the subsequent receipt of an integrated result. In [9, p. 236; 10, p. 4455], it was proposed to reduce the complexity of algorithmic problems, including coverage problems, by redistributing processing processes with the use of intelligent data processing tools.

As you can see, coverage algorithms are embedded as a component in a variety of data processing processes. However, their classic implementation has not been modified to optimize them either in speed or in complexity.

The **aim** of the work is to accelerate the coverage in the well-known method "minimum column – maximum row" by selecting suitable data structures.

To achieve the goal, the **task** of developing a method and algorithm for the accelerated coverage search is being solved. Along the way, the **task** of separating the MO from the computational process that implements the modified method "minimum column – maximum row" is being solved.

Basic material

Consider the basic definitions.

A *covering* is a subset A_i of a set A such that their union includes all elements b_j of the supporting set B, $b_j \in B$.

The optimal coverage is the shortest (the minimum number of subsets A_i) or the minimum coverage (the cost of coverage $C = \sum c_i$ is the minimum).

The *coverage table* (CT) is the matrix T of the membership relation of the subsets A_i to the supporting set B; rows are mapped to A_i elements, and columns are mapped to b_j elements.

In this paper, we consider an approximate algorithm for constructing a coverage close to the shortest one, based on the method of "minimum column – maximum row".

The method consists in repeating the following operations:

– search for minimum CT column;

– search for maximum CT row;

– simplification of CT by deleting the maximum row with its memorization, as well as columns covering it.

Operations are repeated until all columns are removed from CT.

The algorithm is based on this method, its idea is to reuse the procedure that implements the method.

The algorithm uses the following data structures.

The coverage table itself is a rather complex data structure, since it contains not just a two-dimensional array of independent data, but data *related* to the subsets of the basic set A belonging to the subsets of the *reference* set *B*.

To speed up the process of finding the coverage, we *introduce integer characteristic vectors* VCOL and VROW, containing initially the sum of digits "1" for all columns and rows, respectively, thereby *preparing* a description of the *initial state* of the CT. In the process of processing CT, these vectors reflect the *current state* of CT, despite the fact that CT itself *does not change.*

In fig. 1 shows a diagram of the interaction of the introduced vectors and CT. Here x = {0, 1} – are the values of the membership function of the elements of the string to the elements of the reference set, i.e. column names; y = ∑x – values of the sums of elements of columns, z = ∑x – values of the sums of elements of rows, y, z = {0, 1, 2, …}. Next to the vector image in Fig. 1 item numbers are affixed.

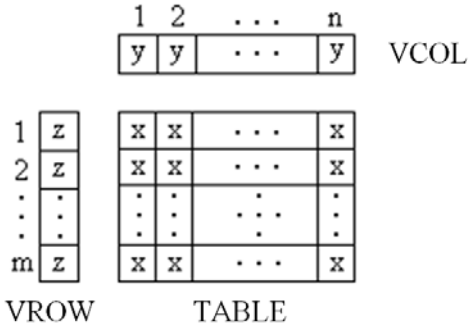

Fig. 1. Illustration of using of the vectors VCOL and VROW

To reduce the enumeration of the elements of the minimum CT column, we introduce a mask, for which we use a modified row vector VROW', MASKA[i]=VROW'[i]; the modification is that we replace the integer nonzero elements of the row vector VROW[i] with marks "1" (Fig. 2a).

Consider the example of the work of mask (Fig. 2б). The resulting vector is determined by the formula VREZ[i] = MASKA[i]⊗TABLE[i,k], where ⊗ – operation of elementwise multiplication of vectors, *k* – minimum column number.

Fig. 2. The example of mask operation on the k-th column of the CT

Data Structures

TABLE – coverage table;

VCOL – integer characteristic vector of columns whose *j*-th element is the sum of the marks "1" of the *j*-th column;

VROW – integer characteristic vector of rows whose *i*-th element is the sum of the marks "1" of the *i*-th row;

MASKA – binary vector representing vector VROW, MASKA initially consists of the marks "1";

VMIN – vector displaying the current minimum CT column;

VREZ – resulting vector obtained from VMIN in masking;

LNR – current row number list displaying VREZ;

LCOV – list of covering row numbers (coverage list);

counter – counter for counting the zero elements of a column vector;

k, l – variables indicating the minimum column number or the maximum row number respectively;

i, j – cycle parameters.

Verbal description of the algorithm

0. Entering the coverage table with dimension *mxn.*

1. Initialization. All vectors, lists, and the counter of zero elements are reset to zero; the initial mask are formed with filling it "1'.

2. The VCOL vector is filled with column information. To do this, the columns are iterated over and nonzero elements for the selected column are summarized. If in this case the sum is 0, then it formed the message "There is not coverage" and it required to go to step 14.

3. The VROW vector is filled with row information. To do this, the CT rows are iterated over and the nonzero elements in the selected row are summarized.

4. The main loop is organized with the postcondition for ending: "the VCOL vector is zero",

which imitates deleting all columns in the CT – see step 12.

5. The minimum element of the vector VCOL is determined with fixing its number *k*; in the case of several identical minimal elements the first one encountered mat be selected.

6. CT column *k* is Introduced as a vector: VMIN[i]:=TABLE[i,k], which is masked: VREZ[i]:=MASKA[i]⊗VMIN[i].

7. The current list of row numbers (LNR) is made to zero and it is formed a new list: those rows that contain "1" in VREZ[i] are written to it.

8. The maximum element in LNR is determined with fixing its number *l*; in the case of several identical maximum elements, the first one encountered is selected.

9. The number *l* to the current list of coverages LCOV is written.

10. The elements with number l in VROW and in MASKA are made to zero. This simulates the removal of the row number l from the CT.

11. The loop j defined by the condition TABLE [l, j] = 1 are gone through, and the *j*-th elements of the vector VCOL are zeroed out. This simulates the removal of covered columns from CT. For each j, it requires to do the following:

1) adjust VROW: subtract "1" from the element VROW [i] if TABLE [i, j] = 1;

2) if VROW [i] becomes equal to 0, then adjust the mask: MASKA [i]: = 0;

3) increase the counter value of the deleted columns by "1": counter: = counter + 1.

12. Checking whether the CT processing is ended by the condition "Counter< n?". If so, it is necessary to go to step 4.

13. Displaying a list of coverages.

14. The end.

According to the verbal description, the scheme of the modified algorithm is compiled (Fig. 3). Here are indicated: mincol – minimum column; maxrow – maximum row.

Extract of MOs. In [2, 3], as a result of considering the computational processes of exhaustive and boundary search and finding coverage by using the CT properties, the MO lists for each algorithm were compiled, as well as a composite MO list for all three algorithms.

Below is an additional list that takes into account the specifics of operations on vectors and their filling.

1. Entering data into a vector (summation of single elements of a column/row of a table and assignment).

2. Searching for minimum/maximum element among elements of a vector with traversal of the zero element.

3. Converting vector element format from integer form to binary form.

4. Masking a vector (element-wise product of a vector by a mask).

5. Zeroing a vector element according to some feature.

From these MOs, a higher-level hierarchy MO can be composed – MO "Pass through the Search Algorithm", the result of which is to add the next covering row to the coverage list and delete the covered columns, as well as the covering row itself.

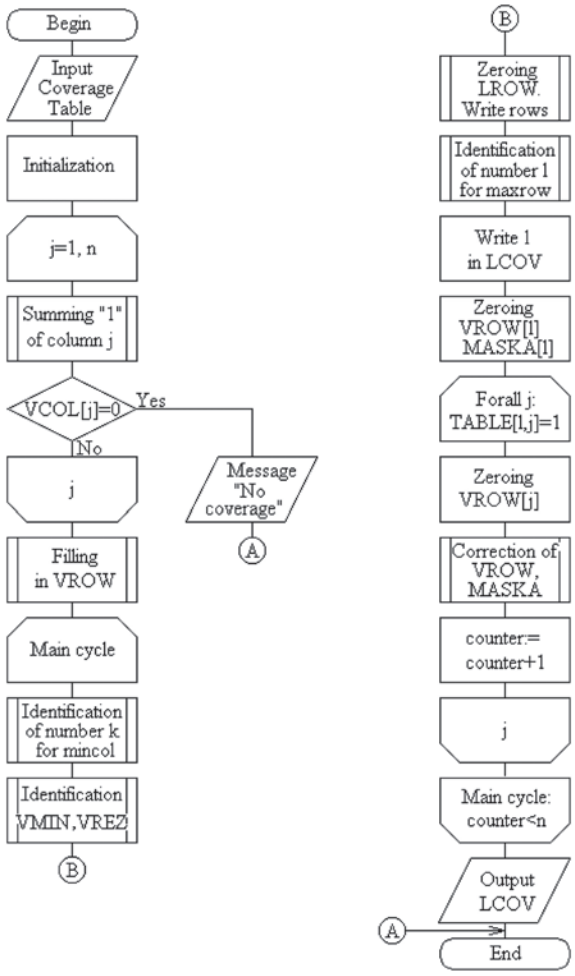

Fig. 3. Scheme of the modified algorithm "minimum column – maximum row"

Experimental part

The experiment is built to test the effectiveness of algorithms, standard and modified, on random matrices of various sizes: 8x8, 8x16, 16x16, 16x32, 16x64 (complexity scores are: 64, 128, 256, 512, 1024).

As a criterion of effectiveness, the *speed* and *number of operations* when solving the coverage problem are taken.

For each table of a certain dimension, 10 experiments are carried out.

The experiment is organized as follows.

1. CT of the selected dimension is filled with random binary values using the Random program, the output data stream of which is converted into a stream of two-digit decimal numbers using the operation mod100.

2. The resulting numbers for each digit are separately converted to binary decimal numbers encoded in 2421 (total – 1 byte, 2 tetrads).

3. All the rows of CT are looked through.

4. The received byte fits into the CT row from left to right, then the next byte is added to it, etc., until the row is exhausted.

5. If the constructed CT has a zero column, then the first 8 bits from the top are replaced with a new non-zero random byte.

At the same time, the time taken to enter the coverage table is excluded from the time of solving the problem for both algorithms.

In the experiment, a preliminary monitoring of the processing time of CT was carried out; average results on matrices of various sizes are summarized in Table. 1; according to the results of the experiment, graphs are built (Fig. 4). Note that the dimension *l* of the CT is determined by the product of the number of columns *n* by the number of rows *m, l = m*n.*

It is seen that the use of characteristic vectors accelerates the processing of CT.

For a more detailed analysis, counters of comparison and assignment operations were built into the compared algorithms. The experimental results for determining the number of operations are shown in the same table. 1; graphs are plotted on them (Fig. 5). It can be seen that there is a correspondence between the results in terms of speed and the number of operations.

Table 1

The results of the experiment

CT size	CT processing time, c		Number of operations	
	Method with vectors	Method without vectors	Method with vectors	Method without vectors
64	1066	1193	253	342
128	1656	1964	472	671
256	2706	3628	752	1239
512	4735	6948	1313	2358
1024	8464	13193	2313	4385

In conclusion, the analysis of the dependence of the complexity of the algorithm Q on the dimension *l* CT was made; an estimate of the complexity of the algorithm can be described by the expression T = O (*l*) for time complexity and K = O (*l*) for operational complexity.

Conclusions

1. In order to speed up the procedure for finding a coverage that is close to the shortest, characteristic vectors have been introduced for tracking the state of CT, which do not change the table during its processing. This brought the problem of finding coverage to a higher level of generalization.

2. The introduction of specific data structures (vectors) was followed by many auxiliary operations with respect to the standard solution, however this is

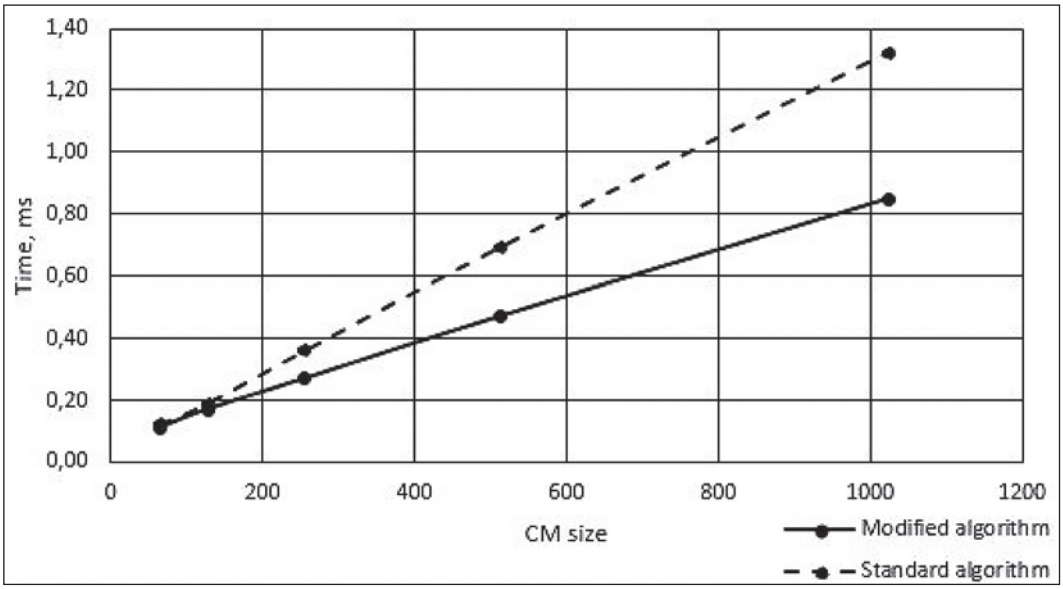

Fig. 4. Dependence of coverage matrix (CM) processing time on its size

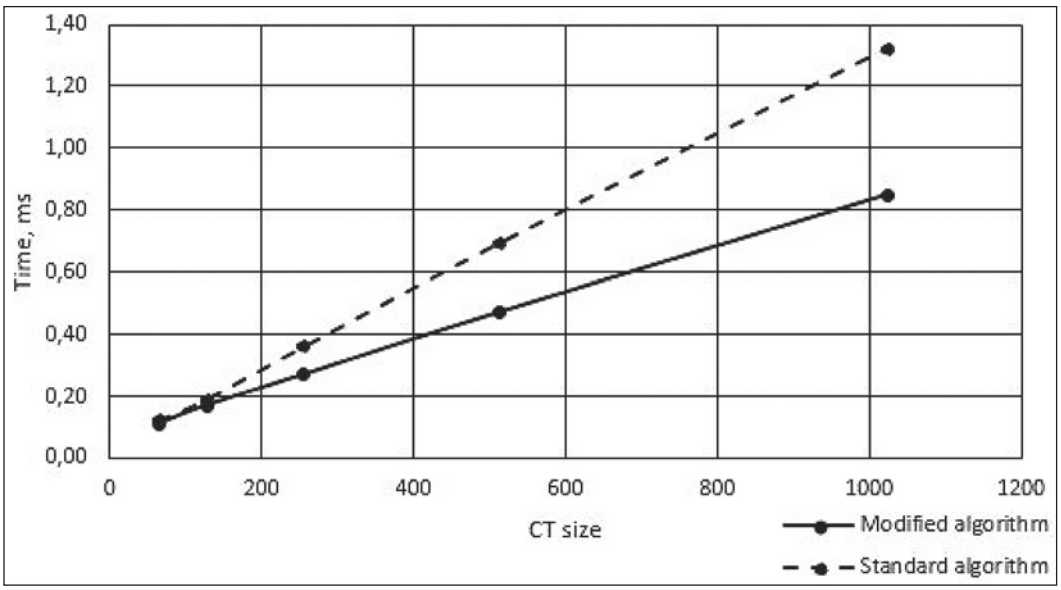

Fig. 5 Dependence of CT processing time on its size

fully compensated by the fact that the total number of comparison and assignment operations is significantly reduced.

3. The results of the experiments show a higher efficiency in terms of the time spent on the modified algorithm compared to the standard algorithm due to the introduction of more suitable data structures – characteristic vectors.

4. An increase in the efficiency of the algorithm is determined by a decrease in the number of comparison and assignment operations in the process of processing CT compared to the standard algorithm, and this decrease is greater, the larger the dimension of the matrices.

5. The complexity assessment of both the standard and the modified algorithm on the array of the studied CT is determined by a linear expression relative to the dimension of the CT: T= O(l) for time and K= O(l) – for operations. The difference between these estimates is manifested in the coefficients of l, which are not taken into account in such a representation.

References:

1. Новоселов В.Г., Скатков А.В. Прикладная математика для инженеров-системотехников. Дискретная математика в задачах и примерах : учеб. пособие. Киев : УМК ВО, 1992. 200 с.

2. Паулин О.Н. Вычислительные модели алгоритмов покрытия. *Информатика и математические методы в моделировании*. 2016. Т. 6, № 4. Одесса : ОНПУ. С. 385–396.

3. Паулин О.Н. Методы и алгоритмы покрытия (Часть 2). *Информатика и математические методы в моделировании*. 2017. Т. 7, № 4. С. 333–338.

4. Paulin O.N., Komleva N.O., Marulin S.U., Nikolenko A.O. Method for Constructing the Model of Computing Process Based on Petri Net. *Applied Aspects of Information Technology*. 2019. Vol. 2 No. 4. P. 260–270.

5. Mathur A.P. Foundations of software testing: fundamental algorithms and techniques. *New Delhi:Dorling Kindersley*. 2013. 324 p.

6. Mansoor A. Automated Software Test Data Optimization Using Artificial Intelligence. Int. J. Inf. Commun. Technol. Trends. 2013. V. 9. P. 5–19.

7. Ahuja K., Khosla A. A novel framework for data acquisition and ubiquitous communication provisioning in smart cities. *Future Generation Computer Systems – The International Journal Of Science*. 2019. V. 101. P. 785–803.

8. Sanchez-Gomez J. M., Vega-Rodríguez M. A., Pérez C. J. Parallelizing a multi-objective optimization approach for extractive multi-document text summarization. *Journal of Parallel and Distributed Computing*. 2019. V. 134. P. 166–179.

9. Zhanyang X., Xihua L., Gaoxing J., Bowei T. A time-efficient data offloading method with privacy preservation for intelligent sensors in edge computing. *Eurasip Journal On Wireless Communications And Networking*. 2019. V. 1. P. 236–246.

10. Hui L., Haining L., Shu Z., Zhaoman Z., Jiang C. Intelligent learning system based on personalized recommendation technology. *Neural Computing & Applications*. 2019. V. 31. Is. 9. P. 4455–4462.

Паулін О.М., Комлева Н.О., Синегуб М.І., Сарафанюк Д.Е.

ПРО МОДИФІКАЦІЮ АЛГОРИТМУ ПОКРИТТЯ ЗА МЕТОДОМ «МІНІМАЛЬНИЙ СТОВПЕЦЬ – МАКСИМАЛЬНИЙ РЯДОК»

У роботі розглядається алгоритм пошуку покриття за методом «мінімальний стовпець – максимальний рядок». На відміну від точних алгоритмів, заснованих на методах «повний перебір», «граничний перебір», «із використанням властивостей таблиці покриття», даний алгоритм є наближеним – тут відсутня гарантія отримання найкоротшого покриття. Однак замість деякого погіршення якості обчислювального процесу (ОП) пошуку покриття досягається його значне прискорення.

Із метою подальшого прискорення ОП пошуку покриття нами пропонується використовувати додаткову структуру даних, а саме цілочисельний характеристичний вектор. Характеристичний вектор призначається для початкової заготовки даних про таблиці покриття (ТП) у вигляді суми одиничних елементів стовпців/рядків ТП і відображення поточного стану ТП під час імітації її спрощення (насправді ТП не змінюється). Такий підхід виводить рішення задачі про покриття на більш високий рівень узагальнення.

Для додаткового прискорення вводиться маска (двійковий вектор), що отримується з вектора рядків заміною цілочисельних значень його елементів одиничними значеннями. За допомогою маски в мінімальному стовпці обнуляються ті елементи, які в перетвореному векторі рядків мають значення «0», що зменшує число рядків, які перебираються під час знаходження максимального рядку.

З ОП знаходження покриття виділяються макрооперації, додаткові до отриманих у попередніх роботах. Нові макрооперації пов'язані з характеристичним вектором.

У роботі проводиться експеримент із визначення часу рішення задачі для стандартного і модифікованого алгоритмів знаходження покриття. Для уточнення чинників, за рахунок яких досягнуто зниження часу оброблення ТП, підраховується кількість ітерацій і присвоювання в обчислювальному процесі. Зіставлення даних за часом і за операціями показує, що виграш за часом визначається зниженням числа операцій.

*За результатами експерименту побудовані графіки, з яких випливає перевага запропонованих для вирішення задачі знаходження покриття введених структур даних. Виграш за часом і складності тим більше, чим більше розмірність l таблиці покриття, що визначається добутком числа стовпців n на число рядків m, l = m * n. Із цих же даних слідує оцінка складності алгоритму виду O (l).*

Ключові слова: *таблиця покриття, алгоритм, обчислювальний процес, мінімальний стовпець, максимальний рядок, характеристичний вектор, двійковий вектор, маска, макрооперація, експеримент, час виконання завдання, кількість операцій, оцінка складності алгоритму.*