Reduction of the Graph Isomorphism Problem to Equality Checking of n-variables Polynomials and the Algorithms that use the Reduction

Alexander Prolubnikov

Omsk State University, Omsk, Russian Federation
a.v.prolubnikov@mail.ru

Abstract. The graph isomorphism problem is considered. We assign modified characteristic polynomials for graphs and reduce the graph isomorphism problem to the following one. It is required to find out, is there such an enumeration of the graphs vertices that the polynomials of the graphs are equal. We present algorithms that use the redution and we show that we may check equality of the graphs polynomials values at specified points without direct evaluation of the values. We give justification of the algorithms and give the scheme of its numerical realization. The example of the approach implementation and the results of computational experiments are presented.

Keywords: graph isomorphism, complete invariant, computational complexity

1 The graph isomorphism problem

In the graph isomorphism problem (GI), we have two simple graphs G and H. Let $V(G)$ and $V(H)$ denote the sets of vertices of the graphs and let $E(G)$ and $E(H)$ denote the sets of their edges. $V(G) = V(H) = [n]$. An *isomorphism* of the graphs G and H is a bijection $\varphi : V(G) \to V(H)$ such that for all $i, j \in V(G)$

$$(i, j) \in E(G) \Leftrightarrow (\varphi(i), \varphi(j)) \in V(H).$$

If such a bijection exists, then the graphs G and H are *isomorphic* (we shall denote it as $G \simeq H$), else the graphs are not isomorphic. In the problem, it is required to present the bijection that is an isomorphism or we must show non-existence of such a bijection.

We may state GI using adjacency matrices of graphs. Let $(A)_{ij}$ denote the ij-th element of a matrix A. The *adjacency matrix* of a graph G is a matrix $A(G)$ which has dimension of $n \times n$. Elements of this matrix are defined as follows:

$$(A(G))_{ij} = \begin{cases} 1, & \text{if } (i, j) \in E(G), \\ 0, & \text{else.} \end{cases}$$

Let $A(G)$ and $A(H)$ be adjacency matrices of the graphs G and H. Then

$$G \simeq H \Leftrightarrow \exists \varphi \in S_n : \ A(H) = P_\varphi A(G) P_\varphi^\top,$$

In: A. Kononov et al. (eds.): DOOR 2016, Vladivostok, Russia, published at http://ceur-ws.org

where S_n is a symmetric group on $[n]$ and

$$(P_\varphi)_{ij} = \begin{cases} 1, & \text{if } i = \varphi(j), \\ 0, & \text{else.} \end{cases}$$

By this formulation of the problem, the two graphs are isomorphic if and only if we can obtain adjacency matrix of the first graph from adjacency matrix of the second one by some permutation of its rows and columns.

GI is one of the fundamental problems of discrete mathematics and there are numerous applications where it arises. For example, without solving GI, we cannot practically solve the following problem. In the problem we need to find an n-vertices graph that has some specified property. We may search the graph doing the exhaustive search on all labelled n-vertices graphs. But, since there are $n!$ of isomorphic labelled graphs, we must check the property only for nonisomorphic labeled graphs during this search. For this purpose, we need to efficiently solve GI. Else our memory expenses would be too high.

GI belongs to NP class since it takes $O(n^2)$ time to check whether some bijection of $V(G)$ onto $V(H)$ is an isomorphism. It is not proved that the problem is NP-complete and there is no any polynomial algorithm has been obtained for the general case of the problem.

GI is solvable in polynomial time for some classes of graphs: for trees [1], for graphs with bounded genus [2], for graphs with bounded multiplicity of their adjacency matrix eigenvalues [3], for graphs with bounded degree of their vertices [4] and for some other restricted classes of graphs. The more regular structure of the graphs, the harder to obtain solution of GI for them. Such classes as regular graphs, strongly regular graphs, isoregular graphs give the instances of GI that cannot be solved in polynomial time by existing algorithms.

For GI, the designed algorithms may be divided in two classes. The first class algorithms are designed to solve GI for some restricted cases and the second class algorithms are designed to solve the problem for the general case. The examples of the algorithms which belongs to the first class are the algorithms that solve GI for the mentioned above classes of graphs. The Ullman algorithm [5], the Schmidt-Druffel algorithm [6], B. McKay's NAUTY algorithm [7] belong to the second class of the algorithms. These algorithms are called practical, since they are exponential in the worst case but they are effective for most of the input graphs.

It stated in [8] that GI may be solved in quasipolynomial time. It is in the process of proof-checking by other mathematicians at the moment when this paper is written.

To solve GI, the algorithms checks *graph invariants* during their implementation. Graph invariants are properties of graphs which are invariant under graph isomorphisms, i.e., graph invariant is a function f such that $f(G) = f(H)$ if $G \simeq H$. The examples of graph invariants are such graph properties as connectivity, genus, degree sequence, the characteristic polynomial of adjacency matrix, its spectrum. A graph invariant $f(G)$ is called *complete* if the equality $f(G) = f(H)$ implies that $G \simeq H$. Let us consider some of the well-known graph invariants.

The Weisfeiler-Lehman method is an example of the approach to check invariant characteristics of graphs. Applying this method, we perform iterative classification of

graphs vertices. As a result, we have such colourings of vertices that we may use it to distinguish non-isomorphic graphs. Using this method, we can solve GI for large class of graphs in polynomial time. But it is shown [9] that there exists such pairs of non-isomorphic graphs on n vertices that they are cannot be distinguished by k-dimensional Weisfeiler-Lehman algorithm in polynomial time for $k = \Omega(n)$. This implies that, for the general case, this method not solve GI in polynomial time.

The procedures of graph canonization give complete graph invariants. For a graph G, using some procedure of graph canonization, we obtain its canonical form that is some labeled graph. Two graphs are isomorphic if and only if they have the same canonical form. Using canonical form of the graph, we may compute its canonical code. Canonical code $c(G)$ of a graph G is a bit string such that $G \simeq H$ if and only if $c(G) = c(H)$. The example of a canonical code is the code $c_0(G)$:

$$c_0(G) = \max_{\pi \in S_n} \{(A)_{\pi(1)} || (A)_{\pi(2)} || \dots || (A)_{\pi(n)}\},$$

where "||" denotes concatenation of bit strings and $(A)_i$ denotes the i-th row of the adjacency matrix $A(G)$. For some classes of graphs, canonical codes may be computed in polynomial time [1], [10]. Every complete invariant gives a way for graph canonization.

In [11], complete invariant for hypergraphs is presented. This complete invariant is not a result of canonization. For the case of simple graphs, this invariant is a system of $n^2 + 1$ polynomials over a field of characteristic q, where q is a prime number or zero.

In our work, we assign modified characteristic polynomials for graphs and reduce the graph isomorphism problem to the following one. It is required to find out, is there such an enumeration of the graphs vertices that the modified characteristic polynomials of the graphs are equal. The modified characteristic polynomial of a graph is not a graph invariant in the sense of its immutability to different enumerations of the graph vertices. But the polynomial is complete invariant of a graph in the sense that there is no such enumeration that gives equal polynomials for non-isomorphic graphs.

In addition to the reduction and the theoretical algorithm that we have presented in [12], using the same approach, we present the practical algorithms for GI in this work. These algorithms not check equality of the graphs polynomials by checking equality of their coefficients but they check equality of the polynomials values in randomly taken points. We give justification of the algorithms and give the scheme of its numerical realization. The example of the approach implementation and the results of computational experiments are presented.

2 The modified characteristic polynomial of a graph

Let us consider the characteristic polynomial of a graph and some of its modifications which have been used for characterization of graphs by their structural properties. The characteristic polynomial of a graph G is the polynomial

$$\chi_G(x) = \det(A(G) - xE),$$

where x is a variable and E is the identity matrix. Let $D(G) = diag(d_1, \dots, d_n)$ be a diagonal matrix, where d_i is a degree of vertex $i \in V(G)$.

Some modifications of the characteristic polynomial are considered in [13]. The examples are the graph Laplacian $L(G)$ that is defined as

$$L(G) = D(G) - A(G),$$

the signless graph Laplacian $|L(G)|$ that is defined as

$$|L(G)| = D(G) + A(G),$$

and some other modifications which may be generalized by the polynomial $\xi_G(x, y)$:

$$\xi_G(x, y) = \det(xE - (A(G) - yD(G))).$$

Seidel polynomial [14] is another modification of the characteristic polynomial that is obtained by modification of a graph adjacency matrix. It is the polynomial $\zeta_G(x)$:

$$\zeta_G(x) = \det(xE - (F - E - 2A(G))),$$

where $(F)_{ij} = 1$ for $i, j = \overline{1, n}$.

A generalization of the characteristic polynomial is the polynomial that is presented in [15]. It is a polynomial $\psi_G(x, y, \lambda)$ of the form

$$\psi_G(x, y, \lambda) = \det(A(x, y) - \lambda E),$$

where $A(x, y)$ is the matrix, derived from A, in which the 1s are replaced by variable x and 0s (other than the diagonals) are replaced by variable y.

None of the presented above modifications of the characteristic polynomial is a complete graph invariant.

The modified characteristic polynomial of a graph. Variables of the polynomials above have no connection with graph vertices. We modify the characteristic polynomial $\chi_G(x)$ of a graph G on n vertices assigning variable x_i to $i \in V(G)$. Let $x_1, \ldots, x_n$ be independent variables and let $X = diag(x_1, \ldots, x_n)$.

Definition 1. *For a graph G, $|V(G)| = n$, $\eta_G(x_1, \ldots, x_n)$ is a polynomial of the form*

$$\eta_G(x_1, \ldots, x_n) = \det(A(G) + X). \tag{1}$$

For graphs on $n = 1, 2, 3$ vertices, the polynomials of the form (1) are the following ones:

1) $n = 1$: x_1;

2) $n = 2$: x_1x_2, $x_1x_2 - 1$;

3) $n = 3$: $x_1x_2x_3$, $x_1x_2x_3 - x_1$, $x_1x_2x_3 - x_1 - x_3$, $x_1x_2x_3 - x_1 - x_2 - x_3 + 2$.

It is clear that we have different polynomials for non-isomorphi graphs no matter what enumeration of its vertices we use for $n = 1, 2, 3$.

For a subset c of $V(G)$, let x_c be a product of the form $\prod_{i \in c} x_i$ and $A(G)_c$ be a determinant of the submatrix of $A(G)$ that is obtained by deleting of the rows and columns of $A(G)$ where numbers are belong to the subset c. $A(G)_c$ is the coefficient of $\prod_{i \in c} x_i$ in the polynomial $\eta_G(x_1, \ldots, x_n)$. Having c and $\varphi \in S_n$, let $\varphi(c)$ be the image of c: $\varphi(c) = \{\varphi(i) \mid i \in c\}$. For $x = (x_1, \ldots, x_n)$, $x_\varphi = (x_{\varphi(1)}, \ldots, x_{\varphi(n)})$. The following theorem holds.

Theorem 1. *$G \simeq H$ and $\varphi: V(G) \to V(H)$ is an isomorphism of the graphs if and only if*

$$\eta_G(x_1, \ldots, x_n) = \eta_H(x_{\varphi(1)}, \ldots, x_{\varphi(n)}). \tag{2}$$

for all $x \in \mathbb{R}^n$.

The polynomials η_G and η_H are equal if the coefficient of x_c is equal to the coefficient of $x_{\varphi(c)}$ for every subset c of $V(G)$. Thus, the equality (2) holds if and only if $A(G)_c = A(H)_{\varphi(c)}$ for all subsets c of $V(G)$. Let us prove the Theorem 1.

Proof. If $G \simeq H$ and φ is an isomorphism of the graphs, then (2) holds since

$$A(H) = P_\varphi A(G) P_\varphi^\top, \tag{3}$$

and the coefficients of η_G and η_H, corresponded by φ, are equal to each other.

Let us show that if the equality (2) holds, then $G \simeq H$ and φ is an isomorphism of the graphs. Let us denote $A(G)$ as $A = (a_{ij})$ and $B(G)$ as $B = (b_{ij})$.

If (2) holds, then the coefficient of x_c is equal to the coefficient of $x_{\varphi(c)}$ for all subsets c of $V(G)$. Thus, if we take c such that $c = V(G) \setminus \{i, j\}$ for some pair of vertices $i, j \in V(G)$, we have $A_c = B_{\varphi(c)}$. This is equivalent to $\det \begin{pmatrix} 0 & a_{ij} \\ a_{ij} & 0 \end{pmatrix} = \det \begin{pmatrix} 0 & b_{\varphi(i)\varphi(j)} \\ b_{\varphi(i)\varphi(j)} & 0 \end{pmatrix}$. So $a_{ij} = b_{\varphi(i)\varphi(j)}$ and $(i, j) \in E(G)$ if and only if $(\varphi(i), \varphi(j)) \in V(H)$, i.e., $G \simeq H$ and φ is an isomorphism of the graphs. ■

Remark 1. To check the equality (2) for some φ, it suffice to check equality of the coefficients that correspond to c such that $|c| = n-2$. Since if $A_c = B_{\varphi(c)}$ for such c, then $A(H)$ may be obtained from $A(G)$ by permutation of its rows and columns. And, since the equality (3) holds, then $A_c = B_{\varphi(c)}$ for all subsets c of $V(G)$.

3 The algorithms for *GI* that check equality of the values of the modified characteristic polynomials at the specified points

To check wether the two given graphs G and H are isomorphic, we try to find φ such that the equality (2) holds. If such φ exists, then the graphs are isomorphic, else they are not isomorphic.

The equality checking of coefficients of two polynomials of the form (1) may be performed as it is presented in [12]. But if we check equality of the polynomials this way, we need exponential time since $\eta_G(x_1, \ldots, x_n)$ has 2^n coefficients. Such algorithm has an exponential complexity no matter what instance of GI it solves. The algorithms we present below solve GI by comparing of the values that the modified characteristic polynomials take on the specified points of $\mathbb{R}^n$. This approach make possible to significantly reduce the time that is needed to solve GI instance checking equality of the modified characteristic polynomials of graphs. We present two heuristic algorithms to implement the approach: the Direct algorithm for GI and its recursive modification.

3.1 The Direct algorithm for *GI*

Let $N \in \mathbb{N}$, $S = \{k/10^N \mid 0 < k < 10^N,\ k \in \mathbb{Z}_+\}$, $S \subset (0,1)$. For $i = \overline{1,n}$, let ε_i be selected at random independently and uniformly from S. Let $\varepsilon^{(i)}$ be the following points of $\mathbb{R}^n$: $\varepsilon^{(0)} := 0$, $\varepsilon^{(i)} := \varepsilon^{(i-1)} + \varepsilon_i e_i$, $i = 1, \ldots, n$. I.e., $\varepsilon^{(1)} = (\varepsilon_1, 0, \ldots, 0)$, $\varepsilon^{(2)} = (\varepsilon_1, \varepsilon_2, \ldots, 0)$, $\ldots$, $\varepsilon^{(n)} = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)$.

In the course of implemetation of the algorithms we present below, we trying to set a bijection $\varphi : V(G) \to V(H)$ such that $\eta_G(\varepsilon^{(i)}) = \eta_H(\varepsilon_\varphi^{(i)})$. On the i-th iteration of the Direct algorithm, we searching for such $j \in V(H)$ that

$$\eta_G(\varepsilon^{(i)}) = \eta_H(\varepsilon_\varphi^{(i-1)} + \varepsilon_i e_j). \tag{4}$$

If we can set such φ for the graphs G and H during iterations of the algorithm, we make a conclusion that the graphs are isomorphic and φ is an isomorphism of the graphs, else we make a conclusion that they are not isomorphic.

The Direct algorithm for GI is named below as Algorithm 1. It is a test for isomorphism that may be mistaken for some instances of GI. For any test for isomorphism, there are two kinds of mistakes it can make: 1) wrong conclusion that $G \simeq H$, when $G \not\simeq H$, 2) wrong conclusion that $G \not\simeq H$, when $G \simeq H$. As it shall be stated below, the probability of a mistake of the first kind may be considered as negligible for the Direct algorithm. The algorithm solves the GI instances that presented in [16] but it makes a mistake of the second kind for GI instances obtained for strongly-regular graphs from [17] when $n \geq 13$.

Algorithm 1 (G, H)

```
1  J ← V(H);
2  for i ← 1 to n
3        choose at random ε_i ∈ S;
4        if (∃j ∈ J : η_G(ε^(i)) = η_H(ε_φ^(i−1) + ε_i e_j))
5              φ(i) ← j;
6              J ← J \ {j};
7        else print "G ≄ H".
8  print "G ≃ H, φ is an isomorphism of G and H".
```

3.2 Recursive modification of the Direct algorithm for *GI*

Algorithm 2 we present below is a recursive modification of the Direct algorithm for GI. In addition to the GI instances that presented in [16], in a reasonable time, it solves the GI instances obtained for strongly-regular graphs from [17] ($n \leq 64$).

SET THE CORRESPONDENCE ($i \in V(G)$)

```
1  if i=n
2       flag←true;
3       φ(n)←k, where k such that J = {k};
4       exit.
5  else
6       for j←1 to n
7            choose at random ε_i ∈ S;
8            if (j ∈ J and η_G(ε^(i)) = η_H(ε_φ^(i−1) + ε_i e_j))
9                 φ(i)←j;
10                J ← J\{j};
11                SET THE CORRESPONDENCE (i + 1);
12                if flag = false
13                     J←J∪{j};
14                     φ(i) is not defined;
15 exit.
```

ALGORITHM 3 (G, H)

```
1  J←V(H);
2  flag←false;
3  ∀i∈V(G) : φ(i) is not defined.
4  SET THE CORRESPONDENCE(1);
5  if flag
6       print "G≃H, φ is an isomorphism of G and H";
7  else
8       print "G≄H".
```

The recursive procedure SET THE CORRESPONDENCE gets on input $i \in V(G)$ and set $\varphi(i) \leftarrow j$ for $j \in J \subseteq V(H)$ such that (4) holds. If there is no such $j \in J$, then we modify the correspondence that was already setted up for $i-1$: we use the next element of J for setting $\varphi(i-1)$.

The probability of mistake. Let $\mathsf{P}[\,\cdot\,]$ denote a probability of the event that we specify in square brackets. The following theorem [18, **?**] is known:

Theorem 2. *Let $f \in F[x_1, \ldots, x_n]$ be a non-zero polynomial of total degree $d \geq 0$ over a field F. Let S be a finite subset of F and let $\varepsilon_1, \ldots, \varepsilon_n$ be selected at random independently and uniformly from S. Then*

$$\mathsf{P}[f(\varepsilon_1, \ldots, \varepsilon_n) = 0] \leq \frac{d}{|S|}.$$

If we have set up φ for every $i = 1, \ldots, n$, then $\eta_G(\varepsilon_1, \ldots, \varepsilon_n) = \eta_H(\varepsilon_{\varphi(1)}, \ldots, \varepsilon_{\varphi(n)})$. Let

$$f(\varepsilon_1, \ldots, \varepsilon_n) = \eta_G(\varepsilon_1, \ldots, \varepsilon_n) - \eta_H(\varepsilon_{\varphi(1)}, \ldots, \varepsilon_{\varphi(n)}).$$

Total degree d of the polynomial f is equal to n, and $F=\mathbb{R}$ in this case. So, implementing ALGORITHM 1 or ALGORITHM 2, if we obtain a message that $G\simeq H$ and φ is an isomorphism of the graphs, we have $\eta_G\neq\eta_H$ with the probability $\mathsf{P}[mistake]\leq n/10^N$. If we set $N=n$, then $\mathsf{P}[mistake]\leq 1/10^{n-\lg n}$ and the message is correct with probability no less than $1–1/10^{n-\lg n}$. For the Direct algorithm, the message that the graphs are not isomorphic may be incorrect. The message is always correct for its recursive modification since in this case there is no such φ that (4) holds successively for $i=1,\ldots,n$. By the Theorem 1, it follows that the graphs are not isomorphic.

4 The numerical realization of the algorithms and their computational complexity

Modifications of adjacency matrices. Let d be the maximum degree of vertices of G and H: $d=\max\{d_1,\ldots,d_n\}$ (we suppose that G and H have the same degree sequences). Let A and B be modified graph adjacency matrices of the following form

$$A := A(G) + 2dE,\ B := A(H) + 2dE. \tag{5}$$

$A(H)=P_\varphi A(G)P_\varphi^\top$ for some bijection φ if and only if $A=P_\varphi BP_\varphi^\top$. If $d>0$, then A and B have the strong diagonal predominance:

$$(A)_{ii} = 2d \geq 2d_i > d_i = \sum_{\substack{j=1\\ j\neq i}}^{n}(A)_{ij},$$

so

$$(A)_{ii} - \sum_{\substack{j=1\\ j\neq i}}^{n}(A)_{ij} > 0,$$

$i=\overline{1,n}$. Thus, the matrices of the form (5) satisfy the Hadamard conditions, and we have $\det A\neq 0$, $\det B\neq 0$ [20]. For numerical realization of the algorithms that we have presented, we use modified characteristic polynomials of the following form:

$$\eta_G(x_1,\ldots,x_n) = \det(A+X),\ \eta_H(x_{\varphi(1)},\ldots,x_{\varphi(n)}) = \det(B+X_\varphi), \tag{6}$$

where $X_\varphi=P_\varphi XP_\varphi^\top$. This modification is equivalent to the change of variables: $x_i\to x_i+2d$.

We may consider the presented algorithms as a try to perform series of consistent modifications of the matrices A and B:

$$A^{(i)} := A^{(i-1)} + \varepsilon_i E_i,\ B^{(i)} := B^{(i-1)} + \varepsilon_i E_j, \tag{7}$$

where $A^{(0)}=A,\ B^{(0)}=B,\ i=1,\ldots,n$. We call the modifications of the form (7) *consistent*, if we successively choose, for every $i\in V(G)$, such $j\in V(H)$ that holds

$$((A^{(i)})^{-1})_{ii} = \frac{A^{(i)}_{\{i\}}}{\det A^{(i)}} = \frac{(B^{(i-1)}+\varepsilon_i E_j)_{\{j\}}}{\det(B^{(i-1)}+\varepsilon_i E_j)} = ((B^{(i-1)}+\varepsilon_i E_j)^{-1})_{jj}, \tag{8}$$

where E_i is $n \times n$-matrix such that all of its elements are zeros except the i-th diagonal element, which is equal to 1. It follows from (8) that

$$\eta_G(\varepsilon^{(i)}) = \eta_H(\varepsilon_\varphi^{(i-1)} + \varepsilon_i e_j) \tag{9}$$

since the equality (8) is equivalent to the equality

$$\frac{\eta_{G\setminus\{i\}}(\varepsilon^{(i)})}{\eta_G(\varepsilon^{(i)})} = \frac{\eta_{H\setminus\{j\}}(\varepsilon_\varphi^{(i-1)} + \varepsilon_i e_j)}{\eta_H(\varepsilon_\varphi^{(i-1)} + \varepsilon_i e_j)}. \tag{10}$$

And, since the values $\eta_{G\setminus\{i\}}(\varepsilon^{(i)})$ and $\eta_{H\setminus\{j\}}(\varepsilon_\varphi^{(i-1)} + \varepsilon_i e_j)$ not change when the value of ε_i is changing, if the equality (10) holds for $\varepsilon_i = 0$ and for some non-zero value of ε_i, then the equality (9) holds too. Thus, if we can perform series of consistent modifications for $i = 1, \ldots, n$, then the values of the polynomials of the graphs are equal at the specified points $\varepsilon^{(i)}$. As a result, we solve GI by using series of consistent modifications of graphs adjacency matrices. This idea belongs to R.T. Faizullin. It was presented in [21].

Accuracy of computations and computational complexity. We compute the values of $((A^{(i)})^{-1})_{ii}$ by solving the system of linear equations $A^{(i)}y = e_i$. The value of $((A^{(i)})^{-1})_{ii}$ is the i-th component of y. In order to solve the systems of linear equations, we may use such iterative methods as the Gauss-Seidel method or the simple iteration method. These methods converge at the rate of geometric progression for any starting vector because the systems matrices have the diagonal predominance. Using the standard numeric type double, we can solve instances of GI that was mentioned above choosing $\varepsilon_i \in [\delta, 1)$, $\delta \geq 0.001$. The computational complexity of the Direct algorithm is equal to $O(n^4 k)$, where k is a number of iterations of the method we use to solve the systems of linear equations. In our computational experiments, we choosed $k = 10$. Recursive modification of the Direct algorithm is exponential in the worst case.

5 Eliminating regularities of graphs and reducing the exhaustive search

The Algorithm 2 scheme differs from the scheme of exhaustive search on all bijections of $V(G)$ onto $V(H)$ only in step 8 of the procedure Set the correspondence that it use. Here, in addition to check whether current $j \in V(H)$ is not already setted up as an image for some vertex in $V(G)$ with label less than i, we check equality of the polynomials values in predefined points $\varepsilon^{(i)}$ and $\varepsilon_\varphi^{(i)}$. The algorithm of the same form may be obtained for numerous graph invariant characteristics. For example, we may check the equality of adjacency matrices of the induced subgraphs on i vertices for both input graphs. These subgraphs include the vertices for which the correspondence is setted up and all of the edges whose endpoints are these vertices. We check the equality of the subgraphs adjacency matrices after enumerating vertices of the induced subgraph of H in according to φ that is setted up for vertices from $V(G)$ with labels less or equal than i.

The graph G has more regular structure if it has more symmetries, i.e., if there are such bijections of $V(G)$ onto $V(G)$ which are isomorphisms of G onto itself (graph automorphisms).

Definition 2. *The graph G automorphism group is a set of isomorphisms of the graphs onto itself, i.e.,*

$$Aut(G)=\{\psi\in S_n \mid a_{ij}=a_{\psi(i)\psi(j)},\ i,j=\overline{1,n}\}.$$

Definition 3. *The orbit of the vertex $i\in V(G)$ is the set*

$$O_i(G)=\{\psi(i) \mid \psi\in Aut(G)\}.$$

The considered scheme of the algorithm for GI (exhaustive search on all bijections that is reduced by equlity checking of some invariant characteristics for the input graphs) may be modified to be efficient for some restricted classes of graphs, or, using some invariant characteristics, we may obtain some partitions of the graphs vertices that we may use to reduce the exhaustive search, but the main problem stays the same for every algorithm for GI of that sort: the more cardinalities of $Aut(G)$ and $Aut(H)$ and the weaker the graph invariant (i.e., it's more easy to find two non-isomorphic graphs for which the values of the invariant are the same) that we use to check at step 8 of the procedure SET THE CORRESPONDENCE, the more alternatives for setting of φ we shall obtain in the course of its implementation, and it is more hard to find among them such a bijection that is an isomorphism or to find out that there is no isomorphism for input graphs. Conversely, the graphs with less regular structure, such as, e.g., random graphs, give GI instances that are easy to solve using known algorithms since they have automorphism groups of low cardinalities.

The following lemma shows that, solving GI for the graphs G and H, we may have alternative variants for setting isomorphism of the graphs only if we have such $i\in V(G)$ that $|O_i(G)|>1$.

Lemma 1. *Let $G\simeq H$, $i_1,i_2\in V(H)$. $i_1\in O_{i_2}(H)$ if and only if there exists a vertex $j\in V(G)$ and such isomorphisms $\varphi_1,\varphi_2 : V(G)\to V(H)$ that $\varphi_1(j)=i_1$, $\varphi_2(j)=i_2$.*

Proof. Let $i_1\in O_{i_2}(H)$. It follows that there is $\psi\in Aut(G)$ such that $\psi(i_1)=i_2$. Let φ_1 be an isomorphism of G onto H and let $j=\varphi_1^{-1}(i_1)$, $j\in V(H)$. Let $\varphi_2=\psi\circ\varphi_1$. We have $\varphi_2(j)=(\psi\circ\varphi_1)(j)=\psi(i_1)=i_2$ and φ_2 is an isomorphism of G onto H. Conversely, suppose there are such $i_1,i_2\in V(H)$, $j\in V(G)$ and isomorphisms φ_1, φ_2 that $\varphi_1(j)=i_1$, $\varphi_2(j)=i_2$. Then $\psi=\varphi_2\circ\varphi_1^{-1}\in Aut(H)$ and $\psi(j)=i_2$, i.e., $i_1\in O_{i_2}(H)$. ■

In the course of the transformations (7), we subsequently obtain the graphs with less regular structure than the input graphs have. The matrices $A^{(i)}$ and $B^{(i)}$ in (7) may be considered as adjacency matrices of the graphs $G^{(i)}$ and $H^{(i)}$. These graphs has weighted loops, i.e., edges of the form $(j,j)\in E(G)$. After the i-th iteration of the algorithm, we have $\psi(i)\neq j$ for all $\psi\in Aut(G^{(i)})$ since the i-th and the j-th diagonal elements of $A(G^{(i)})$ are not equal to each other with probability that negligibly close to 1 for all $j\in V(G)$, $i\neq j$. So we have $|O_i(G^{(i)})|=1$. And, in accordance with the Lemma above, we have no more than one way to set the value of $\varphi(i)$ on the i-th iteration of the algorithm. Performing transformations (7) and selecting unique values of the loops weights ε_i, we subsequently obtain such $G^{(i)}$ and $H^{(i)}$ that

$$|O_j(G^{(i)})|=1, j\leq i,\quad |Aut(G^{(i+1)})|\leq|Aut(G^{(i-1)})|,$$

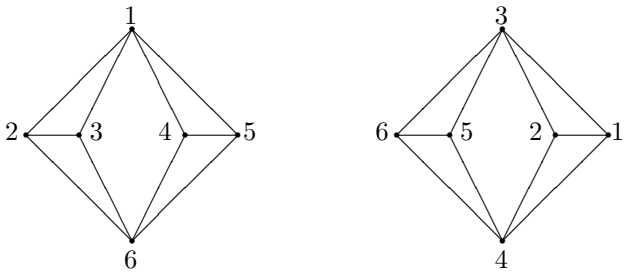
$$|O_{\varphi(j)}(H^{(i)})| = 1, j \leq i, \quad |Aut(H^{(i)})| \leq |Aut(H^{(i-1)})|$$

and finally we get

$$|O_j(G^{(t)})| = |O_{\varphi(j)}(H^{(t)})| = 1, j = \overline{1,n}, \quad |Aut(G^{(t)})| = |Aut(H^{(t)})| = 1$$

for $t \leq n-1$.

Let us consider an example of the algorithm operating that illustrates this reasoning. Let G and H be the input graphs shown on a picture below.



Reduction of the number of variants to set φ is shown in table 1. After the 4-th iteration, we have $|Aut(G^{(4)})| = 1$. In the table 2, the values of $((A^{(i)})^{-1})_{jj}$ are shown. For all $i = \overline{1,n}$, if the value of $\varphi(i)$ is setted up, then the multisets $\{((B^{(i)})^{-1})_{jj}\}_{j=1}^{n}$ and $\{((A^{(i)})^{-1})_{jj}\}_{j=1}^{n}$ are equal. To compute these values, we perform 10 iterations of the Gauss-Seidel method in order to solve the systems of equations of the form $A^{(i)}y = e_i$. The initial approximation that we use is $y^{(0)} = (1, \ldots, 1)$.

Table 1. Reduction of the alternative variants of setting φ for G and H.

i	Variants of setting φ						$\lvert Aut(G^{(i)})\rvert$
	1	2	3	4	5	6	
0	3, 4	1, 2, 5, 6	1, 2, 5, 6	1, 2, 5, 6	1, 2, 5, 6	3, 4	16
1	3	1, 2, 5, 6	1, 2, 5, 6	1, 2, 5, 6	1, 2, 5, 6	4	8
2	3	1	2	5, 6	5, 6	4	2
3	3	1	2	5, 6	5, 6	4	2
4	3	1	2	5	6	4	1

6 Conclusions

In our work, we assign modified characteristic polynomials for graphs and reduce the graph isomorphism problem to the following one. It is required to find out, is there such an enumeration of the graphs vertices that the modified characteristic polynomials of

Table 2. Computed values of $((A^{(i)})^{-1})_{jj}$, $i=\overline{1,3}$.

i	ε_i	$((A^{(i)})^{-1})_{11}$	$((A^{(i)})^{-1})_{22}$	$((A^{(i)})^{-1})_{33}$	$((A^{(i)})^{-1})_{44}$	$((A^{(i)})^{-1})_{55}$	$((A^{(i)})^{-1})_{66}$
0	0	0.078	0.094	0.094	0.094	0.094	0.078
1	0.861	0.070	0.095	0.095	0.095	0.095	0.078
2	0.672	0.070	0.087	0.095	0.095	0.095	0.079
3	0.372	0.071	0.087	0.091	0.094	0.094	0.079
4	0.475	0.072	0.087	0.091	0.089	0.095	0.080

the graphs are equal. The modified characteristic polynomial of a graph on n vertices is a polynomial of n variables. We prove that two graphs are isomorphic if and only if there is exists an enumeration of the graphs vertices such that the polynomials of the graphs are equal. We present algorithms for the graph isomorphism problem that use the redution. The algorithms check equality of the graphs polynomials values at specified points without direct evaluation of the values. We give justification of the algorithms and give the scheme of its numerical realization. The example of the approach implementation and the results of computational experiments are presented.

References

1. Lindell, S.: A logspace algorithm for tree canonization. Proc. of the 24th Annual ACM Symposium on the Theory of Computing. 400–404 (1992)
2. Filotti, I.S., Mayer, J.N.: A polynomial-time algorithm for determining the isomorphism of graphs of fixed genus. Proc. of the 12th Annual ACM Symposium on Theory of Computing. 236–243 (1980)
3. Babai, L., Grigoryev, D.Yu., Mount, D.M.: Isomorphism of graphs with bounded eigenvalue multiplicity. Proc. of the 14-th Annual ACM Symposium on Theory of Computing. 310-324 (1982)
4. Luks, E.M.: Isomorphism of graphs of bounded valence can be tested in polynomial time. Journal of Computer and System Sciences. 25. 42-65 (1982)
5. Ullman, J.R.: An algorithm for subgraph isomorphism. Journal of the ACM. 23. 31–42 (1976)
6. Schmidt, D.C., Druffel, L.E.: A fast backtracking algorithm to test directed graphs for isomorphism using distance matrices. Journal of the ACM. 23(3). 433–445 (1978)
7. McKay, B.D.: Practical graph isomorphism. Congr. Numer. 30. 45–87 (1981)
8. Babai, L.: Graph Isomorphism in Quasipolynomial Time. https://arxiv.org/abs/1512.03547
9. Cai, J-Y., Fürer, M., Immerman, N.: An optimal lower bound on the number of variables for graph identification. Combinatorica. 12 (4). 389–410 (1992)
10. Datta, S., Limaye, N., Nimbhorkar, P., Thierauf, T., Wagner, F.: Planar graph isomorphism is in log-space. 24th Annual IEEE Conference on Computational Complexity. Paris, France. 203–214 (2009)
11. Grigoriev, D.Yu.: Two Reductions of graph isomorphism problem for polynomials. J. Soviet Math. 20. 2296–2298 (1982) (in Russian)
12. Prolubnikov, A.V.: Reduction of the graph isomorphism problem to equality checking of polynomials of n-variables. Trudy Instituta Matematiki i Mekhaniki, Ekaterinburg: IMM of Ural Department of Russian Academy of Sciences. 22(1). 235–240 (2016) (In Russ.)

13. Cvetkovic, D.M, Doob, M., Sachs, H.: Spectra of Graphs. Academic Press, New York (1980)
14. Seidel, J.J.: Strongly regular graphs with (-1,1,0) adjacency matrix having eigenvalue 3. Lin. Alg. Appl. 1(2). 281–298 (1968)
15. Lipton, R.J., Vishnoi, N.K., Zalcstein, Z.: A generalization of the characteristic polynomial of a graph. CC Technical Report, GIT-CC-03-51. 2003. https://smartech.gatech.edu/bitstream/handle/1853/6511/GIT-CC-03-51.pdf
16. Foggia, P., Sansone, C., Vento, M.: A database of graphs for isomorphism and sub-graph isomorphism benchmarking. Proc. of the 3rd IAPR TC-15 international workshop on graph-based representations. 157–168 (2001)
17. Strongly regular graphs on at most 64 vertices. http://www.maths.gla.ac.uk/∼es/srgraphs.php
18. Schwartz, J.: Fast probabilistic algorithms for verification of polynomial identities. Journal of the ACM. 27. 701–717 (1980)
19. Zippel, R.: Probabilistic algorithms for sparse polynomials. Proc. of the International Symposium on Symbolic and Algebraic Computation. 216–226 (1979)
20. Gantmaher, F.R.: Teoria matriz. Moskva: Izd-vo Nauka (1967) (in Russian)
21. Faizullin, R.T., Prolubnikov, A.V.: An algorithm of the spectral splitting for the double permutation cipher. Pattern Recognition and Image Analysis. 12(4). 365–375 (2002)