ORIENTAL JOURNAL OF
COMPUTER SCIENCE & TECHNOLOGY
An International Open Free Access, Peer Reviewed Research Journal
Published By: **Oriental Scientific Publishing Co., India.**
www.computerscijournal.org

ISSN: 0974-6471
December 2015,
Vol. 8, No. (3):
Pgs. 255-269

Approach To Solving *K-sat* Problem Based On Reduction Thereof to the Covering Problem

S. V. LISTROVOY[1] and A. V. SIDORENKO[2]

[1]Doctor of Engineering Science, professor in the Ukrainian State
Academy of Railway Transport (61050, 7, Feuerbach square, Kharkov, Ukraine.
[2]Principal Software Engineer of "Stalenergo" Scientific-Production Enterprise
(61105, 9, Fedorenko St., Kharkov, Ukraine.

(Received: August 16, 2015; Accepted: November 28, 2015)

ABSTRACT

We propose an exact polynomial-time algorithm for solving *SAT* the problem.

Key words: SAT problem, polynomial reducibility.

INTRODUCTION

SAT-problem is important in systems of evidence automatic verification, where formula is a set of clauses, which mean disjunction of a certain amount of literals – variables $\tilde{O}$ and $\overline{\tilde{O}}$. This problem has a great importance upon determination of satisfiability of CIRCUIT-*SAT* schemes (circuit-satisfiability problem) and in pattern recognition problems. The international annual conference (The International Conference on Theory and Applications of Satisfiability Testing) that has been held every year for more than ten years, as well as a (Journal on Satisfiability, Boolean Modeling and Computation) are devoted to this problem. An important place in the study of *SAT*-problem takes design of programs to address them; such programs are called *SAT*-solvers. Modern *SAT*-solvers are able to quickly solve many problems that were considered not solved a few years ago. There are many exponential algorithms of its solving and heuristic approaches of polynomial complexity. Among them one should mention the Monien and Shpikermayer algorithm, 1985, in which a simple search is used for solving *3-SAT* problem: alternate substitution of each variable with 1 or 0 is tried and then problem of a smaller size is recursively solved; it has temporal complexity $\hat{I}(1{,}84^n)$ and algorithm for problem of propositional satisfiability of formulas in conjunctive normal form with the complexity of $\hat{I}(1{,}074^n)$, obtained by Hirsch in 2000. In general, there are two main types of algorithms for solving

SAT problems: local search algorithms that start with a certain set of values (though, it does not satisfy all the formula), and then modify it trying to consistently get closer to performed set, and so-called *DPLL*-algorithms (by names of inventors: Davis, Putnam, Logemann, Loveland; description of basic principles of this method dates back to 1968), which traverse a tree of all possible sets and perform depth-first search. The purpose of this paper is to develop efficient exact algorithm for solving *3-SAT*-problem and an arbitrary *k-SAT*-problem of polynomial complexity.

Formalization of SAT problem and its solution

Let's consider Boolean function

$f(x_1, x_2, .., x_n)$ in conjunctive form

$$f(x_1, x_2, .., x_n) = (x_1^{s_{11}} \vee x_2^{s_{12}} \vee ... \vee x_n^{s_{1n}}) \wedge ... \wedge (x_1^{s_{m1}} \vee x_2^{s_{m2}} \vee ... \vee x_n^{s_{mn}})$$

where

$$x_i^s = \begin{cases} x_i, -\text{ïðè } s = 1 \\ \overline{\delta}_i - \text{ïðè } s = 0 \end{cases}.$$

Operations Ú, Ù are Boolean and simulate simple logic statements: Ú– "OR"; Ù– "AND". For any binary set $\tilde{\sigma} = (x_1, x_2, .., x_n)$ the function takes one of two possible values: one or zero. The problem "satisfiability" is the answer to the question: whether there is a set of variables $\tilde{\sigma} = (x_1, x_2, .., x_n)$, reversing function *f* into one.

As shown in [1] *SAT* problem can be considered as covering problem, for this let's construct by Boolean function a Boolean matrix *B* in which columns correspond to variables $(\delta_1, \delta_2, ..., x_n)$ and $(\overline{\delta}_1, \overline{\delta}_2, ..., \overline{x}_n)$, and rows correspond to disjunctions of the Boolean function. In general, the number of columns in matrix $\hat{\boldsymbol{A}}$ is equal to 2*n*, and the number of rows is equal to the number of disjunctions *m* in the Boolean function.

For example, for Boolean function

$$F = (x_1 \vee x_2 \vee x_3)(\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3) \cdot (x_1 \vee \overline{x}_3)(x_3 \vee \overline{x}_1)(x_1 \vee \overline{x}_2), \quad ...(1)$$

Let's numerate disjunctions of the Boolean function, see Table 1.

Numeration of disjunctions

Table 1:

1- $(X_1 \vee X_2 \vee X_3)$ 2- $(\overline{X}_1 \vee \overline{X}_2 \vee \overline{X}_3)$

3- $(X_1 \vee \overline{X}_3)$ 4- $(X_3 \vee \overline{X}_1)$ 5 $(X_1 \vee \overline{X}_2)$,

Then, matrix *B* will have the form (2)

$$B = \begin{array}{c|cccccc} & X_1 & X_2 & X_3 & \overline{X}_1 & \overline{X}_2 & \overline{X}_3 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 & 1 & 1 \\ 3 & 1 & 0 & 0 & 0 & 0 & 1 \\ 4 & 0 & 0 & 1 & 1 & 0 & 0 \\ 5 & 1 & 0 & 0 & 0 & 1 & 0 \end{array} \quad ...(2)$$

Columns corresponding to the variables X_i è $\overline{X}_i$ in the matrix will be called inverse. If in matrix $\boldsymbol{B}$ there is a covering of rows by units belonging to non-inverse column, it means that *f* function is satisfiable, if there is no such covering, it is unsatisfiable. Let's denote variables by , then matrix $\hat{\boldsymbol{A}}$, for the Boolean function (1), takes the form

$$B = \begin{array}{c|cccccc} & X_1 & X_2 & X_3 & Z_1 & Z_2 & Z_3 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 & 1 & 1 \\ 3 & 1 & 0 & 0 & 0 & 0 & 1 \\ 4 & 0 & 0 & 1 & 1 & 0 & 0 \\ 5 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}, \quad ...(3)$$

Where if, then =0 and if , then =1

As shown in [1], the problem of minimum coverage for any matrix $\boldsymbol{B}$ defined by some Boolean function

$$f = (X_l \vee X_b \vee ... \vee X_k)(X_s \vee X_r \vee ... \vee X_t)...(X_q \vee X_d \vee ... \vee X_h) \quad ...(4)$$

Can be considered as a problem of

finding a minimum set of variables {Xi = 1}, where the Boolean function (1) is satisfiable. That can be written in the following form

$$\min_i \{X_i = 1\} \qquad ...(5)$$

upon constraints

$$(X_l \vee X_b \vee ... \vee X_k)(X_s \vee X_r \vee ... \vee X_t) ...(X_q \vee X_d \vee ... \vee X_h) = 1 \qquad ...(6)$$

If we go to the dual Boolean function we'll obtain

$$\min_i \{X_i = 0\} \qquad ...(7)$$

$$X_l X_b ... X_k \vee X_s X_r ... X_t \vee ... \vee X_q X_d ... X_h = 0 \qquad ...(8)$$

From (7-8) it follows that the problem of the least coverage can be considered non-linear Boolean programming problem that means finding of the smallest number of variables turning into zero the left side of the constraint (8). If we are talking about the existence of at least one covering, then it is necessary to find out whether there is at least one set of variables turning into zero the left side of the constraint (8). For the matrix ***B***, defined by correlation (3), condition for the existence of at least one covering will have the form

$$X_l Z_b ... X_k \vee X_s Z_r ... X_t \vee ... \vee X_q Z_d ... Z_h = 0 \qquad ...(9)$$

upon constraints

if $X_i = 1$, then Z_i =0 and

if $X_i = 0$, then Z_i =1 ...(10)

Thus, to establish the satisfiability of the Boolean function, we need to show the presence of at least one covering of rows by units in the matrix ***B***, defining a given Boolean function, defined by correlation (3) that satisfies condition (10). For this, it is necessary to solve non-linear Boolean equation (9) and therewith, solution of this equation should satisfy condition (10). If we denote an arbitrary summand by , and the number of summands by *m*, then the problem (9-10) can be written as

$$\sum_{i=1}^{m} S_i = 0 \qquad ...(11)$$

upon constraints if $X_i = 1$, then Z_i =0 and if $X_i = 0$,then Z_i =1 ...(12)

In order to solve problems (11-12), variables of the Boolean function x_i^s , taking values X_i , and Z_i we'll characterize by weight characteristics h_i^x and h_i^z showing the frequency of appearance of variables and in summands of equation (11).

In solving the problem (11-12) for the conversion of some Boolean function we in mean a change in the function by substituting in it pairs, ,= 0 or ,= 1, and since as a result of substitution or = 1 appeared clauses with fewer variables, We will produce the absorption type of operation. If the result of substituting in it pairs, ,= 0 or = 1, it follows that the equality, t.e. ravening (11) takes the form 1 = 0, then we say that there was a contradiction and the function "not feasible" . If the conversion process appears clauses, consisting of one variable, these variables are assumed to be zero, and enter them in the decision.

The basic idea of the solution of equation (11) is the choice of a clauses in the original in the original Boolean function *F*, and is it possible to reset all the other terms in (11) due to the fact that we assume to be zero in turn variables belonging to the selected clauses in this form some intermediate functions in which, too, will allocate clauses and try to reset all the terms in these functions on the basis of a clause in the selection of one of these functions and setting variables in the selected clauses of turn zero. This process continues until we get the identity 0=0 or , proceed to the contradiction 1= 0. In the first case of the function is feasible, and the second is not feasible. In general, the whole set of functions formed on the basis of one clauses selected in the initial Boolean function *F* can be represented as a tree, see the following. Figure 1.

In Figure 1. shows a tree of all F_r, who will have to build for the solution of «*k-SAT*» for *k* = 5. To construct the procedure of forming wood introduce the following notation: - clause in the

r- level in the tree; - the number of the variable in the clauses on the *r*- level in the tree; - is a variable in a number of clauses. In view of the above notation, consider the following procedure ***A***, the definition of the solution of (11) with the conditions (12) for the solution of «*k-SAT*» problem, i.e., the feasibility of a task in which each contain clauses on *k*-variables.

Procedure A

Step 1. Select the equation (11) a clause with a minimum number of variables and with the greatest total weight characteristics, choose a variable in it with the highest frequency (this corresponds to the zero level r = 0) and move on to the next step.

Step 2. Assign a variable selected clauses zero, and convert a Boolean function to function and proceed to the next step

Step 3. Check function, "not feasible" and whether there is a disjoint variables that are not considered to be zero (ie. check of the conversion process

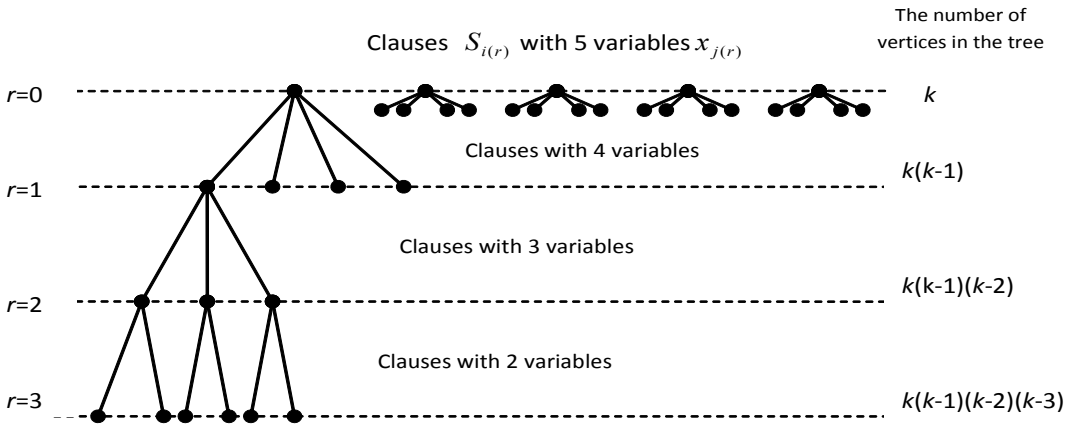

Fig. 1: Tree forming Boolean functions Fr

Table 2:

x_i^σ	x_0	x_1	x_2	x_3	Z_0	Z_1	Z_2	Z_3
h_i^x	5	5	5	3	6	3	5	6

Table 3:

x_i^σ	x_1	x_2	x_3	Z_1	Z_2	Z_3
h_i^x	3	1	1	1	2	1

does not result in a contradiction of the form) If yes, proceed to step 2, otherwise the next step.

Step 4: Check the function of "not feasible", and whether there is a disjoint variables that were assumed to be zero (i.e. Check the conversion process functions to does not lead to a contradiction of the form) If yes, proceed to Step 2, otherwise the case in the next step.

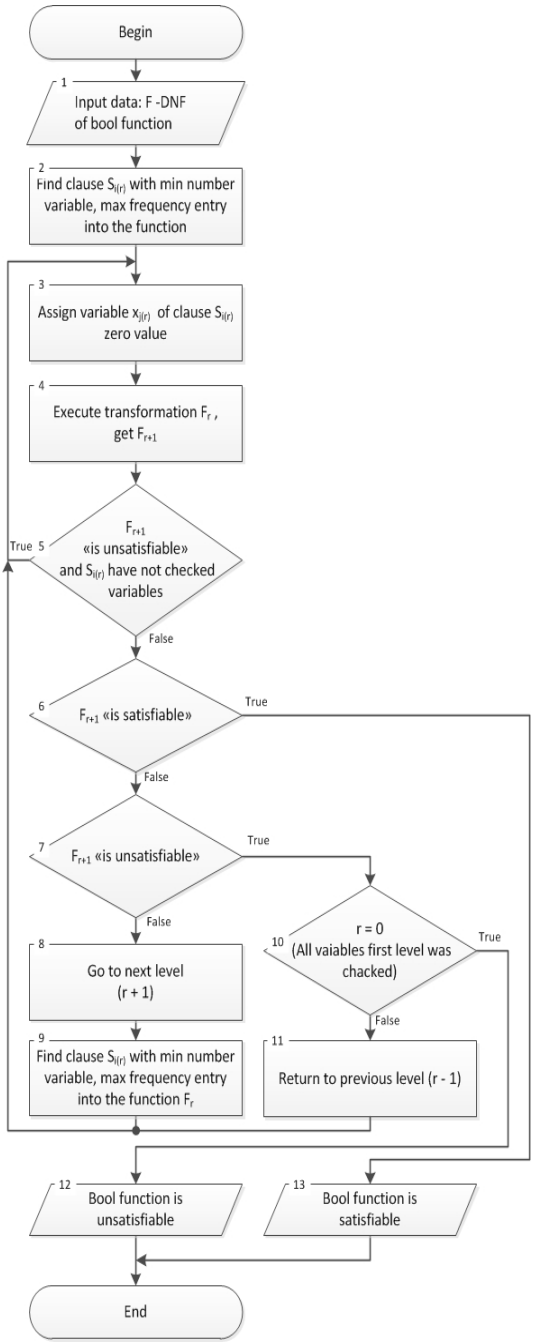

Fig. 2: The block diagram of an implementation procedure A

Step 5: Check the function "feasible" or not (i.e. identity is performed 0=0 in equation (11)), and if so then the algorithm terminates because of the function is feasible, otherwise the next step.

Step 6: Check the function of "not feasible", that is a contradiction of the form when trying to function in zero if the function is turned to zero and thus no contradiction, then go to the next step if the controversy arose, that is, the function "not feasible", then go to step 7.

Step 7: Go to the next level to find this clause with a minimum number of variables and with the greatest total weight characteristics, select it in the variablewith the highest frequencyin the Boolean functionand go to step2.

Step 8: Checking were tested at zero (r = 0), all variables on the ability to reset clauses in equation (11), if yes, then the algorithm terminates, since the function "not feasible" otherwise the next step.
Step 8. Back to the previous (r-1) level branching and go to step 2.

The block diagram of this algorithm is presented in Fig.2. Let's show that procedure ***A***, oriented at solving of *k-SAT* problem, allows determination for polynomial time whether there is a set of variables in equation (11) allowing turning this equation into an identity upon fulfillment of condition (12). If in equation (11) we take arbitrary summand = , it is clear that in order to make =0 and fulfill condition (12) one of k pairs of equalities =0 , =1; =0, =1; …=0, =1 should be fulfilled, i.e. in the set of variables of problem (11-12) solution there is one of these pairs and procedure ***A*** alternately checks which of these pairs belong to the solution. When we choose a clause in original Boolean function F procedure ***A*** does not analyze the entire original Boolean function F, and only that part of clauses which crosses the variables selected clauses. Sequences clauses of Boolean functions, which do not intersect the variables included in the clauses that make up the sequence, they will be referred to

independent sequences of clauses. It is clear that if a Boolean function is the number of variables is 2*n*, and the number of variables in each clause is k, the maximum number of such independent sequences cannot exceed the value,

$\frac{2n}{k}$ i.e. in the worst based on the proposed procedure in the original Boolean function need to zero no more such sequences and therefore have to analyze no more trees. So appreciate the complexity of the analysis of one such tree. In accordance with the constructed tree, the number of functions that will have to be converted equal to the total number of nodes at all levels of the analyzed tree.

As can be seen from Fig.1.the number of nodes at level zero will be equal to k at the first level is *k*(*k*-1) at the third level *k*(*k*-1)(*k*-2) and etc. And so, the number of nodes at level zero is equal to *k* , the second the number of vertices does not exceed ,will not exceed the third and the last level will not exceed . Consider the amount of clear that it cannot exceed the value. For example, for the task "3-*SAT*" in the tree will have to be converted in the worst case *k*+ *k*(*k*-1)+ *k*(*k*-1)(*k*-2)=15 =3+3(3-1)+3(3-1)(3-2)=15 functions. Given that the solution to "*k-SAT*" problem have to analyze trees, in the worst case will have to convert functions. In the case of a decision "3-*SAT*" problem, this number will not exceed the amount. For the analysis of the original equations and arbitrary Boolean functions in accordance with the proposed ***À*** procedure u need different operations. The number *u* is determined 2*mn* a comparison operation to determine the frequency of occurrence of each variable in the selected term and other terms plus *m'"k* operations of addition to determine the total value weight of each term in equation (11) and plus operations comparison of the maximum element in the array of *m* elements, i.e. . In the worst case at each step of the procedure ***A*** will be reset to zero only one summand i.e. in equation (11) after the first step will remain *m*-1 summand further *m*-2, *m*-3,...,1 summands ,i.e. zeroing all components of the procedures ***A*** in the worst case will require completion operations.

Thus, time complexity of work of the proposed procedure ***A*** does not exceed

$$O(\frac{2k^{k-1}m\ (m+1)}{2}u) = O(k^{k-1}m\ (m+1)$$
$$(2m\ + km\ + m\log_2 m) \approx O(2m^3n^2k^{k-1}$$
$$(1+\frac{k+\log_2 m}{2n}) \qquad ...(13)$$

In the case of a decision "3-*SAT*" problem the number of elementary operations in the worst case may not exceed

$$O(0\ m^2(m+1)n^2(1+\frac{3+\log_2 m}{2n}) \approx$$

Table 4:

x_i^{σ}	x_0	x_1	x_2	x_3	Z_0	Z_1	Z_2	Z_3
h_i^x	6	5	4	4	5	3	6	4

Table 5:

x_i^{σ}	x_1	x_2	x_3	Z_1	Z_2	Z_3
h_i^x	2	2	1	2	3	2

$$O(\mathbb{0}\ m^3 n^2 (1+\frac{3+\log_2 m}{2n}) \quad ...(14)$$

It is known that "*k-SAT*" problem can be reduced to "3-*SAT*" problem in polynomial time, in particular in [9] it is shown that if the clause contains more than three literals

$$S_i = (l_1 + l_2 + ... + l_k)\ k>3 \quad ...(15)$$

it can be replaced by (*k*-2) clause

$$S_i = (l_1 + l_2 + x_1)(\overline{x_1} + l_3 + x_2)$$
$$(\overline{x_2} + l_4 + x_3)\cdot ... \cdot (\overline{x_{k-3}} + l_{k-1} + l_k) \quad ...(16)$$

where new variables, with this new set of clauses satisfied if and only if the following clause. Thus the transition from "*k-SAT*" problem to a "3-*SAT*" each clause "*k-SAT*" problem increases the number of variables in Boolean functions on (*k*-3), and the number of clauses are restricted to (*k*-2) clause, and therefore, when the number of variables and clause in a new task will be respectively equal $n' = (n + m(k-3)); m' = (k-2)m$.

Substituting and as new values *n* and *m* in (14) we obtain an estimate of difficulty with this approach to the solution of «*k-SAT*» task using the procedure ***A***

$$O(\mathbb{0}\ (k-2)^3 m^3 (n+m(k-3))^2$$
$$(1+\frac{3+\log_2(k-2)+\log_2 m}{2(n+m(k-3)}) \quad ...(17)$$

Proposed procedure for solving «*k-SAT*» problem is formally polynomial algorithm but with a high degree of the polynomial (13) and consequently implemented in exponential time, with the solution of the «3-*SAT*» A procedure performs tasks in polynomial time. Thus, the transformation «*k-SAT*» problem in polynomial time in the «3-*SAT*» problem leads to an algorithm of polynomial complexity.

In solving «*k-SAT*» task using the procedure _ without reducing it to "3-*SAT*» problem we have exponential complexity of the algorithm and the choice at each step of the procedure ***A*** disjuncts with the minimum number of variables and with the greatest total weight characteristics can significantly to reduce the average time of the algorithm, by resetting the maximum number of terms in each step of the procedure. Thus, the procedure ***A*** is a

Table 6:

x_i^{σ}	x_0	x_1	x_3	Z_0	Z_1	Z_3
h_i^{x}	**2**	**1**	**2**	**3**	**1**	**2**

Table 7:

x_i^{σ}	x_0	x_2	x_3	Z_0	Z_2	Z_3
h_i^{x}	**3**	**2**	**2**	**2**	**3**	**2**

complete listing of options, may be reset clauses in the case of feasibility reset function occurs in polynomial time, and in the case of "not feasible" function after attempts to reset either the first or the *m* clauses it turns out that it can not reset and still going strong screening the number of functions in the tree that must be analyzed and this leads to the fact that the average procedure ***A*** decides «*k-SAT*» problem in polynomial time.

Let's consider examples of work of the procedure ***A***, upon determination of satisfiability of Boolean functions. Let it be required to determine satisfiability of the following Boolean function with four variables and thirteen disjunctions while we assume that the + sign is a sign of logical addition.

$f(x) = (x>_1+x>_0+x>_3)(x_1+x_0+x_2)(x>_0+x>_3+x>_2)$ $(x>_3+x>_2+x_1)(x_1+x_0+x_3)(x>_1+x>_0+x_2)$ $(x>_3+x_0+x_2)$ $(x>_1+x>_2+x_0)(x>_0+x_3+x_2)(x>_2+x_1+x_0)(x>_0+x>_3+x_1)(x>_3+x_1+x_2)(x>_0+x>_2+x_3)$ ...(18)

Let's write initial equation

Table 8:

m = 500; k = 3						
n	100	110	120	130	140	150
Number of elementary operations	$1.06\cdot10^8$	$7.2\cdot10^8$	$8.6\cdot10^9$	$4.2\cdot10^8$	63,209	52,096
	0.2	0.22	0.24	0.26	0.28	0.3
Satisfiability	-	-	-	+	+	+
Minimum time of solution – 5.06 ms				Maximum time of solution – 793,934 ms		

Table 9:

m = 500; k = 3						
n	400	800	1200	1600	2000	2400
Number of elementary operations	286987	106	2.24·106	4·106	6.2·106	8.9·106
	1.25	0.625	0.416	0.313	0.25	0.208
Satisfiability	-	-	-	+	+	+
Minimum time of solution - 36.5 ms				Maximum time of solution - 849 ms		

Table 10:

m = 90; k = 3						
m	100	200	300	400	500	600
Number of elementary operations	16029	26994	37413	4.39·107	2.42·107	1.86·107
	0.90	0.4	0.3	0.23	0.18	0.15
Satisfiability	+	+	+	+	-	-
Minimum time of solution - 2.15 ms				Maximum time of solution - 38,910 ms		

Table 11:

m = 500; n = 90, ? = 18						
k	3	10	20	30	40	50
Number of elementary operations	2.42·107	60,807	48,555	55,488	64,268	74,508
	0.2	0.22	0.24	0.26	0.28	0.3
Satisfiability	-	+	+	+	+	+
Minimum time of solution - 7.57 ms				Maximum time of solution - 160 ms		

$Z_1Z_0Z_3 + x_1x_0x_2 + Z_0Z_3Z_2 + Z_3Z_2x_1 + x_1x_0x_3 +$
$Z_1Z_0x_2 + Z_3x_0x_2 + Z_1Z_2x_0 + Z_0x_3x_2 + Z_2x_1x_0 + Z_0Z_3x_1 +$
$Z_3x_1x_2 + Z_0Z_2x_3=0;$

...(19)

We determine frequencies of appearance of variables in each summand, see table 2

Let's choose summand with $Z_0Z_3x_1$ maximum value of sum of frequencies h_i^x equal to 6+6+5=17 and assume $Z_0=0$; $x_0=1$, thus initial equation (19) will take the following form

$x_1x_2 + Z_3Z_2x_1 + x_1x_3 + Z_3x_2 +$
$Z_1Z_2 + Z_2x_1 + Z_3x_1x_2=0$ ...(20)

we perform absorption, here x_1x_2 absorb summand $Z_3x_1x_2$, and summands Z_2x_1 and Z_3x_2 absorb accordingly summands $Z_3Z_2x_1$ and $Z_3x_1x_2$, thus equation (20) will take the following form

$$F_{r=1} = x_1x_2 + x_1x_3 + Z_3x_2 + Z_1Z_2 + Z_2x_1 =0$$

...(21)

Check the function is reversed to zero, i.e. "doable" or not. In this case, the function is not converted to zero, and in the process of transformation functions in contradictions have arisen, therefore, proceed on the next *r*+1 level.

Again we determine frequencies of appearance of variables in each summand, see table 3.

Further, among summands containing the minimum number of variables, such summands are x_1x_2; x_1x_3; Z_3x_2; Z_1Z_2; Z_2x_1, we choose the summand with the highest weight characteristics δ. In accordance with Table 3, weight characteristics of these summands are, respectively, (4, 4, 2, 3, 5), so we choose the summand Z_2x_1. There variable with the greatest frequency equal to 3 is variable x_1, so we assume $x_1=0$; $Z_1=1$. Thus, equation (21) will take the following form

$$F_{r=2} = Z_3x_2 + Z_2=0$$ ...(22)

Absorption in (22) cannot be done, but there appeared the summand with one variable, therefore, we believe $Z_2=0$; $x_2=1$ and therefore, equation (22) will take the following form

$Z_3=0$ ...(23)

Thus, a set of variables $Z_0=0$; $x_1=0$; $Z_2=0$; $Z_3=0$ or $x>_0$ x_1 $x>_2$ $x>_3$ draws equation (19) into the

Table 12:

m = 10,000; k = 3						
n	100	200	300	400	500	5,000
Number of elementary operations	324,459	1.25·106	4.89·106	9.5·107	2.36·109	6.1·107
	0.2	0.22	0.24	0.26	0.28	0.3
Satisfiability	-	-	-	+	+	+
Minimum time of solution - 44 ms				Maximum time of solution - 819,261 ms		

Table 13:

Test name	**Number of variables**	**Number of disjunctions**	**Average number of fulfilled operations upon solution**	**Average time of solution in ms**	**Minimum time of solution in ms**	**Maximum time of solution in ms**
CBS_k3_n100_m403_b0	100	403	$51 \cdot 10^6$	4,578	2.28	77,843
.....						
CBS_k3_n100_m403_b900						

identity and hence is a performing set of variables for the Boolean function (18). Let's also consider the process of work of procedure ***A*** in the case when Boolean function is not satisfiable. Let function be set as follows

$F(x)=(x_0+x_1+x_{2})(x_{3}+x_{0}+x_2)(x_0+x_{3}+x_2)(x_1+x_{2}+x_{0})(x_{0}+x_{1}+x_2)(x_3+x_1+x_{0})(x_3+x_0+x_{2})(x_0+x_{2}+x_{1})(x_3+x_0+x_2)(x_3+x_0+x_1)(x_3+x_{2}+x_{0})(x_{2}+x_{3}+x_{1})(x_1+x_{2}+x_{3})$ ...(24)

Let's write initial equation

$$F_{r=0}=x_0x_1Z_2+Z_3Z_0x_2+x_0Z_3x_2+x_1Z_2Z_0+Z_0Z_1x_2+x_3x_1Z_0+x_3x_0Z_2+x_0Z_2Z_1+x_3x_0x_2+x_3x_0x_1+x_3Z_2Z_0+Z_2Z_3Z_1+x_1Z_2Z_3=0 \quad ...(25)$$

Since we solve *3-SAT* problem we have k = 3 and before the start of work of procedure ***A*** variable $\tilde{n}$=1. We determine frequency h_i^x of appearance of variables in each summand (25), see table 4.

We choose summand with $S_1^* = x_0x_1Z_2$ with maximum value of sum of frequencies h_i^x equal to 6+5+6=17, choose variable with maximum value of , this is x_0 and assume; x_0=0, Z_0=1, thus initial equation (25) will take the following form

$$Z_3x_2+x_1Z_2+Z_1x_2+x_3x_1+x_3Z_2+Z_2Z_3Z_1+x_1Z_2Z_3=0 \quad ...(26)$$

In (26) summand x_1Z_2 absorb summand $x_1Z_2Z_3$ and equation (26) will take the following form

$$F_{r=1} = Z_3x_2+x_1Z_2+Z_1x_2+x_3x_1+x_3Z_2+Z_2Z_3Z_1=0 \quad ...(27)$$

The function $F_{r=1}$ becomes zero, and when this controversy arose therefore determined the frequency h_i^x of occurrence of variables in each clause (27) see table 5.

In (26) summand with maximum total value of frequency is x_1Z_2 with total weight equal to 5 and x_3Z_2, there we choose variable Z_2 with maximum weight 3. Further we assume Z_2=0, x_2=1, as a result (27) will take the following form

$$Z_3+Z_1+x_3x_1=0 \quad ...(28)$$

As in (28) acquisitions cannot be held, and the terms containing one variable, we assume Z_3=0, x_3=1 and Z_1=0, x_1=1, and obtain 1=0 i.e. there is a contradiction, so return to the previous level $F_{r=1}$ and move on to try to reset clause x_1Z_2 based on the variable x_1, for in this we assume x_1 =0, Z_1=1 in (28) will receive

$$F_{r=2} = Z_3x_2+ x_2+x_3Z_2+Z_2Z_3=0 \quad ...(29)$$

In (29) x_2 absorbs summand Z_3x_2 and we obtain the following equation

$$F_{r=3} = x_2+x_1Z_2+Z_2Z_1+x_1Z_2=0 \quad ...(30)$$

As in (30) appeared, the terms containing one variable, we assume x_2 =0,Z_2=1 in (30) then we get x_1+Z_1=0, i.e., there was a contradiction, i.e. reset clause x_1Z_2 by variables x_1 and Z_2 without any contradiction is impossible, so back to zero and check whether all variables are checked for the possibility of exempting clause $x_0x_1Z_2$. In this case, not tested variables remained the variables x_1 and Z_2 and thus the variable Z_2 has a greater frequency of occurrence in the clause than x_1, so we assume Z_2=0, x_2=1 in this case equation (25) takes the form

$$F_{r=1} = Z_3Z_0+x_0Z_3+Z_0Z_1+x_3x_1Z_0+x_3x_0+x_3x_0x_1=0 \quad ...(31)$$

In (31) summand x_3x_0 absorb summand $x_3x_0x_1$, and we obtain

$$F_{r=2} =Z_3Z_0+x_0Z_3+Z_0Z_1+x_3x_1Z_0+x_3x_0=0 \quad ...(32)$$

The function becomes zero, and when this controversy arose therefore determined the frequency of occurrence of variables in each clause (32) see table 6.

Among the terms with two variables, select the term with the maximum aggregate rate, this term is a term, this term Z_3Z_0 with a weight characteristic of 5, select it in the variable Z_0 with the largest weight to 4 and assume x_0 = 1 and Z_0 = 0, at the same time (32) will have the following form

$$Z_3+x_3=0 \quad ...(33)$$

i.e. there was a contradiction therefore return to the previous level to zero clause Z_3Z_0 based variable Z_3, for this purpose (32) believe $x_3 = 1$, $Z_3 = 0$, we obtain

$$F_{r=1}=Z_0Z_1+x_1Z_0+x_0=0 \quad ...(34)$$

In (34) there was a term consisting of one variable, so assume $x_0 = 0$ and $Z_0 = 1$, and we obtain the $Z_1 + x_1 = 0$, i. e. there was a contradiction. Next, check whether all the variables at zero were used to clear the clause $x_0x_1Z_2$. In this case, we are left with no proven only one variable x_1, therefore believe $x_1 = 0$, $Z_1 = 1$, the equation (25) becomes

$$F_{r=1}=Z_3Z_0x_2+x_0Z_3x_2+Z_0x_2+x_3x_0Z_2+x_0Z_2+x_3x_0x_2+x_3Z_2Z_0+Z_2Z_3=0 \quad ...(35)$$

After absorption by summand x_0Z_2 of summand $Z_3Z_0x_2$ and $x_3x_0Z_2$ we obtain

$$F_{r=1}=x_0Z_3x_2+Z_0x_2+x_0Z_2+x_3x_0x_2+x_3Z_2Z_0+Z_2Z_3=0 \quad ...(36)$$

The function does not vanish, and thus no contradiction, therefore, determines the frequency of appearance of the variables in each of clauses (36) cm. Table 7.

Among the clauses in the two variables (36) Select clause to the maximum aggregate rate, this term is a term with a weight x_0Z_2 characteristic of 6, select it with the variable x_0 frequency of occurrence in other clauses of (36) equal to 3 and suppose $x_0 = 0$ and $Z_0 = 1$, and (36) takes the form

$$Z_3x_2+x_2+x_3Z_2+Z_2Z_3=0 \quad ...(37)$$

In (37) x_2 absorb Z_3x_2 and we obtain

$$x_2+x_3Z_2+Z_2Z_3=0 \quad ...(38)$$

In (38) there was a term with one variable x_2, so expect $x_2 = 0$ and $Z_2 = 1$ in this case we obtain $x_3 + Z_3 = 0$ i.e., a contradiction, so we return to the *r*-1 level i.e. go to the function defined by (36) and proceed to attempt to reset the clause x_0Z_2 based variable Z_2, for this purpose (36) believe $Z_2 = 0$, $x_2 = 1$, we get $Z_3Z_0 + x_0Z_3 + Z_0 + x_3x_0 = 0$, having absorption get $x_0Z_3 + Z_0 + x_3x_0 = 0$, where there was a term with one variable Z_0, therefore believe $Z_0 = 0$, $x_0 = 1$ and obtain $Z_3 + x_3 = 0$ i.e., formed a contradiction therefore return to the $_r$-1 level in the case of zero and check instill variables were used to clear the clause $x_0x_1Z_2$ in this case it was the last variable, and hence the procedure terminates since assuming zero variables in clauses $x_0x_1Z_2$ pay equation (25) into an identity and therefore can not function (24) is not feasible.

It should be noted that these estimates of time complexity of the procedure ***A***, for the worst case, are pretty rough estimate from above because when it is obtained it was assumed that when assigning $X_i = 0$ and Z_i =1 at each step only one summand is reset to zero, but in fact h_i^x summands are reset to zero and therefore it is of interest to obtain an estimate of the average work of the procedure ***A***. In the experiment, 2*n* variables X_i and Z_i posted by disjunctions according to uniform distribution law. Results of experimental studies were obtained with a confidence probability of 0.95, while for the average value of the number of elementary operations performed by procedure A at each point from 50 to 70 Boolean functions of studied dimension were generated. Upon experiment the number of disjunctions *m* ranged from 100 to 10,000, and the number of variables *n* changed from 90 to 2400, experimental results are shown in tables (8-12). Period of work of procedure ***À*** is given in milliseconds.

Tables (8-12) show that the largest number of operations is made by the procedure at ratios $a = \frac{n}{m}$ close to 0.24, while for a uniform law of distribution of variables X_i and $\overline{X}_i$ by disjunctions at <0.24 Boolean functions are generally not satisfiable, and at> 0.24 Boolean functions become satisfiable. Sign (+) in tables means that functions are satisfiable, and sign (-) indicates non-satisfiability of Boolean functions. When á is close to 0.24 probability of occurrence of satisfiable and unsatisfiable Boolean functions becomes the same and thus the number of operations fulfilled by procedure ***A*** is increased. Thus, in case where Boolean function contained number of disjunctions *m* = 800 and number of variables *n* = 200, i.e. á = 0.25, the average number of solution was equal to 1,96 10^{11}, at the upper value equal to 1,6'"10^{14}, and the average time of solution was 4.9 hours. With increasing of *k*, see. Table 10,

where the summand $\frac{k+\log_2 m}{2n}$ in (17) is

significantly less than unity, there is a decrease of time complexity of the procedure's work, due to the increased number of zeroed summands at each step of the procedure's work. However, upon further

increase of the summand $\frac{k+\log_2 m}{2n}$,

when it becomes significantly greater than unity, there is a further increase in the number of elementary operations fulfilled by procedure ***A***. In order to test the developed algorithm we used 900 test of specialized library SAT Live [12] containing 403 disjunctions and 100 variables with the value of parameter á = 0.248 generated by pseudo-random filling of disjunctions. Test results are shown in Table 13

Algorithm testing was carried out on a computer ASER with Intel Pentium processor T4400, 2.2Ghz, 3GB Memory.

Summary

Since the issue of existence of an exact polynomial algorithm for solving *SAT* problem is closely connected with the issue of the relationship of classes *P* and *NP*, let's consider it in more detail.

Today, seven mathematical problems included in the list of millennium tasks are known, one of them is the problem of the relationship of classes *P* and *NP*. The issue of the relationship of classes *P* and *NP* is now considered one of the main open issues of modern mathematics and theoretical cybernetics. The founders of this problem are Stephen Arthur Cook, Professor from University of Toronto and a Turing Award winner, and Professor Leonid Levin. Cook's works introduced the concept of *NP*-complete problems and proved that the problem of "satisfiability" also known as the *SAT*-problem is universal *NP*-complete problem. Further development of the theory of *NP*-complete problems was made by a professor at Harvard University, Richard Manning Karp. In recent times, there have appeared papers of Indian mathematician, Vinet Deolalikar, and article "On the Relationship between Classes *P* and *NP*" written by the Ukrainian professor Anatoly D. Plotnikov in Journal of Computer Science 8 (7): 1036-1040, 2012. These papers prove the divergence of these classes. And the paper of Russian professor A. V. Panyukov proves that *P = NP*. Let's show the incorrectness of these efforts of evidence on the basis of results already obtained by the authors in [1, 2, 3] and results obtained in papers of Lavrov and Zykov [7, 8, 9], as papers [1, 2] show that *SAT*-problem is not universal.

It should be noted that the class of *NP*-complete problems is based on the concept of universal problem. All problems belonging to the class of *NP*-complete should be universal and polynomially reducible to each other. Therefore, if polynomial algorithm is received for solving some problem I belonging to the class of *NP*-complete, then in accordance with the Cook's theorem [4] there should be polynomial solvability of all problems belonging to the class of *NP*-complete. In [4] Cook argues that the problem "satisfiability" is *NP*-complete. Once one *NP*-complete problem has become known, the process of proof of *NP*-completeness of the problem *A* is simplified. In order to prove the *NP*-completeness of the problem *À*Î*NP* it is enough to show that any of known *NP*-complete problems *À*′ can be reduced to A, since the property of polynomial reducibility is transitive, i.e. if the problem *A* is converted to the problem *B* during the polynomial time, and if *B* is converted to *C* during the polynomial time, then *A* will be converted to *C* during the polynomial time. First this scheme was used to prove *NP*-completeness of six main problems: "three-dimensional combination"; "partition"; "vertex cover", "Hamiltonian cycle; "clique". Since these were the first problems introduced in the class of *NP*-complete problems after "satisfiability" problem, the proof of their *NP*-completeness was reduced to introduction of rule *Ï*, based on which using some arbitrary problem of "satisfiability" $y \in Y$ there was built structure *S*, with í property, if and only if ã possesses the value truly. For example, for the problem of "vertex cover" graph *G* was as the structure *S*, and í property laid in the fact that graph *G* has a vertex cover with the number of elements not more than *K*, if and only if the following set of disjunctions $\tilde{N} = \{\tilde{n}_1, \tilde{n}_2, \ldots, c_m\}$, defining an arbitrary individual "3-satisfiability" problem, is satisfiable. In general, the problem of "satisfiability"

is a set of *Y* individual problems defined in various ways of defining logic function. It should be noted that when proving *NP*-completeness of all six listed problems first there was selected an arbitrary individual task $ó \in Y$ and it is used for construction during a polynominal time of graph *G*, having a structure interesting for us with necessary í property, only if the logic function corresponding to a given individual problem, possesses the value truly. Since graph G can be arbitrary, we solve some individual problem $z \in Z$, where Z-set of problems generated by using various types of graphs G.

Upon solving any *NP*-complete problem of graph theory there is an inverse problem: an arbitrary graph *G* is given, and it is required to set whether this graph *G* has structure with *v* property or not.

The following issues arise: what individual task (*y*) from a set of individual problems "satisfiability" *Y* corresponds to the problem $z \in Z$ generated by graph *G*, and whether there is inverse for all problems Z or not, and, if so, how it can be constructed according to the original graph and whether such construction is a polynomial or not?

It should be noted that Cook's evidence of universality of problem "satisfiability" was present a priori and it was not clear whether there is at least one *NP*-complete problem or not. However, [2, 3, 4] show that "satisfiability" problem cannot claim to be a universal problem, i.e. *NP*-complete problem. As follows from [2, 3, 4] set of objects that are described by unsatisfiable Boolean functions in an exponential number of times are greater than the number of objects that are described by satisfiable Boolean functions, and properties of polynomial reducibility are by default extended to objects described by unsatisfiable Boolean functions. It was shown in [2, 3] in the vertex cover in graphs. According to [2, 3] it is possible to construct an exponential set of graphs the set of disjunctions , defining an arbitrary individual problem, of which is unsatisfiable on any set of variables, and hence reduction of the problem of vertex cover to the *SAT*-problem for these graphs is impossible. It is easy to show that this fact happens for the problem of Hamiltonian cycle in graphs, as well as any other problem in the theory of graphs included in the list of *NP*-complete problems.

Initially, in the theory of *NP*-complete problems, there was laid an error associated with the fact that upon proving of *NP*-completeness the concept of individual problem does not take into account the topology of studied object. Description of any object is defined by some basic subset of elements describing this object and ratio set on this subset and determining topology of this object, since various ratios can exist for the object of the same dimension, these relationships set different topologies of this object. Let's start with the concept of problem. The mass problem means a common issue that should be answered. The problem is defined by the following information:
- General list of all of its parameters;
- Formulation of those properties to be satisfied by the solution of the problem.

Individual problem is obtained from the mass one, if specific values are assigned to all parameters of the mass problem, but should be more specific, and upon proving of *NP*-completeness this fact is simply ignored. For example, upon proving *NP*-completeness of the problem of Hamiltonian cycle Karp [7] finds a very exotic graph structure for which *SAT*-problem is satisfiable, if and only if it has a Hamiltonian cycle. Then, there is an issue whether this property will be performed for all structures, and if such Boolean function cannot be constructed for some topologies, on what basis we should include these problems in the class of *NP*-complete problems. If we take two graphs of the same dimension, but different in topology, for example one of them is perfect and the other is not, then it is clear that the set of problems that we can solve for the perfect graph does not coincide with the set of problems solved on an imperfect graph. But from the standpoint of Karp evidence it does not matter.

Thus, Cook theorem is valid only for objects described by satisfiable Boolean functions. It should be borne in mind that the number of such objects is negligible compared with objects described by unsatisfiable Boolean functions [2, 3]. So we found that *SAT*-problem is not universal,

and limits to applicability of Cook theorem are very limited. Now let us show that the very proof of NP-completeness of the problem requires clarification. For this let's consider the issue of the proof of *NP*-completeness of arbitrary problem. As shown in papers [2–7], polynomial reducibility of recognition problem I_1 to the recognition problem I_2 means availability of function *f*, which is based on some rules $\ddot{I}_i$ represents a subset of problems D_{i1} in a subset of problems D_{i2} (D_{i1}® D_{i2}), and thus satisfy fulfillment of two conditions:

1. f is calculated by polynomial algorithm;

2. For all $I\varepsilon\ D_i\ I\varepsilon Y_{\tilde{a}1}$,when and only when $f(I)\ \varepsilon\ Y_{\delta 2}$,

where Y_δ is set of problems with answer "yes", while $Y_\delta\varepsilon\ D_i$.

Let's consider three subsets of problems $\{I_j\}$; $\{Z_j\}$; $\{C_j\}$. Let problem *I-NP* be complete and represent a universal problem, and problems *Z* and $\tilde{N}$ are also *NP*-complete, then in accordance with the manner how class of *NP*-complete problems is introduced, they should be polynomially reduced one to another and, thus, if a polynomial algorithm for one of them is found, there should exist polynomial algorithms for all individual problems $\{I_j\}$; $\{Z_j\}$;$\{C_j\}$. As any of *NP*-complete problems can be a universal problem, all the following information should be valid:

$\{I_j\}\rightarrow\{Z_j\}\rightarrow\{C_j\}$; ...(39)

$\{I_j\}\rightarrow\{C_j\}\rightarrow\{Z_j\}$; ...(40)

$\{C_j\}\rightarrow\{I_j\}\rightarrow\{Z_j\}$; ...(41)

$\{C_j\}\rightarrow\{I_j\}\rightarrow\{Z_j\}$; ...(42)

$\{Z_j\}\rightarrow\{C_j\}\rightarrow\{I_j\}$; ...(43)

$\{Z_j\}\rightarrow\{I_j\}\rightarrow\{C_j\}$. ...(44)

Thus, there are rules Λ_{iz} and Λ_{zc}, allowing to reduce problems $I_p\rightarrow Z_p$ and thus, $\{I_\delta\}\Lambda\ Y_{\delta i}$ and problems $Z_p\rightarrow C_p$ and thus, $\{Z_\delta\}\ \Lambda\ Y_{\delta z}$, i.e. rules of conversion of Λ_{iz} and Λ_{zc} satisfy conditions of polynomial reducibility 1 and 2. Let's consider the case when *S* structures are such that generate a set of individual problems {Z}, which by power is greater than the set of individual problems {I}. If the subset {I} contains *n* individual problems, and sets {*Z*} and {*C*} contain *n+k* individual problems each then, for some subset of problems $\{Z_{n+1}, Z_{n+2},\ldots,Z_k\}$ we could not assign any problem from $\{I_j\}$. Consequently, reduction (39) and (40) is possible for all problems and reductions (41), (42), (43) and (44) are possible not for all problems, they are impossible for problems $\{\tilde{N}_{n+1}, \tilde{N}_{n+2},\ldots,\tilde{N}_k\}$ and $\{Z_{n+1}, Z_{n+2},\ldots,Z_k\}$, and, therefore, in this case, the statement that all *NP*-complete problems are polynomially reduced to each other is not satisfied. Thus, the concept of *NP*-complete problem needs clarification. In order to make *NP*-complete problem universal and reducible in all directions inside the class, it is necessary to have a perfect correspondence between all individual problems $\{I_j\};\{Z_j\};\{C_j\}$, i.e. for any pair of individual problems there should be direct and indirect polynomial reduction determined by conditions 1 and 2.

So, if we have subsets of problems $\{I_j\}$; $\{Z_j\}$; $\{C_j\}$, and power of set of individual problems $\{I_j\}$ is different from power of set of problems $\{Z_j\}$ and $\{C_j\}$, then in order to prove that certain problem *I* is *NP*-complete, it is not enough to show that any individual problem $\{I_j\}$ is polynomially reduced to set of problems $\{Z_j\}$ and $\{C_j\}$, i.e. conditions 1 and 2 are fulfilled, as upon proving *NP*-completeness of "satisfiability" problem in papers of Cook and Levin, but in this case it is necessary to show that there are problems $\{z_{n+1}, z_{n+2},\ldots, z_k\}$, polynomially reduced to problems $\{\tilde{N}_{n+1}, \tilde{N}_{n+2},\ldots,\tilde{N}_k\}$ and $\{Z_{n+1}, Z_{n+2},\ldots,Z_k\}$, and "checkability" of these problems of recognition should be polynomial.

Papers [2, 3, 4] show that all problems related to class *NP*-complete can be divided into subsets of problems , inside of which polynomial reducibility is possible and hypothesized that between subsets apparently there only possible polynomial reduction of certain individual problems. Therefore, here we can talk only about polynomial solvability of individual problems, which can be reduced to a problem *I*. The only thing that unites problems which we attribute to *NP*-complete is the fact that no algorithm for solving polynomial complexity is known for them. It is therefore proposed to use for these tasks the term hardest problems instead of the term *NP*-complete problems. In paper [2] it is hypothesized that the issue of the existence of the universal problem is algorithmically unsolvable problem. As follows from [8, 9, 10], this hypothesis is supported by the fact that now the list of *NP*-complete problems includes more than three thousand tasks, and thus it includes almost all major problems of the theory

of graphs, then proceeding of the declared Cook polynomial reducibility of problems within that class, for solving all problems of graph theory included in this list there should exist an algorithm to solve them with some arbitrarily high complexity, which leads to their polynomial reducible to each other, but this is contrary to results obtained in papers of I. A. Lavrov (1963) [8] which shows the impossibility of construction of such algorithm and to what A. A. Zykov focused attention in papers [10].

CONCLUSIONS

1. The problem of relationship of classes *P* and *NP* posed by Cook and ranked in the list of millennium tasks is just ill-posed mathematical problem; therefore, it is not surprising that nobody could solve it. This problem should be excluded from the list of millennium tasks, as to this date scientists are spending their precious time to solve it, it is evidenced by papers of Indian mathematician Vinet Deolalikar and article of Ukrainian Professor A. D. Plotnikov and Russian Professor A. V. Panyukov. The presence of this problem in the list of millennium tasks inhibits further development of mathematics. It should also be noted that the theory of *NP*-completeness cannot be used to study properties of optimization problems [11], and thus all results obtained in the theory of algorithms, based on the "total" reducibility within the class of *NP*-complete problems, declared by Cook, require serious revision.

2. The paper shows that *SAT* problem belongs to the class *P*, but that does not mean equality of classes *P* and *NP*, but only determines the possibility of solving during polynomial time some subset of problems that are polynomially reducible to *SAT*-problem and allows to create fast *SAT*-solvers able to solve many problems of discrete optimization of large dimension in real time, which previously were considered unsolvable.

REFERENCES

1. S. V. Listrovoy, Minukhin S. V. Investigation of the Scheduler for Heterogeneous Distributed Computing Systems based on Minimal Cover Method // *International Journal of Computer Applications* (0975 – 8887) **51**(19): : 35–44 (2012).
2. Listrovoy S. V. On Correlation of Ð And NP Classes // I. *J. Modern Education and Computer Science,* **3**: 21-27 (2012).
3. Listrovoy S. V. On NP classes and NP-complete problems. // *Electronic simulation,* **33**(1): p. 31–45 (2011).
4. Listrovoy S. V. "On polynomial reducibility in NP class", Ukrainian Mathematical Congress ", Algebra and Number Theory. http://www.imath.kiev.ua/ (2009)
5. Gary Ì., Johnson D. Computing machines and hard problems. – Ì: *Mir*, 1982. – 336 p.
6. Cook S. À. Complexity of procedures of a conclusion of theorems. - Cybernetic collection of a new series, vol.12. - Moscow.: Mir, -Ð.5 - 15 (1975).
7. Karp R. Ì. Reducibility of combinational problems. - Cybernetic collection of a new series, vol. 12. - Moscow: Mir, -Ð.16-38 (1975).
8. I. A. Lavrov. Effective inseparability of a set of identically true and a set of finitely refutable formulae of some elementary theories. Algebra and logics (Materials of the workshop of Institute of Computational Mathematics and Mathematical Geophysics of Siberian Branch of the USSR Academy of Sciences) 2 (1963), No.1, 5-18. RzhMath, 1964, 1À112.
9. Papadimitriou, K. Steiglitz Combinatorial optimization Algorithms and complexity– M.: Mir, 1985.-509p.
10. A. A. Zykov. Elements of graph theory // Moscow: Nauka, p. 381 (1987).
11. S. V. Listrovoy. On The Theory Of Np-complete Problems // *International Journal of Computers & Technology,* **11**(4): Ð. 2481–2483 (2013).
12. SAT Live [Digital resource]. – Access mode: www.satlive.org, free.