STUDIA INFORMATICA
Nr 1-2(14) Systems and information technology 2010

# Investigations of the efficiency of a parallel program for construction of the shortest covering of a Boolean matrix

**Adam Adamus, Sergey Novikov**
Institute of Computer Science
Siedlce University of Natural Sciences and Humanities
3 Maja Str. 54, 08-110 Siedlce, Poland

**Abstract**: The paper presents results of investigations of the efficiency of a parallel program for the solution of the NP-complete problem of constructing of the shortest covering of a Boolean matrix. It was explored dependencies of the time of constructing of the shortest covering from the number processing processors of a cluster and from the dimension of a Boolean matrix.

**Keywords:** Boolean matrix, shortest covering, parallel program, cluster.

## 1 Introduction

In modern Computer-aided design (CAD) systems are required to solve various tasks of combinatorial optimization with large dimension. Among them, NP-complete (difficult) problems of a synthesis, modeling and technical diagnostics of VLSI devices with thousands of components as element's modules of electronic circuits. Obviously, circuit's number of elements is an important parameter of the difficulty of solving of an optimization problem. Therefore, to solve a NP-complete (difficult) task with large dimension, approximate heuristic algorithms are used. They do not guarantee an optimality but let's get decisions sufficiently close to optimal in a reasonable time. However, even approximate algorithm requires too much computing for solving NP-complete large-scale problems. It is therefore of obvious interest the research opportunities an increase of the efficiency of solutions of our tasks with the help of modern multiprocessor computing systems (clusters).

There are several different forms of parallel computing: bit-level, instruction level, data, and task parallelism. However, we can't increase the efficiency of decisions NP-complete large-scale problems significantly without developing appropriate parallel algorithms.

## 2 Parallel algorithm for constructing of the shortest covering of a BM

This (now classic) NP-complete problem arises in the CAD of discrete devices for construction of minimal tests for technical diagnostics, at coding of states of a finite automaton and minimization of switching functions in the automata theory, in the graph theory and in designing of software systems. It is well known the sequential algorithm for the solution of the task of constructing of optimal coverings of a Boolean matrix proposed by A. Zakrevskij [1].

The parallel algorithm for solving the problem of constructing of the shortest coverings of a Boolean matrix (BM) ***M*** with ***n*** columns and ***q*** rows with the help of the ***(m +1)***-processor cluster was proposed by one of the authors [2]. When parallelizing computations with the help of our parallel algorithm to solve the following subproblems:

1) The partition of the original matrix ***M*** on ***m*** submatrices ***M1, M2 ,..., Mm***.

With the help of the program ***A1(M; M1, M2, ..., Mm)*** the control processor $p_0$ partitions the set of columns ***B = {b1, ..., bn}*** of the original matrix ***M*** on ***m*** subsets ***B1, ..., Bm***. Then for each of the subset ***Bi***, $i \in \{1,2,\ldots,m-1\}$, the program ***A1*** constructs the BM ***Mi*** with ***[n/m]*** columns and $q' \leq q$ rows, because consisting of all zeros strings are deleted. The exception is the MB ***Mm***, where there may be more columns. Elements of a row $a_j$ of original matrix ***M***, belonging to the columns of the set ***Bi***, form the corresponding row $a_{ji}$ of the matrix ***Mi***. A line, which consists only of zeros, is not included in the ***Mi***. Columns ***Mi*** are denoted the same symbols as corresponding columns in the matrix ***M***, even if they have a lower height.

Then the control processor $p_0$ sends submatrix ***M1, M2 ,..., Mm*** to processing processors $p_1,\ldots,p_m$ as input data.

2) Construction of the shortest coverings of Boolean matrices ***Mi*** **.**

Each processing processor constructs the shortest covering of a Boolean matrix ***Mi*** using the same program ***A2 (Mi; P(Mi)).***

Then a processing processor sends the solution ( the shortest covering P(Mi) ) on the control processor.

3) Construction of coverings of the original Boolean matrix ***M*** **.**

The control processor adds solutions (one covering for each matrix ***Mi***) and records the amount as a covering of the matrix ***M*** , i.e. $P'(M) = P(M1) \cup \ldots \cup P(Mm)$. We have $ai_1 \cup ai_2 \cup \ldots \cup ai_s = ai$ , where $1 \leq s \leq m$, in the sum of solutions with the help of the program ***A3({P(M1), ...,P(Mm)};P'(M)).***

Variants of the covering of the ***M*** form a set ***{P '(M)}.***

4) Preparation of the data to eliminate redundant elements in the ***P'(M).***

A covering P'(M) may contain redundant elements (rows of ***M***). Therefore, the control processor must prepare the source data for processing processors to eliminate redundant elements. The control processor using the program ***A4(M,P'(M);Ps(M),P'n(M),Ni)*** divides the elements of the covering ***P'(M)*** on important elements, which form a set ***Ps(M)*** , and non-essential elements ( ***P'(M) = Ps***

***(M)*** $\cup$ ***P'n (M)*** ), and builds for each element from ***{P '(M)}*** Boolean matrices ***Nj***. The control processor deletes from the matrix ***M*** columns, covered by elements from ***Ps(M)***. The remaining columns and rows from the set ***P'n(M)*** form the matrix ***Nj***.

5) Eliminating redundant elements from ***P'n (M).***

A processing processor must analyze the rows of the matrix ***Nj*** on a redundancy by using the same program ***A5(Nj;Pn(M))***. Any row $a_i$ of the matrix ***Nj*** is redundant if it can be covered by the disjunction of remaining rows of the matrix ***Nj.*** Thus a processing processor builds a set ***Pn(M)*** $\subseteq$ ***P'n(M)***.

Irreducible covering ***P(Nj) = Pn(M)*** a processing processor sends on the control processor.

6) We must carry out items 4 and 5, until with the help of the working processors all the options coverings from {P'(M)}will be analyzed. If the number of options equal to the magnitude | ***{P '(M)}*** | = ***K***, then the cycle is executed $]\,K/m\,[$ times.

7) Construction of the shortest coverings of original Boolean matrix ***M*.**

The control processor with the help of the program ***A6(Pn(M),Ps(M);P*(M))*** summarizes sets ***Pn(M)*** and ***Ps(M)*** and obtains a covering - set ***P(M) = Pn(M)*** $\cup$ ***Ps(M).*** There may be many options for coverage of ***M***. So the control processor must choose the set with a minimal number of elements. This set ***P*(M)*** is the shortest (optimal) covering of the original Boolean matrix ***M.***

## 3 Parallel program POKRMB

Program POKRMB, implementing a parallel algorithm, consists of the following six subprogrammes [3]:

- P1 for the preparation of data (from a file or the construction of BM by using the random number generator) with the help of the control processor $p_0$;
- P2 for the implementation (by the control processor $p_0$) the partition of the original matrix BM ***M*** on $w \le m$ submatrices ***M1, M2, ..., Mw***;
- P3 to build (by a processing processor $p_i$ ) a covering of BM ***Mi***;
- P4 for analysis (by a processing processor $p_i$) a covering of BM ***Mi*** on the redundancy;
- P5 to sum solutions (by the control processor $p_0$) to build a covering of BM ***M***;
- P6 for analysis (by the control processor $p_0$) a covering of BM ***M*** on the redundancy.

Program POKRMB can operate in two modes.

In the first mode (automatic generation of data) the user specifies as input three parameters: ***N***-number of columns of BM ($N \le 20000$); ***M*** - number of rows of BM ($M \le 17500$); ***p*** - the probability of occurrence of "unit" among the elements of the BM ($0 < p \le 0{,}5$).

The program P1 writes the required parameters, generates the BM with the required parameters and asks to indicate a number of submatrices on which this BM is required to divide. Number of submatrices may not exceed the number of processing processors in the cluster. Our cluster (Siedlce University of Natural Sciences and Humanities) allows the use of not more than 34 processing processors.

In the second mode, the user specifies the address of the file, containing the BM, and the number of submatrices ***w*** for the parallel construction of the shortest covering columns of BM.

The program P1 checks the correct preparation of input data. Each column of the original BM must contain at least one “unit”. Our problem has no solution if this condition is violated.

Then the control processor with the help of the program P2 divides the original matrix on $\boldsymbol{w \leq m}$ submatrices. Then the control processor sends submatrices to processing processors.

Then, in parallel each employed processing processor uses P3 to construct the shortest covering of the corresponding submatrix. The obtained covering may be reduced with the help of the program P4 to find the shortest covering of the corresponding submatrix.

The control processor using the P5 unites the shortest coverings of submatrices ***M1, M2, ..., Mw*** and builds a covering for the original BM ***M***. The obtained covering may be reduced with the help of the program P6 to finding the shortest covering of the original BM ***M***.

Our program POKRMB is written in C++ using a free IDE Dev-C++ for C/C++ (GPL license for Windows and Linux). The program takes into account all the requirements of windows-port MinGW, which is a prerequisite for using a core compiler GCC.

The user that runs on his PC with Windows, has an opportunity to run their programs on the cluster with Unix (with the help of the channel of distribution Debian). Program modules, which run simultaneously on multiple processors can exchange messages through the use of library MPI (Message Passing Interface).

We prepared the sequential program-model POKRMB' to test our parallel program POKRMB. After debugging of the model POKRMB' on our PC and making the appropriate upgrades we have completed the debugging of our parallel program POKRMB in cluster of Siedlce University of Natural Sciences and Humanities Siedlce.

## 4 Studies of the effectiveness of the parallel program POKRMB

Experiments with our program POKRMB was conducted on pseudo Boolean Matrices using specially developed for the purpose of the program generator, which is part of the program P1.

In the current study the following was found.

Sequential program POKRMB' on a personal computer (CPU Intel Pentium M 1,70 GHz, 2GB RAM, Microsoft Windows XP Professional SP3) builds the shortest coverings for BM , which dimensions do not exceed 265 million elements. The maximum dimension of BM was equal to BM ***16300 * 16300***, i.e. number of rows (***M***) and the number of columns (***N***) Boolean matrix equal to ***M = N = 16300***. Our parallel program POKRMB using the cluster of NHU Siedlce allows to increase the "ceiling" of the dimension solved problems to MB with 350 million elements.

Our cluster has the following components:

- administrative server HP RX2600 (two processors Intel Itanium2 with a clock frequency of 1,3 GHz, cache size L2 3MB and 4GB RAM);
- 17 computer servers HP RX2600 (per server - two processors Intel Itanium2: 1,3 GHz, cache L2 3MB, 2GB RAM);
- Firewall HP Compaq Prowiant ML370, (two processors Intel Xeon: 2,8 GHz, 2GB RAM);
- disk array for 1000 GB HP MSA 1000;
- two application servers HP Compaq Prowiant ML570 (four processors Intel Xeon: 2 GHz, 8GB RAM).

Primarily the acceleration of calculations by using parallel program POKRMB was investigated.

An acceleration of computations using a parallel program is measured using the formula $\boldsymbol{S_m (n) = T_1 (n) / T_m (n)}$, where $\boldsymbol{T_1(n)}$ - time to solve the problem using a single processor, $\boldsymbol{T_m(n)}$ - time required to solve the problem with by ***m*** processing processors, ***n*** - the main parameter input problem [4]. The above formula shows that $\boldsymbol{S_m(n) \leq m}$ **.**

When the number of processing processors was increased, as expected, was significantly reduced the building time of the shortest covering of the original BM M. However, the acceleration of the computing was not proportional to the number of processing processors working simultaneously.

The dependence of the time constructing the shortest covering of ***700*5000*** - BM with the probabilistic occurrence of "unit" among the elements of the BM equal to ***0,2*** from the number of used processing processors is shown by Table 1. The table used the notation: ***m*** the number of used processors in calculations; |***R(m)***| - number of elements in the covering BM, constructed by summing the shortest coverings for submatrices; $\boldsymbol{Z_m}$ - the number of redundant elements in the coating ***R(m)***; |$\boldsymbol{R_m(m)}$| - number of elements in the shortest covering of BM; $\boldsymbol{T_m}$ - time of constructing the shortest covering of BM with by ***m*** processing processors.

**Table 1.** Dependence of the time constructing the shortest covering of BM from the number of used processing processors

| m | \|R(m)\| | $Z_m$ | \|$R_m$ (m)\| | $T_m$ |
|---|---|---|---|---|
| 1 | 11 | 0 | 11 | 25.6301 |
| 2 | 16 | 6 | 10 | 10.5765 |
| 3 | 19 | 8 | 11 | 6.57365 |
| 4 | 25 | 14 | 11 | 4.55367 |
| 5 | 28 | 16 | 12 | 4.14462 |
| 6 | 30 | 20 | 10 | 3.69568 |
| 7 | 31 | 21 | 10 | 2.90172 |
| 8 | 34 | 24 | 10 | 3.22919 |
| 10 | 33 | 23 | 10 | 2.58431 |
| 12 | 39 | 28 | 11 | 2.42903 |
| 14 | 41 | 30 | 11 | 2.12174 |
| 16 | 42 | 31 | 11 | 2.15647 |
| 18 | 44 | 33 | 11 | 2.32607 |
| 20 | 47 | 36 | 11 | 1.99392 |

We have received ***12*** < $S_{20}$ < ***13*** for BM with ***700*5000*** elements and ***p= 0,2.*** An acceleration of computations using a parallel program POKRMB was observed in the construction of the shortest coverings for all studied BM.

Figure 1 shows the time (in seconds) for constructing the shortest covering of BM (dimensions ***5000*10000, 10000*10000*** and ***10000 *15000*** with the probabilistic occurrence of "unit" among the elements of the BM equal to ***0.05***) from the number of used processors.

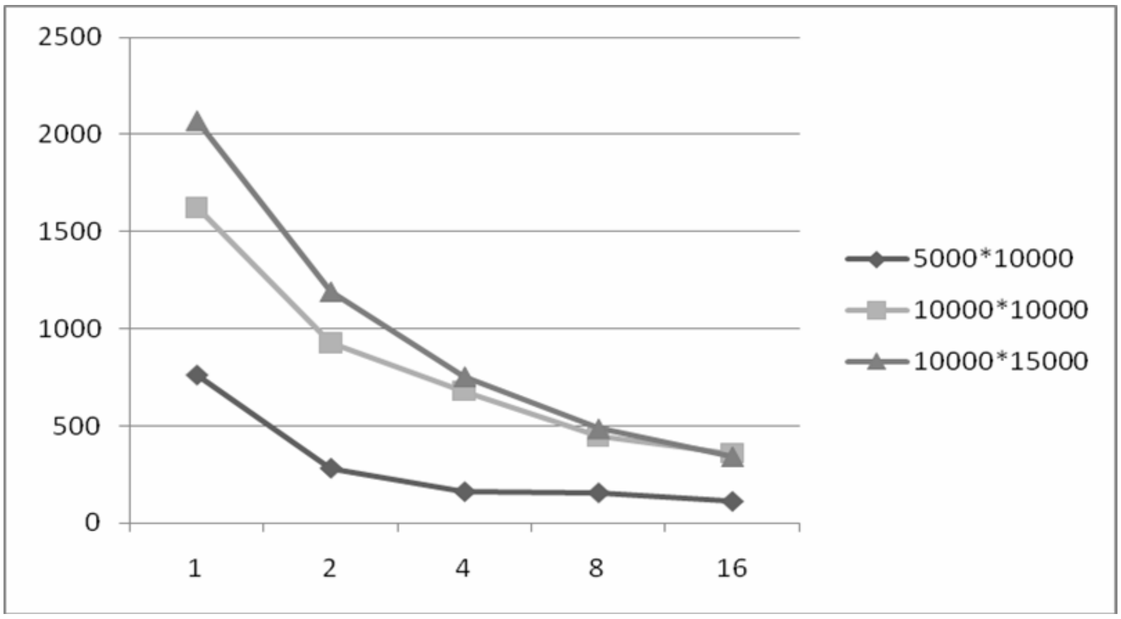

**Figure 1.** Dependence of the time constructing the shortest cover of MB from the number of processors

To study the efficiency of the parallel program POKRMB were generated BM with parameters ***M*N***: 1) ***5000*10000*** (50 mln.); 2) ***10000*10000*** (100 mln.); 3) ***10000*15000*** (150 mln.); 4) ***10000*20000*** (200 mln.); 5) ***12500*20000*** (250 mln.); 6) ***15000*20000*** (300 mln.) with the probabilistic occurrence of "unit" among the elements of BM to 1) $p = 0,05$; 2 ) $p = 0,1$; 3) $p = 0,25$; 4) $p = 0,4$; 5) $p = 0,5$. For each of the 30 generated BM the shortest coverings with the use of ***m*** processing processors, where $m \in \{1,2,4,8,16\}$, have been constructed**.** There were more than 150 experiments performed.

The dependence of time solutions on the dimension of the problem (***50, 100, 150, 200, 250, 300*** million cells) and the probabilistic occurrence of "unit" among the elements of BM is shown in Figure 2 (for $m = 4$), Table 2 (for $m = 8$) and Table 3 (for $m = 16$).

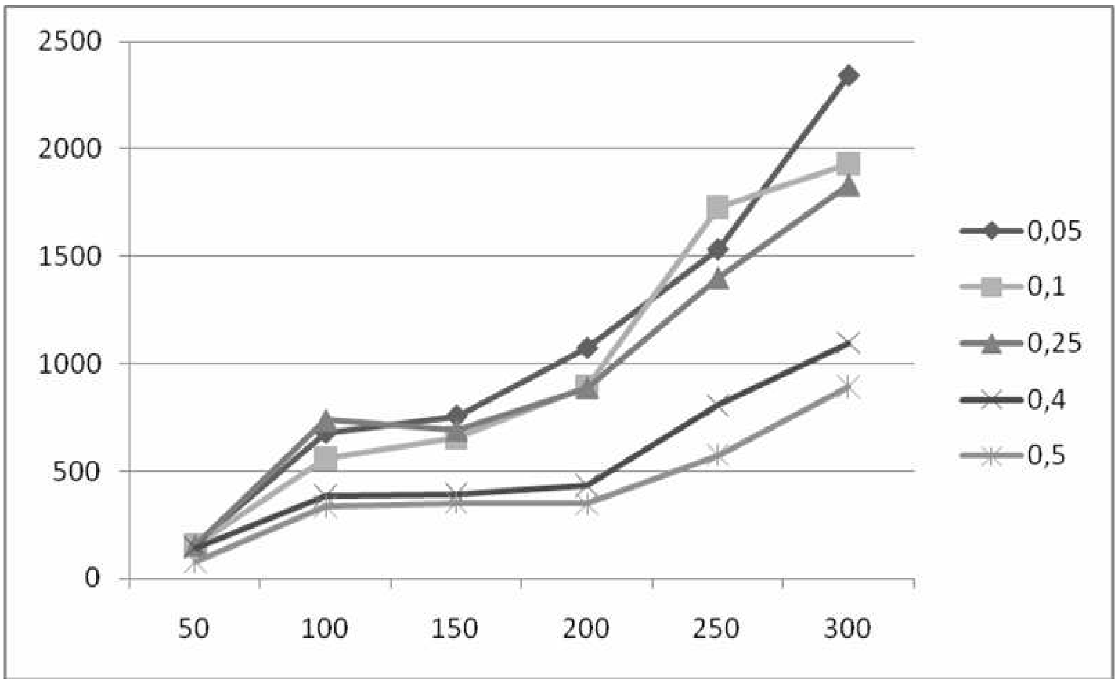

**Figure 2.** Dependence of time solution on the dimension from BM for m = 4

**Table 2.** Dependence of time solution from the dimension of BM for m = 8

| Dp | 50 | 100 | 150 | 200 | 250 | 300 |
|---|---|---|---|---|---|---|
| 0,05 | 155,56 | 444,351 | 488,085 | 680,996 | 945,579 | 1313,12 |
| 0,1 | 174,265 | 595,365 | 519,118 | 873,541 | 1247,71 | 1573,08 |
| 0,25 | 142,423 | 468,215 | 599,43 | 773,15 | 1375,71 | 2112,31 |
| 0,4 | 119,859 | 370,778 | 382,322 | 425,224 | 620,992 | 981,086 |
| 0,5 | 93,0742 | 367,937 | 296,675 | 396,83 | 484,789 | 722,821 |

**Table 3.** Dependence of time solution from the dimension of BM for m = 16

| Dp | 50 | 100 | 150 | 200 | 250 | 300 |
|---|---|---|---|---|---|---|
| 0,05 | 111,212 | 359,673 | 345,028 | 557,59 | 821,949 | 1287,52 |
| 0,1 | 111,883 | 407,034 | 430,06 | 522,003 | 1081,32 | 1306,05 |
| 0,25 | 126,112 | 442,16 | 431,397 | 747,425 | 998,739 | 1485,33 |
| 0,4 | 113,771 | 350,711 | 361,365 | 488,415 | 776,691 | 1171,41 |
| 0,5 | 78,9964 | 303,274 | 258,298 | 383,609 | 546,983 | 917,184 |

The time of constructing of the shortest covering increases much faster with the number of rows of BM. Figure 3 shows the dependence of the time solution (in seconds) from ***M*** - number of rows and ***N*** - number of columns of BM.

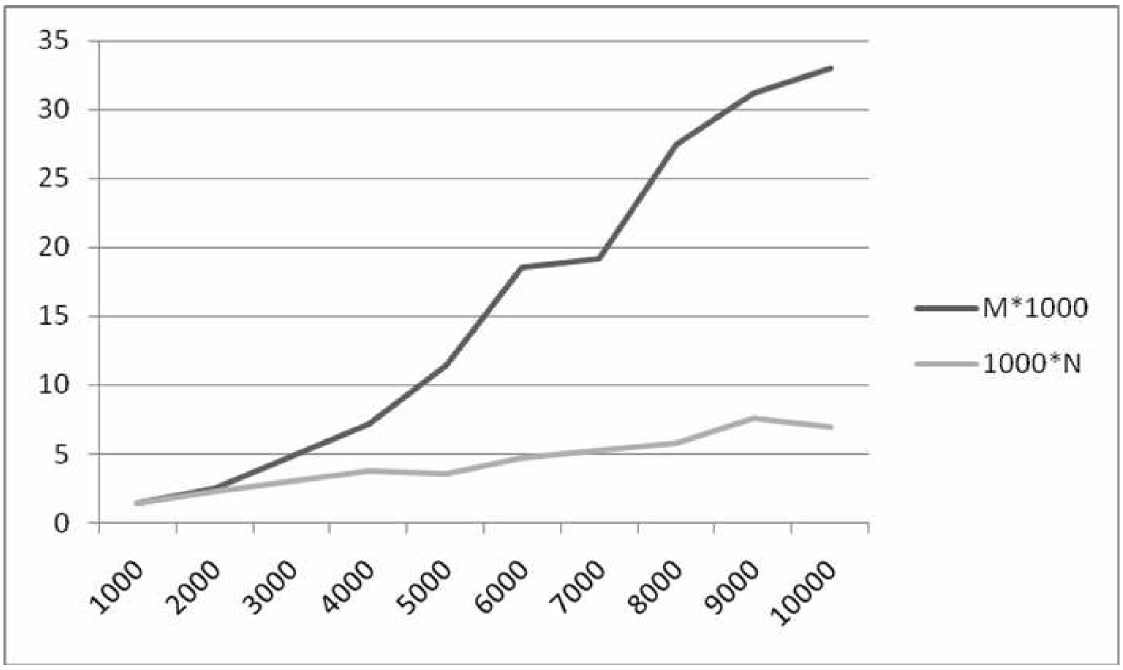

**Figure 3.** Dependence of time solution from the number rows and columns of BM for m = 16, p = 0,05

In our experiments, was not observed a significant reduction in the quality of solutions with increasing the number of parallel processors.

The sum of the shortest coverings for submatrices which is obtained in order to constructing the shortest coverings of the original BM often has redundant elements. The redundancy depends on the number of used processors and the probabilistic occurrence of "unit" among the elements of the BM. This redundancy increases with increasing values of ***m*** and ***p***. For example, when was got the shortest covering of BM with dimension ***5000*10000*** for the case ***p = 0,05***, ***m = 2*** the redundancy was 50%, and for the case ***p = 0,5***; ***m = 16*** it was 725%. However, all the redundant elements are eliminated by our program P6.

Different versions of the partition of the original BM at ***2, 3, ..., 16, …, m*** submatrices allow to construct different shortest coverings of the original BM with different numbers of elements (Table 1). From these coverings we can easily choose the best solution. The number of elements in the optimal (shortest) covering of the MB is equal to

$$|\, Ropt \,| = Min\ (|\, R_1 \,|, |\, R_2 \,|, ...., |\, R_m \,|).$$

Table 4 shows the results which allow to compare the quality of the shortest coverings constructed using POKRMB for different variants of the partition on ***4, 8, 16*** submatrices of MB with the parameters ***5000*10000*** and the probabilistic occurrence of "unit" among the elements equal to ***0.05*** and ***0,1*** ; ***0,25***; ***0,4***; ***0,5***.

**Table 4.** Dependence of the quality of the shortest covering of the BM from a version of the partition on submatrices

| | 0,05 | 0,1 | 0,25 | 0,4 | 0,5 |
|---|---|---|---|---|---|
| $\|R_{opt}\|$ | 24 | 16 | 9 | 8 | 7 |
| $\|R_4\|$ | 24 | 16 | 10 | 8 | 8 |
| $\|R_8\|$ | 24 | 16 | 9 | 8 | 8 |
| $\|R_{16}\|$ | 25 | 17 | 9 | 8 | 8 |

The ratio of the number of elements in the "best" solution to the number of elements in the "worst" solution, marked with $\boldsymbol{v}$, in the experiments varied in the range $\boldsymbol{0,875 \leq v \leq 1}$. In 20% of cases it was found that for all the variants of the partitions of BM were constructed coverings with the same number of elements (that is, all coverings were optimal). The worst decision only  for 23%  cases was worse by more than one item compared  the "best" solution. More than in half of generated BM the best solution was obtained for partitioning the original BM at 16 submatrices. In these cases we  get the best (optimal) solution in minimal time.

## 5 Conclusion

Our studies suggest the following conclusions.

Using of parallel program POKRMB and cluster allowed to raise the "ceiling" dimension of the task for more than 30%.

The acceleration of computations using a parallel program POKRMB and cluster depends on the number of used processing processors and the probabilistic occurrence of "unit" among the elements of the BM. In the experiments in the case $\boldsymbol{m = 16}$ the acceleration of calculations was varied in the range $\boldsymbol{3 \leq S_{16} \leq 12}$.

The maximum acceleration of calculations was obtained for BM, where a probabilistic  occurrence of "unit" among the elements of BM is equal to ***0.05***. In these cases the submatrices ***MBi***, obtained by dividing the initial BM, contain many zero rows.

The time constructing of the shortest coverage depends on the dimension of BM and a probabilistic  occurrence of "unit" among its elements.

In our experiments with increasing a dimension of the BM to six times (and the same probabilistic  occurrence of "unit" among the elements of the BM) time solving of the problem has increased eight times for $\boldsymbol{m = 8}$ and in ***11.6*** times in the case $\boldsymbol{m = 16}$.

The time solution of our problem is reduced with increasing probabilistic occurrence of "unit" among the elements of the BM (with the same dimension) from

***0,05*** to ***0,5***. The maximum reduction of the time (more than 3-fold) was observed for BM with the dimension ***10000*20000*** and ***m = 4***.

The time constructing of the shortest covering of BM much faster increases with increasing the number of rows of BM.

In our experiments, was not observed significant reduction in the quality of solutions with increasing number of parallel processing processors. The maximum decline in the quality did not exceed 14%. Reducing the quality was not observed in the construction of the shortest coverings by ***16*** processing processors for more than half of generated BM. In these cases we proved that the quality of the solution and time were optimal.

## References

1. A. Zakrevskij, Yu. Pottosin, L.Cheremisinova. Combinatorial Algorithms of Discrete Mathematics. -Ed. A. Keevalik.-Минск, 2009.
2. С.В. Новиков. Распараллеливание вычислений при решении двух построения оптимальных покрытий. Вестник Гродненского университета, Гродно, серия 2, 1(92), 2010, стр. 30-36.
3. Adam Adamus. Równoległy program znalezienia minimalnego pokrycia macierzy Boole'a o dużych rozmiarach. Praca magisterska. Akademia Podlaska, Siedlce, 2010.
4. В.П. Гергель. Введение в методы параллельного программирования. Нижний Новгород, 2007, http://www.software.unn.ru/ccam/multicore/materials/tb/mc_ppr04.pdf