Вестник РУДН. Серия: Инженерные исследования
RUDN Journal of Engineering Research

2022;23(2):108–116

ISSN 2312-8143 (Print); ISSN 2312-8151 (Online)
journals.rudn.ru/engineering-researches

DOI 10.22363/2312-8143-2022-23-2-108-116
UDC 510.52:519.16

Research article / Научная статья

Necessary and sufficient conditions for dividing the structure of algorithms into non-intersecting sets: polynomial and enumeration algorithms

Natalia L. Malinina

Moscow Aviation Institute (National Research University), *Moscow, Russian Federation*
✉ malinina806@gmail.com

Article history
Received: January 28, 2022
Revised: March 11, 2022
Accepted: April 15, 2022

Keywords:
algorithm, alphabet, graph, graph-scheme, block-graph, cyclomatic number, isomorphism

Abstract. The article is devoted to a rigorous proof of the first millennium problem, which is named as $P \neq NP$. This problem was raised in 1971 by S. Cook and marked the beginning of a long struggle in order to understand and prove it. The problem is closely related to the concept of a combinatorial explosion, which concept was aroused in the early 1970s and became a symbol of the enormous difficulties that developers of algorithms and programs have to face, since the complexity of the tasks that have to be solved is growing every day. The presented proof is based on the achievements of graph theory and algorithm theory. Necessary conditions (normalizing), to which arbitrary algorithm must satisfy in order to be solved with a help of a Turing machine, are proved in the article. Further, using the theory of algorithms and graph theory, it is proved that normalized (necessary condition) graphs (visualization of algorithms) with respect to such a characteristic of their complexity as a cyclomatic number fall into three non-intersecting sets that have different properties. These properties are determined by the structural features of graphs, and they can be taken into account when developing algorithms and programs for solving mass problems. The division of algorithms of mass problems into three non-intersecting sets is proved. Such division corresponds with graph-schemes, or block-schemes of polynomial (P) or enumeration (NP) algorithms. This proves a sufficient condition, to which algorithms must satisfy in order to belong to different classes and actually confirm that $P \neq NP$.

For citation
Malinina NL. Necessary and sufficient conditions for dividing the structure of algorithms into non-intersecting sets: polynomial and enumeration algorithms. *RUDN Journal of Engineering Research*. 2022;23(2):108–116. http://doi.org/10.22363/2312-8143-2022-23-2-108-116

Необходимые и достаточные условия разделения структур алгоритмов на непересекающиеся множества: полиномиальные и переборные алгоритмы

Н.Л. Малинина

Московский авиационный институт (национальный исследовательский университет), *Москва, Российская Федерация*
✉ malinina806@gmail.com

История статьи
Поступила в редакцию: 28 января 2022 г.
Доработана: 11 марта 2022 г.
Принята к публикации: 15 апреля 2022 г.

Аннотация. Представлено строгое доказательство первой проблемы миллениума, а именно: $P \neq NP$, которая была озвучена в 1971 г. в статье Стивена Кука и положила начало долгой борьбе за ее осмысление и доказательство. Проблема тесно связана с понятием комбинаторного взрыва,

Ключевые слова:
алгоритм, алфавит, граф, граф-схема, блок-схема, цикломатическое число, изоморфизм

возникшего в начале 1970-х гг. Она стала символом тех громадных трудностей, с которыми приходится сталкиваться разработчикам алгоритмов и программ, поскольку сложность решаемых задач с каждым днем растет. Предлагаемое доказательство основано на достижениях теории графов и теории алгоритмов. Обосновывается необходимое условие того, чтобы произвольный алгоритм мог быть решен с помощью машины Тьюринга и приводятся необходимые теоремы. Далее с помощью теории алгоритмов и теории графов доказывается, что нормализованные графы (визуализации алгоритмов) относительно такой характеристики их сложности, как цикломатическое число, распадаются на три непересекающихся множества, которые обладают различными свойствами. Эти свойства определяются структурными особенностями графов, их можно учесть при разработке алгоритмов и программ для решения массовых задач. Доказывается разделение алгоритмов массовых задач на непересекающиеся множества, которые соответствуют граф-схемам (блок-схемам) полиномиальных (P) или переборных (NP) алгоритмов. Этим обосновывается достаточное условие, которое, собственно, и подтверждает, что $P \neq NP$.

Для цитирования
Малинина Н.Л. Необходимые и достаточные условия разделения структур алгоритмов на непересекающиеся множества: полиномиальные и переборные алгоритмы // Вестник Российского университета дружбы народов. Серия: Инженерные исследования. 2022. Т. 23. № 2. С. 108–116. http://doi.org/10.22363/2312-8143-2022-23-2-108-116

Introduction

The $PvsNP$ problem, or the Cook problem [1; 2], as it is called, ranks first on the list of millennial problems. This article continues a twelve-year-old project presented by the author at the International Mathematical Congress in Hyderabad [3]. The relationship between the classes P and NP is considered in the theory of computational complexity (a branch of the theory of computation), which studies the resources needed to solve a certain problem. Since 1971, many topologists, algorithm designers, and other scientists have devoted their time and effort in order to solve it. The most cited author who writes on the topic is A. Razborov [4]. There is also a website dedicated to this problem.[1] It contains links to 116 articles on possible solutions of the problem.

Let us dwell only on those articles that were published in refereed journals. One of them is an article by M. Giannakakis [5]. The article does not solve the problem itself, but only verifies that some particular approach to the proof does not work. There are a few more references to papers proving that $P = NP$. They are mainly devoted to the successful attempts to create polynomial algorithms for some special cases, and thus authors pretended that $P = NP$. However, we should not forget that the number of mass problems that we cannot accurately solve without the use of enumeration algorithms is expanding every day.

A little less works have been written to prove that $P \neq NP$. It is almost impossible to carefully review all the works (there are more than 50 of them). A part of the works is based on the fact that there are models (specific special cases) for which polynomial algorithms cannot be found and, accordingly, it was concluded that $P \neq NP$. In particular, this is the work of R. Valeyev [6]. An interesting article was presented by A. Anilla [7], in which he comes to the proof of $P \neq NP$ by investigating computational complexity applying the principle of increasing entropy. In the work of V. Ivanov [8], the proof is based on more accurate estimates of the lower bounds on the time complexity, which are valid for all algorithms for solving. The most recent work, which is also devoted to the proof of $P \neq NP$ problem [9] is posted on the Internet, and the opinion of the mathematical community is also not yet known. A number of proofs, in particular, the works of Anand, Deliokar, Vian, Barbos[2] and some others, have got responses and were critisized.

It should be noted that only the proof of the fact that $P \neq NP$ will give us nothing for solving problems from the practical area. It is necessary to divide the tasks according to some structural features. This will allow

[1] The P-versus-NP page. Available from: https://www.win.tue.nl/~gwoegi/P-versus-NP.htm (accessed: 26.09.2016).

[2] The P-versus-NP page. Available from: https://www.win.tue.nl/~gwoegi/P-versus-NP.htm (accessed: 26.09.2016).

at the first stage to understand how difficult the algorithm for solving a practical problem will be.

What is the most important thing we know and don't know about the $PvsNP$ problem [2]?

1. We know that for the mass problems belonging to the P class, we can create linear or polynomial algorithms, and obtain exact solutions or solutions that are of good approximation.

2. We know that for the mass problems from the NP class, we can get exact solutions if we will use enumeration algorithms or we can get acceptable solutions if we will use nonlinear or exponential algorithms.

3. We do not know whether these areas are divided into disjoint sets and how to determine the belonging of the task (or the algorithm that solves it) to the concrete area?

It seems that it is possible to approach the solution of the $PvsNP$ problem from the other side, from the standpoint of the need to solve practical problems. This will allow applying the achievements of the graph theory and the theory of algorithms.

1. The problem of computation

So, in order to search for the solution of the $PvsNP$ problem, we should be interested in the possibility of computing, that is, in the possibility to create algorithms and programs that solve some problems. Wherein it is not only desirable, but it is necessary to obtain economical algorithms and programs. On the one hand, it is necessary to reduce the resources of time and memory needed for the solution of problems from the NP class. On the other hand, it takes a lot of energy resources to build and maintain the powerful servers that are needed to solve such problems. The brute-force algorithms also need energy. The number of mass problems from the NP class multiplies every year. And these are not only obligatory and vital tasks, such as management of complicated systems in different social and technic areas, but also entertaining tasks such as games and social media.

We perform calculations using computers. We know that a computer can process data or solve only those problems (programs) that correspond to the Church – Turing thesis [10; 11], that is, algorithms and, accordingly, programs must have the property of effective recursiveness.

How can we achieve this magical property? This property is prescribed by the Markov principle of normalizing: "an algorithm must be normal in order to be processed by a Turing machine" [12]. Basically, all algorithms are normalizable, this is confirmed by the practice of developing algorithms and programs [12].

It is well known that all algorithmic schemes and their compositions (up to equivalence) lead to normal algorithms. Operators in algorithms are implemented in a specific order or in the order of their numbering. In turn, the numbering of operators can be performed if the set of operators is recursive. However, none of the algorithmic systems still has any predetermined way of numbering the operators [12; 13]. It is also well known that in order to create the property of recursiveness, it is necessary to expand the alphabet of the algorithm and it is possible to be done by adding only one letter [12]. It is also known that algorithms can be visualized using directed graphs.

The next question arises: how to add this letter and add it somehow automatically? How to make an arbitrary algorithm normal or efficiently recursive? This problem was solved in 1972 in L. Malinin's doctoral dissertation, but was published only in 2009 in the book "Graph Isomorphism in Theorems and Algorithms" [14].[3]

2. Extension of graph theory and connection with theory of algorithms

Let's turn to graph theory. Before the publication of the [14], devoted to the solution of the graph isomorphism problem, there was no solution to the problem of a clear definition of the possible duality of graphs in graph theory.

Everyone knows that an edge graph (algorithm's graph-scheme) always has a dual vertex graph (algorithm's block-scheme). But a vertex graph does not always have its dual edge graph [15] (Figure 1).

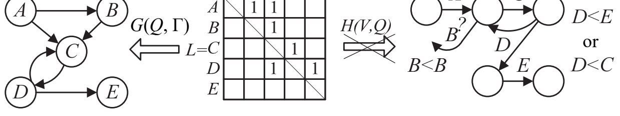

Figure 1. On the non-mandatory duality between edge and vertex graphs

The extension of graph theory in [14] allows one to solve this problem. A wide study of the dua-

[3] The English variant of the book may be seen on ResearchGate. Available from: https://www.researchgate.net/publication/358570634 (accessed: 26.09.2016).

lity of graphs was carried out in this work, and the necessary and the sufficient conditions were proved for the adjacency matrix to be simultaneously the adjacency matrix of both edge and vertex graphs. It is necessary to present the main theorems, which lead to the concepts of duality of graphs and show us, how to achieve this duality.

In the basic, or main, theorem, the conditions are proved that the adjacency matrix must meet in order for the graph corresponding to it to have duality properties.

Theorem 1 "On a quasi-canonical adjacency matrix."[4] A theorem on the quasi-canonical adjacency matrix determines both necessary and sufficient conditions that the direct path's L matrix has a dual nature that is at the same time it might be both E – adjacency matrix of the G graph's vertices and R – adjacency matrix of the H graph's edges on condition that they may have not equal cyclomatic numbers. Theorem was proved for the case of the directed graphs.[5]

It is given: a set $Q = \{q_i\}$ and $L = QQ$ by the way of $q_i < q_j$, and $L = \left\|l_{ij}\right\|_1^n = \left\|e_{ij}\right\|_1^n$ for the $G(Q, \Gamma)$ graph. Then $\left\|l_{ij}\right\|_1^n = \left\|r_{ij}\right\|_1^n = R$ for the $H(V, Q)$ graph only if

$L = \left\|l_{ij}\right\|_1^n$ generates $C_n = \left\|c_{ij}\right\|_1^n = [0]$.

A minor $\left|l_{ij}\right|_1^{n-1}$ of the every $l_{ij} = 1$ generates

$$C_{n-1} = \left\|c_{ij}\right\|_1^n = [0], \tag{1}$$

where

$$c_{ij} = l_{ij}\left(\Delta_{j/i} s_{ij} + \Delta_{i/j} s_{ij}\right),$$

$$\Delta_{j/i} s_{ij} = \left(s_{ij} - \min_j s_{ij}\right)_i,$$

$$\Delta_{i/j} s_{ij} = \left(s_{ij} - \min_i s_{ij}\right)_i,$$

$$k = n, (n-1),$$

$$s_{ij} = l_{ij}\left(\sum_{\substack{i=1\\j}}^{k} l_{ij} + \sum_{\substack{j=1\\i}}^{k} l_{ij}\right),$$

$$\left(\min_j s_{ij}\right)_i = \min_{j/i} s_{ij} \in \{s_{ij} \neq 0\},$$

$$\left(\min_i s_{ij}\right)_j = \min_{i/j} s_{ij} \in \{s_{ij} \neq 0\}.$$

For the proof of the theorem it must be testified that the L matrix, which meets the conditions (1) and is considered as the R matrix – the adjacency matrix of H graph's edges, has all the information for a single-valued representation of the F matrix – the adjacency matrix of H graph's vertices. The proof is presented in [14].

Cyclomatic numbers of graphs always satisfy the condition

$$\nu(G_q) \geq \nu(H_q). \tag{2}$$

Condition (2) reflects a certain degeneracy of the duality (quasi-duality) of the quasi-canonical adjacency matrix. In addition, this condition reflects the possibility of the presence of complex vertices in the H_q graph.

Theorem 1 defines the conditions for the existence of a quasi-canonical adjacency matrix, although in practical applications such ready-made matrices can occur only by chance. It becomes very important to find a way to transform any arbitrary matrix of direct paths, which does not satisfy the requirements of theorem 1, to the required form. The transformation of the matrix L must be such that the system of relations between the initial elements remains unchanged, that is, the transformation must be conservative with respect to the system of binary relations defined on the $Q = \{q_i\}$ set.

Transformation of the direct path matrix to a quasi-canonical form. Definitions:

1. By the *conservative transformation of the binary relation* between the two q_i and q_j elements we'll denote such a transformation, which will allow either to insert the additional elements into the $Q = \{q_i\}$ set or to exclude them without changing the relation between the (q_i, q_j) elements. Such a transformation may be based on the transitivity property of the binary relation. For example, the initial pair is defined as $(q_i, q_j) \in Q$. Let the elements be connected by the relation: $q_i < q_j$. Let's accept two conditions: both $q_i < q_z$ and $q_z < q_j$, and transform the initial expression. We'll find that $q_i < q_z < q_j$. It is obvious, that two relations both $q_i < q_j$ and

[4] The numbers of the theorems in the article correspond to the numbers of the theorems in the book.

[5] The directed graph can be transformed to the undirected graph by doubling the edges of the graph.

$q_i < q_z < q_j$ are equivalent according to the initial pair of the elements. Therefore, the inserting of the q_z element into the $q_i < q_j$ relation is the conservative operation regarding to this relation in the initial pair.

2. Let us settle that by the *Δn-transformation of the L matrix* we will comprehend the addition of one row (both line and column) to the L matrix at the condition of replacement the $q_x < q_y$ relation with the pair of binary both $q_x < q_{n+1}$ and $q_{n+1} < q_y$ relations. It is evident, that the Δn-transformation is the conservative operation regarding the binary relation in the initial (q_i, q_j) pair and does not break such a structural similarity criterion as the binary relation's system.

Theorem 2 "On a quasi-normalization of the L matrix's binary relations." Any direct path's $\|l_{ij}\|_1^n$ matrix can be transformed to the quasi-canonical (quasi-normal) $\|l_{ij}\|_1^{n+s_q}$ form, where $s_q \le n^2 - 1$, by means of applying the Δn-transformation to such s_q elements of the $\|l_{ij}\|_1^n$ matrix, which do not satisfy to the conditions of theorem 1.

The convergence of Δn-transformation and the proof of the theorem are presented in [14]. If we correlate said above with the theory of algorithms and Markov's thesis, then the proved Δn-transformation automatically adds the missing letter to the alphabet of an arbitrary algorithm, and makes it normal or recursive. It is proved that the Δn-transformation (normalizing) of the algorithm is local, although with the help of some tweaks it can become linear. In the special case when $\nu(G_q) = \nu(H_q)$, the matrix $L_q = R_q$ is called canonical or normal. For the case of strict duality (the cyclomatic numbers are equal), the following theorem was proved.

Theorem 4 "On the canonical adjacency matrix." Let the G initial graph be specified as the $\|e_{ij}\|_1^n$ matrix – an adjacency matrix of vertices, which corresponds to the quasi-canonical $\|r_{ij}\|_1^{n+s_q}$ matrix – an adjacency matrix of edges of the connected edge H_q graph. In order that the H_q graph's cyclomatic $\nu(H_q)$ number might be equal to the initial G graph's cyclomatic $\nu(G)$ number, it is necessary and sufficient for all the H_q graph's vertices to be simple, or, for every $r_{xy} = 1$ such condition must be fulfilled:

$$\left.\begin{array}{l} \text{If } \sum\limits_{\substack{i=1 \\ j=y}}^{n+s_q} r_{ij} \ge 1, \text{then } \sum\limits_{\substack{j=1 \\ i=x}}^{n+s_q} r_{ij} = 1 \\ \text{If } \sum\limits_{\substack{j=1 \\ i=x}}^{n+s_q} r_{ij} \ge 1, \text{then } \sum\limits_{\substack{i=1 \\ j=y}}^{n+s_q} r_{ij} = 1 \end{array}\right\}. \quad (3)$$

As a result of graph normalizing and subsequent ordering, we get an edge graph in the form of a Koenig graph. In [14] it is proved that canonical graphs without contours, obtained as a result of normalizing, possess the recursive property, which is based on the partition of canonical adjacency matrices into mutually non-intersecting submatrices. This allows us to build recurrent local algorithms for their ordering. Such possibility gives the graphs the property of effective recursiveness (numbering is carried out in one pass along the set of rows of the matrix as it is required in [14]) to the graphs (algorithms and programs with such a structure).

The obtaining of an ordered Koenig graph is a necessary condition, but not a sufficient one. Theoretically, we get the opportunity to create a normal algorithm in the form of an ordered Koenig graph. However, the possible presence of complex vertices in the original contour graph and in the resulting graph H_q leads to some problems. The cyclomatic number of some graphs grows under the Δn-transformation, that is, it is not entirely clear what happens to graphs that have contours. So, the possibility of automating the algorithm normalizing process does not yet solve the problem of proving $P\text{vs}NP$.

We are faced with another problem – how to divide mass tasks into classes so that, according to certain signs of the task graph, it is immediately clear to which class the task belongs: P or NP. Here we face the problem of graph complexity. A characteristic of graph complexity is the cyclomatic number ν. Therefore, one should turn to such a characteristic of graph complexity as a cyclomatic number and figure out whether it is a graph invariant and, if so, in what cases.

3. Cyclomatic number and isomorphism

Basically, in order to compare graphs we have a sufficient number of invariants, which can help to compare graphs with each other. And among the invariants there is such a characteristic of the graph as the cyclomatic number ν. There are works of [15–19], which indicate that the cyclomatic number is equal to the maximum number of independent cycles in the graph and thus becomes the characteristic of the complexity of the graph. In order to understand whether the cyclomatic number is an invariant that we can rely on, we should turn to the study such a transformation of graphs as the transformation of a vertex graph into an edge graph; of the resulting edge graph back to the vertex graph, and so on [14]. Let's call this operation as the graph conversion (Figure 2).

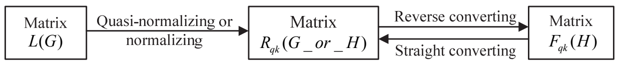

Figure 2. Graph transformation scheme

By straight conversion we'll call the operation of constructing a vertex graph from a given edge graph. An arbitrary directed graph can be subjected to the operation of the straight conversion if and only if the adjacency matrix of its vertices has a canonical ($\nu = \text{const}$) or quasi-canonical form ($\nu \uparrow$).

By the reverse conversion we'll call the operation of constructing an edge graph from a given vertex graph. A graph can be subjected to the operation of reverse conversion if and only if the adjacency matrix of its vertices has a canonical or quasi-canonical form. Both conversion operations were thoroughly discussed in [14]. It is necessary to present only two theorems.

Theorem 9. If the graph H_1 in the process of its sequential direct conversion generates only canonical graphs H_{kj} $(j = 1, 2, 3 \dots, M)$, then the number of vertices of these successively obtained graphs is determined by a linear dependence on the number of the conversion operation $(j-1)$, that is

$$n_j = n_1 + \Delta n^*_{(j-1)}. \tag{4}$$

Theorem 10. If the H_1 graph in the process of its consecutive straight converting generates both the canonical and the quasi-canonical graphs or only the quasi-canonical graphs, then the numbers of the vertices of these graphs, received step by step, are determined by the following expression:

$$n_j = n_1 + \sum_{\xi=1}^{\xi=(j-1)} \Delta n_\xi, \tag{5}$$

where

$$\Delta n_\xi = \Delta n_{(\xi-1)} + \Delta \nu(H_{(\xi-1)}, \tag{6}$$

where $\xi = 1, 2, \dots (j-1)$; $j = 1, 2, 3, \dots M$.

It was shown that the increase in the number of graph vertices obtained by sequential straight conversion is associated with the cyclomatic number and the type of conversion (canonical or quasi-canonical), and the increase in the cyclomatic number during this conversion depends on the structure of the graph.

The proved theorems [14] represent that all directed graphs can be divided into two classes:

1) graphs for which the cyclomatic number is always an invariant of the direct conversion;

2) graphs for which the cyclomatic number is not an invariant at some steps or at all steps of the direct conversion.

As a result of a thorough study of converting operations and determining the properties of graph structures with respect to combinations of various types of vertices and edges between them the necessary and sufficient signs of these graphs were identified. The proved theorems are given in [14]. Also, the concept of a path and a contour in a directed graph was considered more precisely. Real processes can only correspond to such circuits that have at least one "input" and at least one "output." The concept of path and contour were already introduced much earlier [15; 16; 18], but in [14] a function was added that allows one to distinguish paths in a graph from one another. This sign was determined using the function of the sums of the degrees of the vertices through which the path passes. It turned out that these functions can be divided into two classes, which are described in [14]. Accordingly, paths can also be divided into two classes. The proved theorems are presented in [14].

The study of various paths in graphs led to the study of various combinations of intervals between the vertices of the graph [14]. The vertices of the graph were determined as the positive (one input and many outputs) one and the negative (many inputs and one output) one. In addition the vertices of the graph were also divided into the simplest (one input and one output), the simple (one/several input and several/one outputs) and complex (several inputs and several outputs). Particular attention was drawn to the interval of the l_{31} type, which has complex vertices at both ends (Figure 3, *a, b*).

The main feature of this interval l_{31} is that there is a negative vertex at the input end, and a positive vertex on the output end. Such an interval, during sequential conversion, first turns into a complex vertex (Figure 3, *c*). At the next conversion step this complex vertex (one) is converted into four independent cycles (Figure 3, *d*). The appearance of new cycles causes an increase in the cyclomatic number. Accordingly, the complexity of the graph also grows.

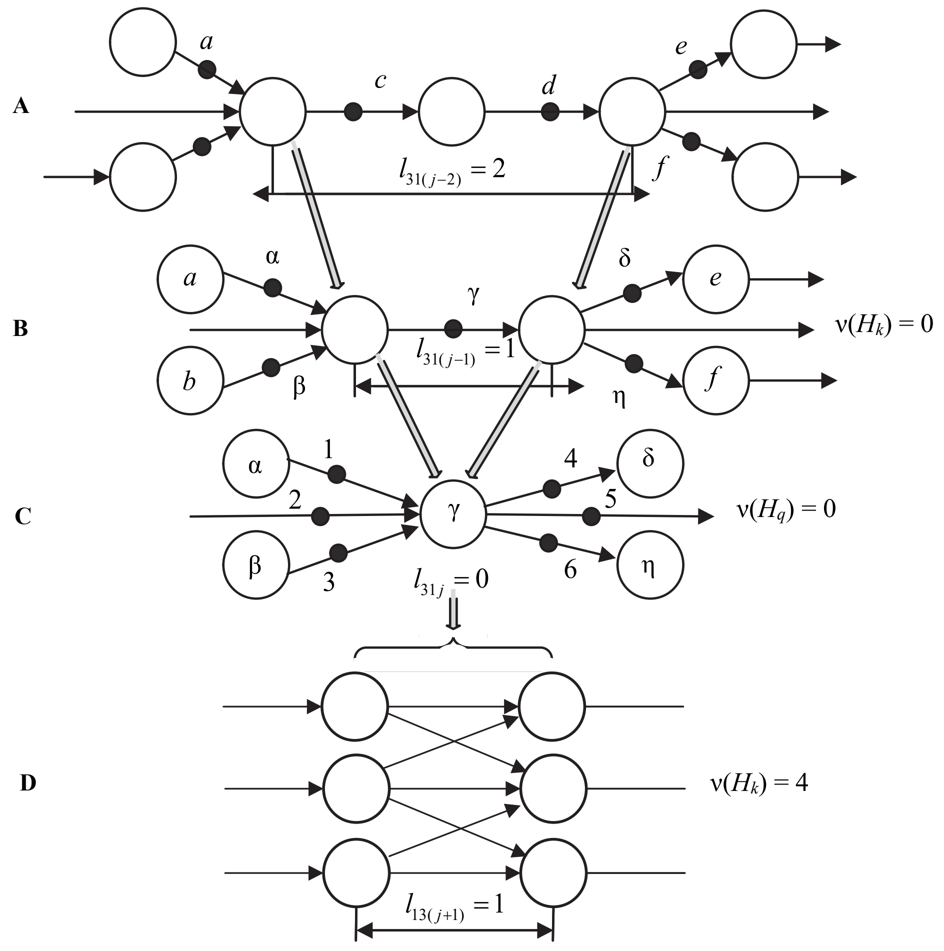

Figure 3. The transformation of the interval l_{31} into a complex vertex and the appearance of the independent cycles

As a result of research, it turned out that directed graphs can be divided into three classes (the disjoint sets):

1. ***Holonomic graphs.*** For them, the cyclomatic number is a regular conversion invariant, regardless of the number of sequential conversion steps. Due to this, the number of graph vertices obtained from the original graph as a result of its sequential conversion depends linearly on the number of conversion steps. These graphs should not contain contours and intervals of type l_{31}.

2. ***Bounded heteronomous graphs.*** Such graphs have a heteronomy boundary in terms of the number of sequential conversion steps. Until this limit is reached, the cyclomatic number is not a regular conversion invariant. After reaching this boundary, as a result of the next conversion step, a holonomic graph is generated and the cyclomatic number becomes a regular conversion invariant regardless of the number of steps of further sequential conversion. The structure of such graphs may contain intervals of type l_{31}, but should not have contours.

3. ***Progressive heteronomous*** graphs do not have heteronomic boundaries in terms of the number of steps of sequential conversion. As a result, the cyclomatic number of a progressive heteronomous graph does not become a regular invariant of sequential conversion, for any, however large, number of conversion steps. The structure of such graphs contains both contours and intervals of the l_{31} type.

The comparison of graphs and algorithmic schemes

Initial graph's view	Ordered graph	No in Table 1 [14]	Complete name of algorithm's scheme
Гамильтонов граф $G^{(H)}$ при $\sum_j l_{ij} = 2$ $\big(j = 2,3,\dots,(n-1)\big)$	$G^{(H)}\left(L^{(H)}\right)$	6	Normal single-channel two-address algorithm's flow block
	$G_K^{(H)}\left(R_K^{(H)}\right)$	7	Normal single-channel two-address algorithm's flowgraph by Kaluznin
	$H_K^{(H)}\left(F_K^{(H)}\right)$	8	Normal ordinary single-channel two-address algorithm's flowgraph
		9	Generalized normal single-channel two-address algorithm's flowgraph
	$D_K^{(H)}\left(F_K^{(H)}\right)$	10	Normal operator single-channel two-address algorithm's flowgraph
Произвольный граф $G^{(K)}$ при $\sum_j l_{ij} = 2$ $\big(j = 2,3,\dots,(n-1)\big)$	$G^{(K)}\left(L^{(K)}\right)$	16	Two-address algorithm's flow block with arbitrary number of channels
	$G_K^{(K)}\left(R_K^{(K)}\right)$	17	Normal two-address algorithm's flowgraph by Kaluznin with arbitrary number of channels
	$H_K^{(K)}\left(F_K^{(K)}\right)$	18	Normal ordinary two-address algorithm's flowgraph with arbitrary number of channels
		19	Generalized normal two-address algorithm's flowgraph with arbitrary number of channels
	$D_K^{(K)}\left(F_K^{(K)}\right)$	20	Normal operator two-address algorithm's flowgraph with arbitrary number of channels
Произвольный граф G	$G(L)$	26	N-address algorithm's flow block with arbitrary number of channels
	$G_K(R_K)$	27	Normal conjugate N-address algorithm's flowgraph with arbitrary number of channels
	$H_K(F_K)$	28	Normal ordinary N-address algorithm's flowgraph with arbitrary number of channels
		29	Generalized normal N-address algorithm's flowgraph with arbitrary number of channels
	$D_K(F_K)$	30	Normal operator N-address algorithm's flowgraph with arbitrary number of channels
Полный граф G_0^*	$G_0^*(L_0^*)$	36	Complete algorithm's flow block
	$G_{0u}^*(R_{0u}^*)$	37	Normal conjugate complete algorithm's flowgraph
	$H_{0u}^*(F_{0u}^*)$	38	Normal ordinary complete algorithm's flowgraph
		39	Generalized normal complete algorithm's flowgraph
	$D_{0u}^*(F_{0u}^*)$	40	Normal operator complete algorithm's flowgraph
$G^{(K)}, G, G_0^*$	$G^{(H)}\left(L^{(H)}\right)$	41	Normal algorithm's flow block (single-channel two-address)

The correspondence between graph structures and various block diagrams and graph diagrams of various algorithms (Table) is investigated in [14, chapter 4], where presented in the form of Table 1.

Conclusion

So, we can conclude that the set of all algorithms is divided into three classes (or three disjoint sets), according to the above partition of the set of directed graphs. After the operation of normalizing, the graphs, and, consequently, the algorithms, that have a similar structure, acquire the properties of recursiveness.

For the holonomic graphs, the cyclomatic number becomes an invariant. Algorithms, which after the operation of normalizing will have a similar structure, automatically receive the property of effective recursiveness and will belong to the polynomial area.

Bounded-heteronomic graphs must be subjected to some finite number of direct conversion steps in order for the cyclomatic number to become an invariant. Algorithms with such a structure can be called reducible to a set of polynomial algorithms.

Progressively-heteronomic graphs will never have a cyclomatic number invariant. Therefore, algorithms with a similar structure will always belong to the non-polynomial area, although in particular cases the normalization operation can reduce the number of search options.

And finally, the main thing that can be said in support of $P \neq NP$ thesis.

The necessary condition: in order for the arbitrary massive task be implemented with the help of the computer, it must have a form of the ordered Koenig graph, which can be obtained by the operation of normalizing an arbitrary graph of the task.

But the sufficient condition is divided into three parts, because the arbitrary graphs are divided into three non-intersecting classes according to their structural properties. Therefore, the algorithms that correspond to them also fall into three non-overlapping classes. There will remain a class of algorithms that cannot be reduced to the class of polynomials. They will always remain exhaustive, that is, NP-hard.

All this proves that $P \neq NP$.

References

1. Cook SA. The complexity of theorem-proving procedures. *Conference Record of Third Annual ACM Symposium on Theory of Computing.* New York: Association for Computing Machinery; 1971. p. 151–158. https://doi.org/10.1145/800157.805047

2. Gary M, Johnson D. *Computing machines and intractable problems.* Moscow: Mir Publ.; 1982. (In Russ.)

3. Malinina NL. On a principal impossibility to prove *P = NP*. *International Congress of Mathematicians.* Hyderabad: Hundistan Book Agency; 2010. p. 484–485.

4. Razborov AA. Lower bounds for the polinomial calculus. *Computational Complexity.* 1998;7:291–324.

5. Yannakakis M. Expressing combinatorial optimization problems by liner programs. *Journal of Computer and System Sciences*. 1991;43:441–466.

6. Valeyev R. The lower border of complexity of algorithm of elementary NP-complete task. *World Applied Science Journal.* 2013;24(8):1072–1083.

7. Annila A. Physical portrayal of computational complexity. *ISRN Computational Mathematics.* 2009;2012: 321372. https://doi.org/10.5402/2012/321372

8. Ivanov V. A short proof that NP is not P. *International Journal of Pure and Applied Mathematics.* 2014; 94(1):81–88. http://doi.org/10.12732/ijpam.v94i1.9

9. Dowd M. *On the provability of P = NP*. 2020:1–13. Preprint. Available from: https://www.researchgate.net/publication/339426546_On_the_Provability_of_PNP (accessed: 22.02.2020).

10. Church A. A note on the Entscheidungsproblem. *The Journal of Symbolic Logic.* 2014;1(1):40–41. https://doi.org/10.2307/2269326

11. Turing A. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society.* 1937;s2–42(1):230–265.

12. Markov A, Nagorny N. *The theory of algorithms.* Boston: Kluwer Academic Publiser; 1988.

13. Glushkov VM. *Theory of algorithms*. Kiev: KVIRTU PVO; 1961. (In Russ.)

14. Malinin LI, Malinina NL. *Graph isomorphism in theorems and algorithms.* Moscow: Librocom Publ.; 2009. (In Russ.)

15. Ore O. *Theory of graphs.* Rhode Island: American Mathematical Society; 1962.

16. Berge Cl. Théorie des graphes et ses applications. *Collection universitaire de Mathématiques* (No. 2). Paris: Dunod Editeur; 1958.

17. Zykov AA. *The theory of finite graphs.* Novosibirsk: Nauka Publ.; 1969. (In Russ.)

18. Harary F, Palmer E. *Graphical enumeration.* London: Academic Press; 1973. https://doi.org/10.1016/c2013-0-10826-4

19. Harary F. *Graph theory.* New York: Basic Books; 1972.

About the author

Natalia L. Malinina, Candidate of Physical and Mathematical Sciences, Associate Professor of the Department 604, Aerospace Faculty, Moscow Aviation Institute (National Research University), 4 Volokolamsk Shosse, Moscow, 125993, Russian Federation; ORCID: 0000-0003-0116-5889, eLIBRARY AuthorID: 502378; malinina806@gmail.com