The American Journal of Engineering and Technology
(ISSN – 2689-0984)
Published: February 24, 2021 | **Pages:** 56-66
Doi: https://doi.org/10.37547/tajet/Volume03Issue02-09

IMPACT FACTOR
2021: 5. 705
OCLC - 1121105677

Pseudo Polynomial Reduction Of Problems And Strong Np-Hard Problems

Tatyana Matveevna Kosovskaya
Doctor Of Science, Professor, St. Petersburg State University, St. Petersburg, Russia

Elmurod Solijon Ugli Kodirov
Postgraduate Student, St. Petersburg State University, St. Petersburg, Russia

ABSTRACT

This scientific article is devoted to the study of complexity theory for solving artificial intelligence (AI) problems and the classification of problems according to NP completeness. is similar to NP-complexity generation methods for boundary programming problems. It is wiser and more efficient to look for sufficiently accurate prediction algorithms than to find evidence that the problem in question belongs to the NP class and therefore wastes time. This scientific article provides a typical example that illustrates all algorithms.

KEYWORDS

NP-complexity, NP hardness, polynomial time, NP completion problem, NP-kit

INTRODUCTION

Most of the practically interesting problems have polynomial (running in polynomial time) algorithms for their solution. That is, the running time of the algorithm at an input of length n does not exceed O (NK) for some constant k (regardless of the input length). Of course, not every problem has a solution algorithm that matches this feature. Some

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
Published: February 24, 2021 | **Pages:** 56-66
Doi: https://doi.org/10.37547/tajet/Volume03Issue02-09

IMPACT FACTOR
2021: 5. 705
OCLC - 1121105677

problems are not solved by any algorithm. A classic example of such a problem is the "stop problem" (to see if a particular program stops at a particular record). There are also problems with the algorithm that solves them, but such an algorithm works "for a long time" - its time is not equal to O (NK) for any constant k.

A class of algorithms that run in polynomial time is a reasonable first approximation if we want to draw a rough but formal line of interest between "practical" algorithms and algorithms that are only of theoretical interest. [1] We consider a class of problems called NP-complete. No polynomial algorithms have been found for these problems; however, the existence of such algorithms has not been proven. Examining All NP Problems deals with a question called "P = NP". This question arose in 1971 and is now one of the most difficult problems in computation theory. Why would a programmer know all NP problems? If one can prove that his NP is complete for a problem, there is reason to believe that it is almost insoluble. In this case, it is better to spend time creating a rough algorithm than to keep looking for a fast algorithm that clearly solves it.

1. Problem that is NP-hard in the strong sense

Examples of proving NP-complexity in the strong sense of a set of programming problems are given. The technique of proving such results is in many ways similar to the methods of generating NP-complexity (in general terms) for boundary programming problems. To prove NP stiffness or other extreme problem, we prove the stiffness of the corresponding recognition problem. To do this, it suffices to show a reduction in the problem that fills some NPs that are well known in a strong sense. In this chapter, the 3-part problem is selected as follows. There are many $N^U = \{1,2,\ldots.,3n_0\}$ each element of which is assigned a natural number (weight) e, and $\sum_{i\in N^0} e_i =$ n_0E and E/4 <e_i <E/2, $i \in N^0$. Is there such a partition of courage. V_0 into three-element subsets N °, N * ..... N ° such that $\sum_{i\in N_j^0} e_i = n_0$ for all ; j, j = l, n_0 ? Note that the condition $E / 4 < e_i < E / 2, i \in N^0$ is essential. It means that the sum of the weights of any four (or more) elements will be greater than E. and any two elements - less than E. We also emphasize that the partition problem is historically the first problem for which W-completeness in the strong sense.

To provide a pseudo polynomial reduction of a 3-partition problem to some recognition problem, it is necessary to transform the input information of the 3-partition problem into the input information of some of its individual recognition problems. The number of actions performed in this case should polynomial depend on the length of the input to the 3-partition problem represented in the unary number system, in other words, on the value$O(n_0E)$ Sometimes it is possible to establish the usual polynomial reducibility. In this case, the number of actions on the transformation of the input information polynomial depends on n_0log_2E or even from no. The individual recognition problem constructed as a result of transforming the input of the 3-partitions problem must have an answer if and only if the 3-partitions problem has the same answer.

1.1. Serving requests without delay

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
Published: February 24, 2021 | **Pages:** 56-66
Doi: https://doi.org/10.37547/tajet/Volume03Issue02-09

IMPACT FACTOR
2021: 5. 705
OCLC - 1121105677

Establish NP hardness in the strong sense FS|M|t_{iL} >= o|NW||$t_{\max}(s)$ In this problem, the condition means that job I is not serviced by the server L. For this interpretation of zero durations, the optimal schedule without delays does not necessarily belong to the class of permuted schedules. In other words, tasks FS|M|t_{iL} >= o|NW||$t_{\max}(s)$ and FS|M|t_{iL} >= o|NW, in this case are not equivalent. Recall that the tasks FS|M|t_{iL}|NW||$t_{\max}(s)$ and FS|M|t_{iL}|NW, Π||$t_{\max}(s)$ are equivalent if all either among f.t are equal to zero and the record means that the duration.

Theorem 1.1. A task FS|2|t_{iL} >= o|NW||$t_{\max}(s)$ is NP hard in the strong sense if the condition t_{iL} = o means that customer i is not serviced by server L. Evidence. Let us formulate the corresponding recognition problem: determine whether there exists a schedule s ° for which $t_{max}(s^0)$ =< y for a given number of y. Let us show that the 3-partition problem reduces to this problem. To transform the input information of the 3-way problem into the input information of the individual recognition problem, the values n, M, t_{iL} and u express through n_0, e_i and E. We put M=2, n= 4 n_0 + 2 and divide the set N into two groups: and-requirements, denotedu_{ij} $i = 1,3$ n_0 and v-requirement, denoted $v_i, j = 0, n_0 + 1$. We put $t_{ai^1} = 0$, $t_{ai^2} = e_i, i = 1, 3n_0$; $t_{0^1} = 0, t_{0^2} = 2E$; and set y=(2$n_0$ + 3)E. Let us show that in the constructed problem the schedule S ° for which$t_{max}(s^0)$ <= y, exists if and only if the 3-partition problem has a solution. has a solution to the problem of 3 partitions.

1. Let the 3-partition problem have a solution and N_j^0, j=1, n_0 found three-element subsets of elements of the set No. Across $\Pi_j(u)$ denote an arbitrary permutation of u-requirements with numbers from the set N_j^0, j=1, n_0. Device 1 operates without downtime in the time interval [o; 2(n_0+1)E], [o; y]. In this case, device 1 in the interval 2(j-1)E; 2jE service requirement $v_i, j = 1, n_0$+1 Server 2 in the interval serves the requirement vo, and in each of the intervals (2jE; 2(j+1)E) sequence of requirements$V_{j_1}\Pi_j$(u), Demand $V_{n_0} + 1$ served by device 2 in the interval (2($n_0 + 1$)E; (2$n_0 + 3$)E).

2. Let there be a schedule s °. As $t_{max}(s^0) = y$ and device 2 operates in the interval [o; y] without downtime.

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
Published: February 24, 2021 | **Pages:** 56-66
Doi: https://doi.org/10.37547/tajet/Volume03Issue02-09

IMPACT FACTOR
2021: 5. 705
OCLC - 1121105677

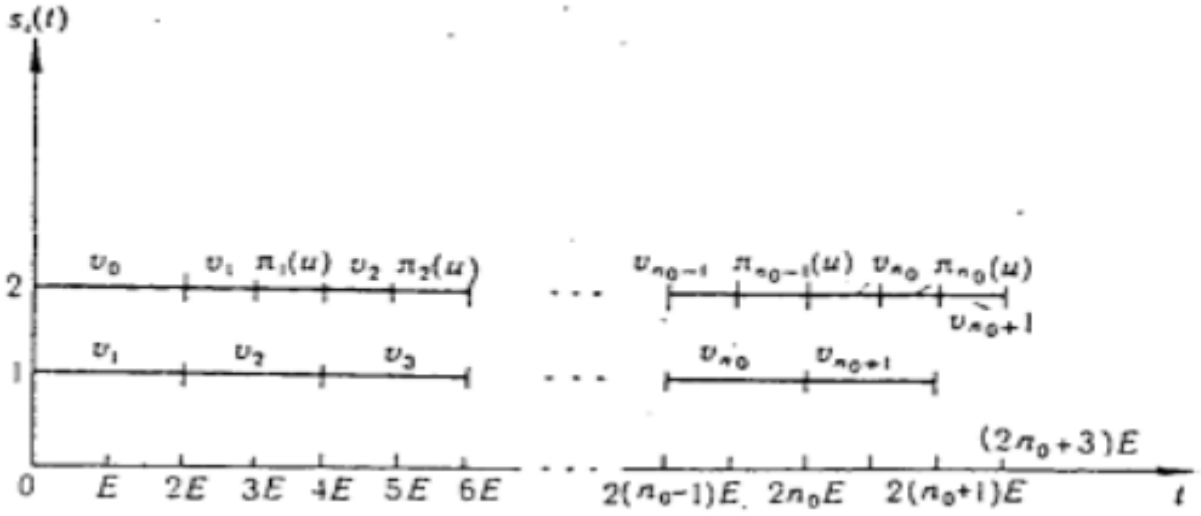


Fig. (1.1. Subsets of elements of the set)

Device 1 completes servicing all requests not earlier than the moment of time y - E. Taking into account that delays are prohibited, the last request that will serve device 2 will be some v-requirement, and device 1 must operate without downtime in the interval [0; y=E]. Since the requirements $v_j, i \neq 0$, do not differ from each other in the duration of service, without breaking the generality, we can assume that they are served by the device I in ascending order of their numbers. This immediately implies that the requirements $v_j, i \neq 0$, are served with the schedule s ° in the same way as with the schedule shown in Fig. 1.1. The v_e requirement is served by device 2 in the interval [0; 2E] , since this is the only unoccupied subinterval of length 2E interval [0; y]. Requirements u should be serviced by device 2 at intervals ((2j +1)E; 2(j +1)E), j=1, n_0 Denoting through N_j^0 set of numbers and-requirements served by device 2 in the interval ((2j +1)E; 2(j +1)E), j=1, $n_0$we obtain a solution to the 3-partition problem. Thus, for O(n_0) of actions, the 3-beat-beating problem is reduced to the problem of the existence of a schedule S °. The theorem is proved.

Theorem 2. Problem FS|3|t_{iL}|Pr ||$t_{max}(s)$ is NP-hard in the strong sense.

Theorem 2. Problem

JS|2|$t_{ig} = 1|\vec{G} = C_1 \cup C_2 \cup \ldots \cup$
CP| tmax(s) is NP-hard in the strong sense.

Evidence. Let us formulate the corresponding recognition problem. The servicing system, consisting of two servers A and B, at time d = 0 receives a partially ordered set of requests.

N {1,2 ...., n}….. Each connected component of the reduction graph of the relation -, given on N, is a chain. The duration of each operation is equal to one. It is required to determine whether there is a schedule S ° for which $t_{max}(s^0)$ <= y for a given value y.

Let us construct a pseudo-complete reduction of the 3-partition problem to the formulated recognition problem.

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
Published: February 24, 2021 | **Pages:** 56-66
Doi: https://doi.org/10.37547/tajet/Volume03Issue02-09

IMPACT FACTOR
2021: 5. 705
OCLC - 1121105677

Let's put:$n = 4n_0E$, y=$2n_0E$, N=$\bigcup_{k=1}^{3n_0+1} N_k$, where $N_1 = \{1,2, \ldots..2e_1\}$,

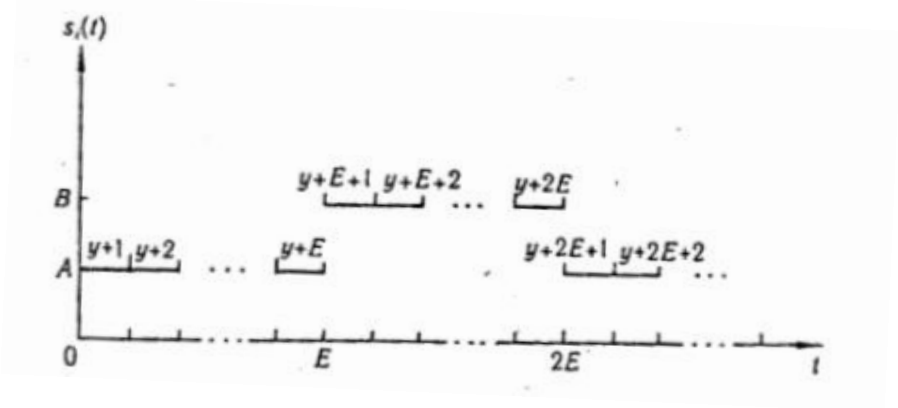
$N_k = \{2\sum_{i=1}^{k-1} e_i + 1,\ 2\sum_{i=1}^{k-1} e_i + 2, \ldots, 2\sum_{i=1}^{k} e_i\}$, k=2,3……,$3n_0$,

$N_{3n_0+1} = \{2n_0E + 1, 2n_0E + 2, \ldots, 4n_0E\}$. On the set N we establish a relation -, put / - I if and only if $i \in N_k, j \in N_k, 1 \le k \le 3n_0 + 1, and\ i < j$. Obviously, each connected component of the reduction graph of this relation is a chain. Let the process of servicing each customer $i \in N$ it remains to perform a single operation. For requirements from the set N1, we put L1 = (B) for l=1, $e_1L^l = (A)$. For each k, $2 \le k \le 3n_0$, put

$$L^{2\sum_{i=1}^{k-1} e_i+1} = (\boldsymbol{B}), if\ l = 1, 2, \ldots., e_k;$$

$$L^{2\sum_{i=1}^{k-1} e_i+1} = (\boldsymbol{A}), if\ l = e_k + 1, e_k + 2, \ldots., 2e_k;$$

For a requirement from the set N_{3n_0+1} - linearly ordered and the number of its elements is $2n_0E = y$, then for any schedules with a total customer servicing time equal to y, the customers of the set N3n +1 must be serviced continuously one after the other starting from the time instant d = 0. In Fig. 2 shows a fragment of the service schedule for the requirements of the set N_{3n_0+1} at which the total time of servicing the customers is equal to This schedule is periodic with a period



Pic(1.2. Service schedule of requirements)

Thus, servicing the requirements of the set N\$N_{3n_0+1}$ for any schedule with total customer servicing time equal to y, it can be performed only in those unit intervals that are determined by the sequence $([[B]^E[A]^E]^{n_0})$. Let us show that the schedule So can be constructed if and only if the 3-change problem has a solution. Let there be a partition of the set # into no three-element subsets $N_1^0, N_2^0, \ldots., N_{n_0}^0$ such that $\sum_{i \in N_j^0} e_i = E$ at j = 1.0. Then, serving many requirements

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
Published: February 24, 2021 | **Pages:** 56-66
Doi: https://doi.org/10.37547/tajet/Volume03Issue02-09

IMPACT FACTOR
2021: 5. 705
OCLC - 1121105677

$\bigcup_{k\in N_j^0} N_k$ interval in accordance with the relation, we obtain a schedule with the total service time of customers equal to y. Let there be a schedule S ° for which $t_{max}(s^0)$. Since the total duration of servicing all requests by the device A (by the device B) is equal to y, then in the schedule S, both the device A and the device B in the interval [o; y] is not idle. By condition, service of each request $k \in N^0 = \{1,2,\dots,3n_0\}$, for which $L^i = (B)$.

Taking into account that in the example constructed in the proof of Theorem 3, the process of servicing each customer $i \in N$ consists of a single operation, it can be concluded that the task JS|2|$t_{ig} = 1|\vec{G} = C_1 \cup C_2 \cup \dots \cup C_p||\vec{t}_{max}(S)$. is NP-hard in the strong sense.

2. Algorithm

Stage 1. Construct a contour less graph (Q, u') $\in P(G)$ as a result of performing the next step no more than q - 1 times. Initially put G^0=G.

Let after execution $l \geq 0$ steps obtained a mixed graph $G^{(1)} = (Q, u^{(1)}, V^{(1)})$, $V^{(1)} \neq \emptyset$ for which l vertices are marked and there is an unmarked vertex i, which is not entered by a single arc outgoing from the unmarked vertex. (If there is no such vertex, then the graph (Q, U) contains a contour. End.)

Step 1. Mark vertex i, and if the graph G (1) contains edges incident to this vertex, then replace them with arcs outgoing from it. The resulting graph is denoted by $G^{(y+1)}$ = (Q, $U^{(l+1),}$ $V^{(l+1),}$). If $V^{(l+1)} = \emptyset$, then the uncontrolled graph is constructed: (Q, U') = $G^{(l+1)}$ Otherwise, perform this step, replacing l with l + 1.

Stage 2. In accordance with Algorithm 3.1, it is possible to construct which relative to the directed graph Q, $U' \in P(G)$ and mixed graph G. End. It is easy to verify that the described algorithm is correct, which completes the proof of the theorem.

Obviously, the complexity of stage 1 of the algorithm does not exceed O(|U|+|V|+q) and, therefore, the total complexity of constructing an active schedule admissible with respect to the mixed graph G does not exceed O(q^2) .

If for each graph G^1 consider all possible options for choosing the vertex i in the algorithm, then we obtain the entire set P (G) = {G1 G2, ..., G} of uncontrolled graphs generated by the mixed graph G. Therefore, the entire set of active schedules admissible with respect to G can be constructed in $O^{(s^2\lambda)}$ elementary actions.

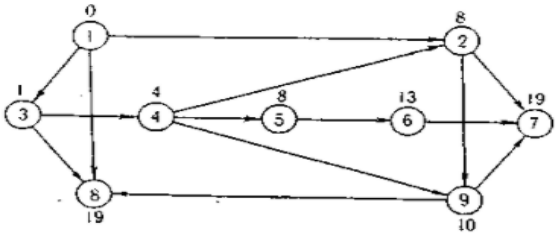
Example 1. Under the conditions of Example 3.1 (see Fig. 1), we will use Algorithm 2 to construct a schedule admissible with respect to C = (Q, V, V).

At stage 1 of the algorithm, we construct an uncontrolled directed graph from the set P (G). We put G^0 = G and select, at step 1, vertex 1, into which no arc enters. We mark vertex 1 and replace edges incident to it with arcs. We get a mixed graph $G^{(1)} = (Q, U \cup \{(1,3),(1,8)\}, V\backslash\{[1,3],[1,8]\})$.

At step 2, we perform similar transformations for vertex 3, at step 3 - for vertex 4, at step 4 - for vertex 2, and at step 5 - for vertex 9. The resulting graph $G^{(5)} = (Q^{(5)}, U^{(5)}, V^{(5)}), V^{(5)} = \emptyset$ presented in Fig. 1 as a network diagram.

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
Published: February 24, 2021 | **Pages:** 56-66
Doi: https://doi.org/10.37547/tajet/Volume03Issue02-09

IMPACT FACTOR
2021: 5. 705
OCLC - 1121105677



Pic (2.1. Algorithm construction)

At stage 2, we use Algorithm 1 to construct an active schedule that is feasible with respect to the directed graph $G^{(5)} \in P(G)$ by ranks $R_1 = \{3\}, R_2 = \{4\},$

$R_3 = \{2,5\}$, $R_4 = \{6,9\}$, $R_5 = \{7,8\}$ finally, using formula 1, we calculate the values $t_j^0(Q, U')$ for all peaks j∈ $Q \backslash R_0$ sequentially in the order of non-decreasing ranks of the vertices. Received values $t_j^0(Q, U')$ indicated near the corresponding vertices i∈ Q network graphics.

2.1 Sequential analysis of options

Above, we considered the issues of constructing an active schedule that is admissible with respect to a directed or mixed graph. It has been said that no more than O(q^2) elementary actions, you can build an admissible schedule or establish the impossibility of building such a schedule.

If at each step of Algorithm 2 we consider all possible options for choosing the vertex i, which is not entered by any outgoing from: unmarked vertex of the arc, then we can obtain the entire set P (G = {G1, G2 ... .G} of uncontrolled graphs generated by the graph G = (Q, U, V) The problem can be solved by constructing active schedules admissible with respect to the graphs $G_k, k = l, \lambda$ and choosing among them the schedule S * JS, OS|M|t_{iv}|G||F(s) can be limited by the value O($q^2\lambda$) elementary actions. Unfortunately, the cardinality of the set P (G) grows in the general case exponentially with the growth of the parameters of the problem n and M. For example, for the problem JS, OS|M|t_{iv}|G||F(s) value λ equals n !. Graph G=(Q, U, V) at the same time, it is complete not oriented: $U = \emptyset$ |V| =($\frac{n}{2}$) and q = n. Growth of values | v | and is shown in table.

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
Published: February 24, 2021 | **Pages:** 56-66
Doi: https://doi.org/10.37547/tajet/Volume03Issue02-09

IMPACT FACTOR
2021: 5. 705
OCLC - 1121105677

Table 1.

n	q	$\|V\| = \binom{n}{2}$	$\lambda = n!$
1	1	0	1
2	2	1	2
3	3	3	6
4	4	6	24
5	5	10	120
6	6	15	720
7	7	21	5 040
8	8	28	40 320
9	9	36	362 880
10	10	45	3 628 800
11	11	55	39 916 800
12	12	66	479 001 600

Reducing the number of times the schedule is searched can be achieved by sequential analysis of options. To organize a targeted enumeration of the graphs of the set P (G), we will use the procedure for sequentially dividing the set P (G) into subsets. In this case, the subset P (G) is first partitioned by subsets $P(G^1), P(G^2)$ …… $P(G^k)$, where (G^k), k = 1, h- graphs obtained from G = {Q,U,V} as a result of replacing one or more edges of V with arcs. Then one of the sets P(G^k), for instance P(G^1) in turn breaks down into subsets P(G^{h+1}), P(G^{h+2}),…., P(G^{h+p}) as a result, we obtain a partition of the original set. It is convenient to represent this process as a "growing" emerging tree (Z_m, W_m), where Zm is its vertex set.

The set of all finite vertices of the tree (Z_m, W_m) will be denoted by Z_m. When condition a) is satisfied, we obtain the exact value f (C '). It is easy to see that condition b)

implies the relation f(Q, U_m) ≤ f(Q, U') for any graph (Q, U') ∈ $P(G^l)$. Under condition c), there is a graph (Q, U^n) ∈ $P(G^k)$ such that f (Q, U^n) ≤ $f(G^1)$ Therefore, when searching for the optimal schedule, the graph G^1 can be excluded from consideration if condition b) or c) is satisfied.

3. Pseudo polynomial Reduction of Problems and Strong NP-Hard Problems

Along with the division of problems into NP-hard and polynomial solvable, NP-hard problems, in turn, are subdivided into NP-hard problems in the strong sense and problems with pseudo polynomial algorithms for their solution.

An algorithm for solving problem Π is called pseudo polynomial if its time complexity does not exceed a certain polynomial in two variables Length [I] and Max [I] for any example I of problem Π. The corresponding problem is called pseudopolynomially solvable.

It is easy to see that any polynomial algorithm is also pseudo polynomial. In addition, none of the NP-hard problems that have no numerical

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
Published: February 24, 2021 | **Pages:** 56-66
Doi: https://doi.org/10.37547/tajet/Volume03Issue02-09

IMPACT FACTOR
2021: 5. 705
OCLC - 1121105677

parameters can have a pseudo polynomial algorithm for solving if $P \neq NP$. Otherwise, such a problem would have a polynomial solution algorithm, since Max[I] = o for any example of it.

Similarly, if for problem Π there exists a polynomial such that for p(Length$_{\text{п}}[I]$) $\geq Max_{\text{п}}[I]$ any if it is NP-hard, it cannot be pseudopolynomially solvable.

Thus, there are NP-hard problems that cannot even have pseudo polynomial algorithms for their solution. Such problems form a set of so-called NP-hard problems in the strong sense. To formalize the concept of a NP-hard problem in the strong sense, we introduce the definition of pseudo polynomial reducibility of recognition problems.

Let pairs of functions (Length$_{\Pi}$, Max) and (Lengthu ', Max>) be associated with problems Π and Π', respectively.

The recognition problem Π is said to be pseudo-polynomial reduced to the recognition problem Π 'if there exists a dictionary function Φ that transforms problem Π into problem Π' and satisfies the following conditions:

(a) For any example IЄ $D_{\text{п}}$ ratio I ∈ Y_1 is executed if and only if $\phi(I)$ ∈ $Y_{\text{п}}$;
(b) The time complexity of calculating the function Φ does not exceed a certain polynomial q in two variables Length$_{\Pi}[I]$ and Max$_{\Pi}$ [I];
(c) There exist polynomials qi and qi such that for any I

$$q_1(\text{Length}_{\Pi}[\phi(\text{I})]) \geq \text{Length}_{\Pi}[I]$$

$$q_2\text{Max}_{\Pi}[\text{I}], \text{Length}_{\Pi}[I] \geq \text{Max}_{\Pi}[\phi(\text{I})]$$

A recognition problem Π is called NP-hard in the strong sense if there exists an NP-hard recognition problem Π that reduces pseudopolynomially to Π and for any Max$_{\hat{\Pi}}$[I] $\leq \hat{P}$ ($Length_{\hat{\Pi}}[I]$)

Since any problem is pseudopolynomially reducible to itself, then any NP-hard problem satisfying condition (1.3) is NP-hard in the strong sense. Moreover, any NP-hard problem in the strong sense is NP-hard, and none of the NP-hard problems in the strong sense can have a pseudo polynomial solution algorithm if $P \neq NP$.

The notion of pseudo polynomial reducibility is transitive. Therefore, to prove the NP-hardness in the strong sense of a certain problem, it suffices to pseudopolynomially reduce to it some NP-hard problem.

A recognition problem Π is called NP-complete in the strong sense if $\Pi^{\text{п}} \in NP$ and Π is NP-hard in the strong sense.

An extreme combinatorial problem is called NP-hard in the strong sense if the associated recognition problem is NP-hard in the strong sense.

Examples of NP-hard problems

Below are the formulations of NP-hard problems, which are further used to prove the NP-hardness of optimal planning problems.Partitioning: Given positive integers al, ao, ..., ag and A such, X ∈ N ={1,2,….r}, $\sum_{j\in x} a_j = A$

It is required to determine whether there exists a set X⊂ $N = \{1,2,\dots r\}$ so what $\Sigma_{i=4}^{r} a_j = 2A$

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
Published: February 24, 2021 | **Pages:** 56-66
Doi: https://doi.org/10.37547/tajet/Volume03Issue02-09

IMPACT FACTOR
2021: 5. 705
OCLC - 1121105677

4. Using complex NP tasks.

Public key encryption systems are widespread today, and their characteristic feature is that they do not provide additional advantages when breaking algorithmic data, unlike symmetric systems, which are easily broken if you know how to act according to the basic rule. The most common and most popular example of a public key system today is RSA (the algorithm was published in 1977 at MIT by Rivest, Shamir, Edleman). This system is easy to use and, most importantly, difficult to break. However, the question remains open - is the factorization problem based on the RSA algorithm applicable to complex problems of polynomials of class P or class NP? The P and NP problems are important and related to success theory and, in part, to cryptography theory. A really efficient algorithm for the NP completion problem, the satisfaction algorithm can, on the one hand, create a number of useful algorithms for solving many practical computational problems in this area, but on the other hand, such an algorithm is safe, financial and other operations. It turns out that not today or tomorrow the factorization problem is not NP-hard, and there may even be a polynomial algorithm to solve it. In this case, completely different cryptographic methods will be required. In this article, I will show you some encryption methods based on a well-known NP-hard problem - the bag problem.

CONCLUSION

So what is the practical meaning of studying the theory of complexity and classifying problems in terms of NP-completeness? The answer is clear - it would be wiser and more efficient to look for sufficiently accurate predictive algorithms than to find evidence that the problem in question belongs to the NP-kit class and is therefore a waste of time. polynomial algorithms to solve this problem. It is clear that the central role here is played by the problems filled with NP - the truth is that polynomial time is first, but very close to the concept of "ability to solve practical problems". However, there are many interesting books on this topic that will interest the reader. Among them, I would like to highlight where you can find different topics related to NP in different areas. I think it would be interesting for students who want to know more about the theory of complexity to take a detailed lecture course that covers this topic in more detail.

REFERENCES

1. Gary M., Johnson D. "Computing machines and intractable problems". M .: Mir 1982-466 pp.
2. Ivanova A.P. "Introduction to Applied Programming. Models and Computational Algorithms". M .: Fizmatlit 2002
3. Perepelitsa V.A. "Asymptotic approach and solution of some extremal problems on graphs. Problems of Cybernetics "M .: Nauka, 1973
4. A.V. Yakovlev, A.A. Bezbogov, V.V. Rodin, V. N. Shamkin, CRYPTOGRAPHIC PROTECTION OF INFORMATION
5. Eremeev A.V., Romanova A.A., Servakh V.V., Chaukhan S.S. Approximate solution of one problem of supply management //

Diskretn. analysis and issled. operations. Series 2. 2006. T. 13, No. 1. P. 27–39.

6. Кодиров Э. С. У., Халилов З. Ш. ВЗАИМОСВЯЗИ И РАЗЛИЧИЯ МЕЖДУ “DEEP LEARNING” И “MACHINE LEARNING” //Universum: технические науки. – 2020. – №. 7-1 (76).
7. Кодиров Э. С. У., Халилов З. Ш. ВОЗМОЖНОСТИ И ПРЕИМУЩЕСТВА ИСКУССТВЕННОГО ИНТЕЛЛЕКТА (ИИ) И ЛОГИЧЕСКИХ ВЫЧИСЛЕНИЙ //Universum: технические науки. – 2020. – №. 6-1 (75).
8. Karimov U. et al. USING NEW INFORMATION TECHNOLOGIES IN DISTANCE LEARNING SYSTEM //НОВАЯ ПРОМЫШЛЕННАЯ РЕВОЛЮЦИЯ В ЗЕРКАЛЕ СОВРЕМЕННОЙ НАУКИ. – 2018. – С. 9-11.
9. R.Khamdamov, U.Begimkulov, N.Taylokov. Information technology in education. Tashkent, 2010.