Mathematical
Structures and Modeling
2015. N. 4(36). PP. 53–66

UDC 510.52

SEARCH PROBLEMS WITH A PROMISE AND GRAPH ISOMORPHISM

G.A. Noskov
Dr.Sc.(Phys.-Math.), S.R., e-mail: g.noskov@googlemail.com

Institute of Mathematics, SORAN

Abstract. The Graph isomorphism problem is considered from the point of view of the theory of "problems with a promise", developed by Even, Selman and Yacobi [4]. The "tau-invariant" of graphs is studied and with its help the search graph isomorphism problem is solved for asymptotically almost all graphs.

Keywords: decision problem, search problem with a promise, graph isomorphism.

Introduction

In this paper we consider the search graph isomorphism problem SGI in the context of "search problems with a promise" as they defined in [4, 6, 7].

In existing literature (see for instance [9]) the decision and search variants of the graph isomorphism problem are formulated as follows:

P1: Given two graphs G and H with n vertices each, decide whether they are isomorphic.

P2: Given two graphs G and H, decide whether they are isomorphic, and if so, construct an isomorphism from G to H.

Note that P2 contains P1 as a subproblem. Recently, A.N. Rybalov suggested to isolate the search version from the decision version as follows [12]. By definition, the input set for search graph isomorphism problem SGI is the set of all pairs of isomorphic graphs. Given two isomorphic graphs G and H, one needs to construct the isomorphism between G and H. Thus in Rybalov's setup of the SGI the input graphs are already assumed to be isomorphic, whereas in version P2 above the input graphs are arbitrary. The seemingly unusual input set in SGI can be elegantly explained in the framework of "promise problems" in the sense of [4, 6]. Promise problems are a natural generalization of search and decision problems, where one explicitly considers a set of legitimate instances (rather than considering any string as a legitimate instance). Informally, a promise decision problem has the following structure: input x, promise $P(x)$, property $R(x)$, where P and R are unary predicates. An algorithm solves the promise problem if, given an input x, it answers the question whether $R(x)$ given that $P(x)$. The behavior of such an algorithm may be arbitrary on instances x for which the promise P is false [10].

In this paper we formulate the search graph isomorphism problems "with a promise" and study their reducibility and generic solvability in polynomial time. We make use the "type-invariant" of a graph introduced in [2, 13, 14]. Our main result is Theorem 5, which presents the polynomial-time algorithm A, solving the problem $SGI\,(P_{isom}, R)$ for asymptotically almost all inputs $P_{isom} \cap P_{oblique}$.

In Section 1 we give the necessary definitions about computation problems. Section 2 is devoted to the formulation of the search graph isomorphism problem (with a promise). Section 3 contains definitions of computational problems in the case of graphs. Section 4 contains the proof of polynomial reducibility of search graph isomorphism problem SGI to the graph isomorphism problem. In Section 5 we discuss the main tool - the graph invariant τ and oblique graphs. In Section 6 the efficient solvability of our problems in case of oblique graphs. In Section 7 we solve the SGI problem in generic case. Finally, in the last section we apply our results to probabilistic algorithms.

1. Decision and search with a promise

We start with the standard definitions of decision and search computational problems as they presented in [11] and [7]. Let $I = \{0,1\}^*$ be the set of all **words (=binary strings)** in the alphabet $\{0,1\}$. We consider algorithms as means of computing functions. Specifically, an algorithm A computes the function $f_A : \{0,1\}^* \to \{0,1\}^*$ defined by $f_A(x) = y$ if, when invoked on input x, algorithm A halts with output y. We associate the algorithm A with the function f_A computed by it; that is, we write $A(x)$ instead of $f_A(x)$.

Decision problems. Let's denote by I^k the k-th direct power $I \times \cdots \times I$ of I for $k \geqslant 1$. A **decision problem** for a subset **(=language)** $L \subseteq I^k$ is to determine for a given tuple $w \in I^k$ whether w belongs to L or not. An algorithm A solving this problem is the **decision algorithm** for L, and in this case the decision problem for L, as well as the language L, is called **decidable**.

If L is decidable and additionally there are positive constants c, q such that for every instance $x \in I^k$ the algorithm A determines the membership of x to L in at most $c\,|x|^q$ steps, then the decision problem for L, as well as L is called **polynomial-time decidable** (or **decidable in polynomial time**). Here, for a tuple $x = (x_1, ..., x_k)$ we let denote by $|x|$ the maximum $\max |x_i|$ of lengths of words x_i.

Decision problems with a promise. More general class of partial decision problems was introduced in [4] under the name of "promise problems". Formally, a **partial decision problem** is a pair of decidable subsets (L, P) of I^k, where P is the set of **allowed** or **promised** tuples and $\overline{P} = I^k - P$ is the set of **disallowed** tuples. The promise problem (L, P) is **solved by algorithm** A if for every $x \in P \cap L$ it holds that $A(x) = 1$ and for every $x \in P - L$ it holds that $A(x) = 0$. Shortly,

$$A\,(x) = \begin{pmatrix} 1 & \text{if } x \in P \cap L, \\ 0 & \text{if } x \in P - L \end{pmatrix}. \tag{1}$$

Thus, the algorithm A is required to distinguish **yes-instances** $P \cap L$ from **no-**

instances $P - L$ and A is allowed to have arbitrary behavior on inputs $\overline{P}$ that are neither yes-instances nor no-instances. A set L_1 is called a **solution** to the partial decision problem (L, P) if $L_1 \cap P = L$. Clearly, when $P = I^k$ we have the notion of a standard decision problem. A partial decision problem (L, P) is **polynomial-time decidable** if there is a polynomial-time decidable solution L_1 of it.

Remark 1. In [7, 8] a more restrictive definition is used, namely the promise problems (L, P) are considered with the condition $L \subseteq P$. It is not enough for our purposes, see Section 5.

Search problems. A **search computational** problem can be described by a binary relation $R \subseteq I^k \times I^l$ for some fixed $k, l \geqslant 1$. The problem is "given an input $x \in I^k$, find y such that $R(x, y)$ holds, if such y exists". More precisely, one requires, for a given $x \in I^k$ to decide first whether there exists $y \in I^l$ such that $R(x, y)$ holds, and only after that to find such y if it exists.

Let us consider R as a multi-valued function $R(x) = \{y : (x, y) \in R\}$. The associated **solution set** is $S_R = \{x : R(x) \neq \emptyset\}$. A function $f : I^k \to I^l \cup \{\perp\}$ is called a **branch** of $R(x)$ if $f(x) \in R(x)$ for all $x \in S_R$ and $f(x) = \perp$ for all $x \notin S_R$ (thus $f(x) = \perp$ indicates that x has no solution). We say that the branch function f of R **solves** the search problem of R. As before, we write $A(x)$ for $f(x)$ for an algorithm A, computing the function f.

Note, that the search problem for R contains the decision problem for a **solution set** $S_R = \{x : R(x) \neq \emptyset\}$ as a subproblem. Indeed, if A computes the branch of R, then $x \in S_R$ iff $A(x) \neq \perp$.

A relation $R \subseteq I^k \times I^l$ is **polynomially bounded** if there exists a polynomial p such that for every $(x, y) \in R$ it holds that $|y| \leqslant p(|x|)$. The search problem of a polynomially bounded relation $R \subseteq I^k \times I^l$ is **efficiently solvable** if there exists a branch function f of R which is polynomial-time computable. In this case an algorithm A, computing this function, is called a **polynomial-time algorithm** for the search problem R. We denote by $\mathcal{PF}$ the class of polynomially bounded search problems that are efficiently solvable [8].

Search problems with a promise [8, p.143]. A **search problem with a promise** consists of the input set I^k, a binary relation $R \subseteq I^k \times I^l$ and a **promise set** $P \subseteq I^k$. Such a problem is also referred to as the **search problem** R **with promise** P and is denoted by (P, R). The search problem (P, R) with promise P is **solved** by algorithm A if for every $x \in P$ it holds that $(x, A(x)) \in R$ if $x \in S_R$ and $A(x) = \perp$ otherwise. Thus, the restriction function $A|P$ is the branch of the relation $R \cap \left(P, I^l\right)$. "We stress that nothing is required of the solver in the case that the input violates the promise (i.e., $x \notin P$); in particular, in such a case the algorithm may halt with a wrong output" [8, p.143]. Note that the **full** search problem $\left(I^k, R\right)$ with a promise I^k is the standard search problem. And at the other extreme there is the so called **candid search problem** (S_R, R) with promise S_R.

The **time complexity of** A **on inputs in** P is defined by

$$T_{A|P}(n) = \max\{t_A(x) : x \in P \cap \{0,1\}^n\}, \tag{2}$$

where $t_A(x)$ is the running time of $A(x)$. A problem (P, R) is **polynomially-time decidable** if there is an algorithm A such that function $T_{A|P}$ is bounded by a polynomial in n. "In this case, it does not matter whether the time complexity of A is defined on inputs in P or on all possible strings. Suppose that A has (polynomial) time complexity T on inputs in P; then we can modify A to halt on any input x after at most $T(|x|)$ steps. This modification may only affect the output of A on inputs not in P (which are inputs that do not matter anyhow). The modification can be implemented in polynomial time by computing $t = T(|x|)$ and emulating the execution of $A(x)$ for t steps." [7, p.88].

Algorithms which are polynomial-time on asymptotically almost all inputs. By the very definition of the search problem (P, R), it becomes easier when decreasing the promise P. If the given problem (P, R) is hard, then it makes sense to consider a **restriction** problem (P', R) with a promise set $P' \subset P$ and ask whether (P', R) is polynomial-time decidable. This may happen when P' is small enough, for instance when P' is finite. Thus, it is highly desirable to find out a polynomial-time decidable restriction (P', R) with P' being maximally close to P. For a rigorous definition of closedness we need a **size function** on the promise set P, i.e. any computable function $x \mapsto \|x\| \in \mathbb{N}$ such that for every $n \in \mathbb{N}$ the set $P_n = \{x \in P : \|x\| = n\}$ is finite. Furthermore, we need an **ensemble of distributions** $\mathcal{D} = (D_n)$ with D_n a distribution on the set P_n [1]. The closedness of P' to P can be measured by the **asymptotic density (=asymptotic probability)** (if exists)

$$D(P') = \lim_{n\to\infty} D_n(P' \cap P_n). \tag{3}$$

A subset $P' \subseteq P$ is said to be **asymptotically almost certain (=generic)** if $D(P') = 1$. In the theory of random graphs another terminology is accepted: the event $P' \subseteq P$ as above **happens asymptotically almost surely (=a.a.s)** or **with high probability (=w.h.p.)**. The complement of a generic set is said to be **negligible.**

A search problem (P, R) is **polynomial-time decidable with high probability (=for asymptotically almost all** inputs) if it admits a polynomial-time decidable restriction (P', R) such that P' is asymptotically almost certain in P (relative to a fixed size function and fixed distribution ensemble. Similar definitions can be given in case of promise decision problems.

2. Graph problems

Let $\mathcal{G}_n$ denote the set of all graphs on the set of vertices $[1; n] = \{1, \ldots, n\}$. The set $\mathcal{G}_n$ is naturally acted upon by the symmetric group S_n on the set $[1; n]$. The orbits of this action are precisely the isomorphism classes of graphs from $\mathcal{G}_n$.

We encode graphs as binary strings as usual. Namely, let (g_{ij}) be the adjacency matrix of G then we encode G by the string $(g_{12}g_{13}\cdots g_{1n}g_{23}\cdots g_{2n}\cdots g_{n-1,n})$. This encodes $\mathcal{G}_n$ bijectively onto $I_{\frac{n(n-1)}{2}}$. The permutations $\phi \in S_n$ are in one-one correspondence with $n \times n$ monomial matrices and the last ones can be encoded row by row by binary strings in I_{n^2}. We consider the **graph isomorphism problem**

$GI(L,P)$ as a partial decision problem (L,P) whose input set is $I\times I$, whose promise set is a decidable subset $P\subseteq P_{all}$, where

$$P_{all}=\cup_n\left(\mathcal{G}_n\times\mathcal{G}_n\right)=\cup_n\left(I_{\frac{n(n-1)}{2}}\times I_{\frac{n(n-1)}{2}}\right) \tag{4}$$

and whose language L of yes-instances is defined by

$$L=P_{isom}\overset{def}{=}\left\{(G,H)\in P_{all}:\ G\simeq H\right\}. \tag{5}$$

The **standard** GI-problem is (P_{isom},P_{all}). It is to determine for a given tuple $(G,H)\in I^2$ whether $G\simeq H$ or not.

In view of exceptional hardness of (P_{isom},P_{all}) the problems (P_{isom},P) have been considered with P being the pairs of graphs from some interesting classes of graphs, such as trees, planar graphs, graphs of bounded valence, etc. For instance, it is well known, that promise decision problem (P_{isom},P_{trees}), where P_{trees} is the class of all finite trees, is polynomial-time decidable.

Accompanying to GI is SGI - the **search graph isomorphism problem with a promise**. This problem is denoted by $SGI\left(P,R\right)$ and consists of the input set I^2, relation $R\subseteq I^2\times I$ and a promise set $P\subseteq I^2$. The relation R is defined by

$$R=\left\{(G,H,\phi):\exists n\text{ such that }G,H\ \in\mathcal{G}_n,\phi\in S_n\subseteq I_{n^2}\text{ and }\phi:G\simeq H\right\}. \tag{6}$$

The promise set P is an arbitrary decidable subset of P_{all}. The problem requires, for a given pair of graphs $(G,H)\in P$ to decide first whether they are isomorphic and then to find an isomorphism $\phi:G\simeq H$.

The problem $SGI\left(P,R\right)$ with promise P is **solved** by algorithm A if for every $(G,H)\in P$ it holds that $((G,H),A(G,H))\in R$ if $G\simeq H$ and $A(G,H)=\perp$ otherwise. We see, that the **full** search problem (P_{all},R) with a promise P_{all} is the standard SGI-problem as it is defined for instance in [9]. This problem contains the standard decision problem $GI\left(P_{isom},P_{all}\right)$ as a subproblem. Indeed, if an algorithm A solves $SGI\left(P_{all},R\right)$ then $(G,H)\in P_{isom}$ if and only if $A(G,H)=\perp$, thus A recognizes whether an isomorphism between G and H exists or not.

At the other extreme there is the candid search problem (S_R,R) with promise S_R, where $S_R=\{x:R\left(x\right)\neq\emptyset\}$ is the solution set of R . In our case it is clear that

$$S_R=P_{isom}=\left\{(G,H)\in I^2:\ G\simeq H\right\}. \tag{7}$$

And again, the full search problem $SGI\left(P_{all},R\right)$ contains the candid search problem $SGI\left(S_R,R\right)$ as a subproblem. We consider $SGI\left(S_R,R\right)$ as an adequate formulation (in terms of promise problems) for the problem SGI from [12]. We find the following problem rather intriguing:

Problem 1. *Is the standard isomorphism problem* $GI\left(P_{isom},P_{all}\right)$ *polynomial-time reducible to* $SGI\left(S_R,R\right)$*?*

3. The problem $SGI\,(P_{all}, R)$ is polynomial-time reducible to $GI\,(P_{isom}, P_{all})$

Theorem 1. *The full search isomorphism problem $SGI\,(P_{all}, R)$ is polynomial-time reducible to the standard isomorphism problem $GI\,(P_{isom}, P_{all})$.*

Proof. (Cf. [9, Thm.6, p.29]) It is easy to see that SGI can be reduced to the case of graphs without isolated vertices.

Ascent. Suppose we have a polynomial-time algorithm A recognizing the graph isomorphism. Given two isomorphic graphs G, H on n vertices and the list of vertices $V_G = \{g_1, \ldots, g_n\}$, we will construct inductively and efficiently (=in polynomial time) the sets of graphs $G = G_0 < G_1 < \cdots < G_n$, $H = H_0 < H_1 < \cdots < H_n$ and the vertices $h_1, \ldots, h_n \in V_H$ such that:

1) $G_i \simeq H_i$ for all i,

2) Every isomorphism between G_i and H_i takes G_{i-1} to H_{i-1} for all $i = 1, \ldots, n$,

3) Every isomorphism between G_i and H_i takes g_i to h_i for all $i = 1, \ldots, n$.

The construction is defined as follows. Let K_{n+1} denote the complete graph on $n+1$ vertices. Fix $k_1 \in K_{n+1}$ and consider

$$G_1 = G \cup_{g_1 = k_1} K_{n+1}, \quad H_1(h) = H \cup_{h = k_1} K_{n+1},$$

where $\cup_{x=y}$ denote the gluing operation on graphs with identified vertices $x = y$. Since $G \simeq H$, there exists h such that $G_1 \simeq H_1 = H_1(h)$, namely h is the image of g_1 under isomorphism $G \simeq H$. We find out h by applying $\mathcal{A}$ to all n pairs $(G_1, H_1(h))$. Set $h_1 = h$. By assumption, G, H have no isolated vertices, so g_1, h_1 are unique vertices of highest degree in G_1, H_1. Hence every isomorphism between G_1 and H_1 takes g_1 to h_1. Moreover, all vertices of G distinct from g_1 have degree (in G_1) no greater than $n-1$, whereas all the vertices of K_{n+1} have degree (in G_1) at least n. Therefore, every isomorphism between G_1 and H_1 takes G_0 to H_0. Thus 1)-3) hold for $i = 1$.

Fix $k_2 \in K_{n+2}$. For all $h \in V_H$ construct the graphs

$$G_2 = G_1 \cup_{g_2 = k_2} K_{n+2}, \quad H_2(h) = H_1 \cup_{h = k_2} K_{n+2}. \tag{8}$$

Since G_1, H_1 are isomorphic, there exists h such that $G_2 \simeq H_2$ and we can find h applying $\mathcal{A}$ to all n pairs of graphs $G_2, H_2(h)$. Fix any such h and denote it by h_2. Continuing, we find out the desired $h_1, \ldots, h_n \in H$ and $G = G_0 < G_1 < \cdots < G_n$, $H = H_0 < H_1 < \cdots < H_n$.

Descent. The map $g_i \mapsto h_i$ is an isomorphism between G and H! Indeed, by 1) there exists an isomorphism $\phi : G_n \to H_n$. By 2) ϕ takes g_n to h_n. The restriction $\phi|_{G_{n-1}}$ takes G_{n-1} to H_{n-1} by 2) and takes g_{n-1} to h_{n-1} by 3). Then $\phi|_{G_{n-1}}$takes G_{n-2} to H_{n-2} and we can continue this descent process until obtaining that ϕ takes every g_i to h_i. Hence the map $g_i \mapsto h_i$ is an isomorphism in question. ■

4. Graph invariant τ and oblique graphs

Graph invariant τ [2, 13, 14].

Let $\mathbb{N}$ denote the set of all natural numbers (taken as 1,2,3,4,. . .). A **string** over $\mathbb{N}$ is a finite sequence of natural numbers, that is, an integer $n > 0$ and a mapping $\{1, ..., n\} \to \mathbb{N}$. If $n = 0$, the domain is the nullset and there is a unique such mapping, called the **nullstring** and generally denoted ε [3]. Let $\mathbb{N}^*$ denote the set of all finite strings of natural numbers (including the nullstring ε). The **lexicographic order** ranks strings of the same length in $\mathbb{N}^*$, by comparing the letters in the first position where the strings differ. We define the **ShortLex order** on $\mathbb{N}^*$: $v < w$ if and only if v is shorter than w, or they have the same length and v comes before w in lexicographical order. ShortLex order is a well-ordering.

Let $G = (V, E)$ be a simple graph. By $d(x)$ we denote the degree of the vertex x. The set of all vertices adjacent to v is denoted by $N(v)$.

The **type-string** $\tau_G(v) \in \mathbb{N}^*$ of a vertex v is the string of degrees of vertices of $N(v)$ in non-decreasing order. We set $\tau_G(v) = \varepsilon$ for isolated vertex v. In detail, $\tau_G(v) = (d_1, ..., d_{d(v)})$ is the degree sequence of the vertices adjacent to v, arranged in non-decreasing order: $d_1 \leqslant d_2 \leqslant \ldots \leqslant d_{d(v)}$. Clearly a type-string is preserved under graph isomorphisms: if $\phi : G \to H$ is a graph isomorphism then $\tau_H(\phi(v)) = \tau_G(v)$ for all $v \in V(G)$, i.e. $\tau_H \circ \phi = \tau_G$.

The **type-vector** τ_G of G is the ShortLex ordered sequence $(\tau_G(v_1), \ldots, \tau_G(v_n))$, $n = |V|$ of all type-strings of all vertices (thus τ_G is used to denote a vector as well as a map). Clearly the type-vector function $G \mapsto \tau_G$ is a **graph invariant**, i.e. $\tau_G = \tau_H$ for isomorphic graphs G, H. However, as the next example will show, the type-vector is not a complete graph invariant, i.e. there exist non-isomorphic graphs with the same type-vector. The examples will be found among regular graphs. A graph G is said to be k-**regular** if all the vertices of G have the same degree k. In this case $\tau_G = \left(k^{*k}, \ldots, k^{*k}\right)$ has n coordinates, $n = |V_G|$, where $k^{*k} = (k, \ldots, k) \in \mathbb{N}^*$ is a string with k coordinates. Thus all k-regular graphs on n vertices have the same τ-invariant.

Proposition 1. *There exist two non-isomorphic graphs G_1, G_2 on 8 vertices all of whose degrees are equal 5. In particular G_1, G_2 have the same type-vector $(5^{*5}, 5^{*5}, 5^{*5}, 5^{*5}, 5^{*5}, 5^{*5}, 5^{*5}, 5^{*5})$, where $5^{*5} = (5, 5, 5, 5, 5) \in \mathbb{N}^*$.*

Proof. Define G_1, G_2 as on Figure 1.

Degree conditions are clearly fulfilled. It is left to ensure that the graphs are non-isomorphic. For this it is enough to ensure that the complement graphs are non-isomorphic. The complement to the first graph is two disjoint 4-cycles ACHF and BDGE (so it is disconnected), whereas the complement to the second graph is an 8-cycle AFEBHCDG (hence it is connected). The non-isomorphism is clear now.

■

Despite the fact that τ is not complete, it turns out to be **asymptotically complete**, i.e. for asymptotically almost all graphs G, H the equality $\tau_G = \tau_H$ implies that G, H are isomorphic (see [2, 13] and Section 7).

A graph G is said to be **oblique** if the map $\tau_G : V \to \mathbb{N}^*$ is injective or, in other words, there are no distinct vertices $u, v \in V(G)$ such that $\tau_G(u) = \tau_G(v)$ [14]. Figure 3 shows an example of oblique graph on 8 vertices [5].

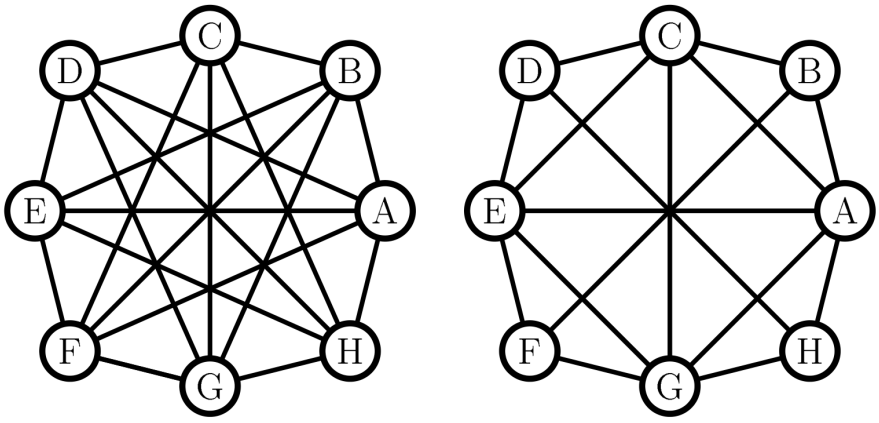

Figure 1. Graphs G_1 and G_2.

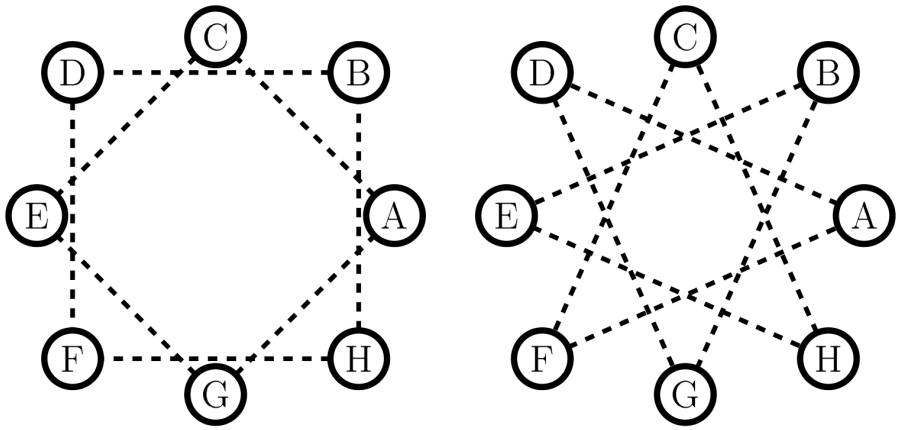

Figure 2. Complement graphs $\overline{G_1}$ and $\overline{G_2}$.

Lemma 1 (uniqueness of the type-vector)**.** *If a graph G is oblique then its type vector $\tau_G = (\tau_G(v_1), \ldots, \tau_G(v_n))$ is unique in the sense that $(\tau_G(v_1), \ldots, \tau_G(v_n)) = (\tau_G(w_1), \ldots, \tau_G(w_n))$ implies that $v_i = w_i$ for all i.*

Proof. Indeed, by assumption $\tau_G(v_i) = \tau_G(w_i)$ for all i, hence by obliqueness $v_i = w_i$ for all i. ■

Lemma 2 (retrieving an isomorphism)**.** *Let $\tau_G = (\tau_G(g_1), \ldots, \tau_G(g_n)), \tau_H = (\tau_H(h_1), \ldots, \tau_H(h_n))$ be the type-vectors of graphs G, H respectively. If G is oblique and $G \simeq H$ then the map $g_i \mapsto h_i$ is the isomorphism of G onto H and there is no other isomorphism between G and H can be drawn. In particular, every oblique graph has a trivial automorphism group.*

Proof. Let $\tau_G = (\tau_G(g_1), \ldots, \tau_G(g_n)), \tau_H = (\tau_H(h_1), \ldots, \tau_H(h_n))$ be the type-vectors of graphs G, H respectively, let G be oblique and let $\phi : G \to H$ be an

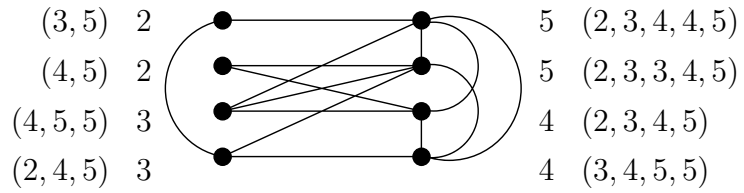

Figure 3. A vertex-oblique graph on 8 vertices. The degrees of vertices and their neighbors are written in columns

arbitrary isomorphism. By assumption $\tau_G : V(G) \to \mathbb{N}^*$ is injective, so $\tau_H = \tau_G \circ \phi^{-1} : V(H) \to \mathbb{N}^*$ is injective also and thus H is oblique. The vector of strings

$$(\tau_H(\phi g_1), \ldots, \tau_H(\phi g_n)) = (\tau_G(g_1), \ldots, \tau_G(g_n)) \tag{9}$$

is the type-vector of H because its coordinates are all type-strings of all vertices of H (looking at the left-hand side) and they are ShortLex ordered (looking at the right-hand side). Thus we have two type-vectors for H:

$$(\tau_H(\phi g_1), \ldots, \tau_H(\phi g_n)) = \tau_H = (\tau_H(h_1), \ldots, \tau_H(h_n)). \tag{10}$$

By the uniqueness Lemma $\phi g_i = h_i$ for all i. Hence the map $g_i \mapsto h_i = \phi g_i$ is an isomorphism of G onto H ; moreover, there is no other isomorphism between G and H. ■

5. Efficient solvability of $GI(P_{isom}, P_{oblique})$ and $SGI(P_{isom} \cap P_{oblique}, R)$

Consider the following sets of pairs of graphs:

$$P_{all} = \cup_n (\mathcal{G}_n \times \mathcal{G}_n), \tag{11}$$

$$P_{isom} = \{(G, H) \in P_{all} : G \simeq H\}, \tag{12}$$

$$P_{oblique} = \{(G, H) \in P_{all} : G \text{ and } H \text{ are oblique}\}. \tag{13}$$

It is easy to see that all these sets are decidable.

Theorem 2. *The promise decision problem $GI(P_{isom}, P_{oblique})$ is polynomial-time solvable.*

Proof. (Cf. [2,13]) This time the promise set $P_{oblique}$ does not contain the language P_{isom}, so we need the formalism of promise problems in full generality. Precisely, an algorithm A solves instance (G, H) of $P_{oblique}$ iff

$$A(G, H) = \begin{pmatrix} 1 & \text{if } (G, H) \in P_{oblique} \cap P_{isom} \\ 0 & \text{if } (G, H) \in P_{oblique} - P_{isom} \end{pmatrix}. \tag{14}$$

On instances out of $P_{oblique}$ the behavior of A may be rather arbitrary. Since $P_{oblique}$ is decidable, we can construct the required algorithm A by splitting into cases depending on whether a given instance (G, H) belongs to $P_{oblique}$ or not. Precisely, we put $A(G, H) = 0$ on instances $(G, H) \in I^2 - P_{oblique}$. The idea of the restriction algorithm $A|P_{oblique}$ is to distinguish all vertices of a graph using the invariant τ. Given a pair $(G, H) \in P_{oblique}$ the required algorithm A first computes the type-vectors

$$\tau_G = (\tau_G(g_1), \ldots, \tau_G(g_n)), \tau_H = (\tau_H(h_1), \ldots, \tau_H(h_n)). \quad (15)$$

If $\tau_G = \tau_H$ then A checks whether the map $\phi : g_i \mapsto h_i$ is an isomorphism from G to H or not. If it is an isomorphism then we put $A(G, H) = 1$. Otherwise Lemma 2 asserts that G, H are not isomorphic, so $A(G, H) = 0$.Thus A is well defined on all inputs I^2 and it solves the given problem. It is easy to see that the running time of A is linear in $|V| + |E|$. ■

Theorem 3. *The problem* $SGI(P_{isom} \cap P_{oblique}, R)$ *with relation*

$$R = \left\{((G, H), \phi) \in I^2 \times I : \exists n \text{ such that } G, H \in \mathcal{G}_n, \phi \in S_n \subseteq I_{n^2} \text{ and } \phi : G \simeq H\right\} \quad (16)$$

is polynomial-time solvable.

Proof. By definition the given problem is solved by algorithm A if for every $(G, H) \in P_{isom} \cap P_{oblique}$ the inclusion $((G, H), A(G, H)) \in R$ holds in case $G \simeq H$ and $A(G, H) = \perp$ otherwise. Given the $n-$vertex graphs G, H the required algorithm A first computes the type-vectors

$$\tau_G = (\tau_G(g_1), \ldots, \tau_G(g_n)), \tau_H = (\tau_H(h_1), \ldots, \tau_H(h_n)). \quad (17)$$

If the coordinates either of τ_G or τ_H are not distinct then at least one of graphs is not oblique i.e. $(G, H) \notin P_{isom} \cap P_{oblique}$. In this case we let $A(G, H) = id \in S_n$. Otherwise both of graphs are oblique and then algorithm compares the type-vectors. If $\tau_G \neq \tau_H$, then G, H are not isomorphic and thus $A(G, H)$ is already defined. If $\tau_G = \tau_H$ then the algorithm checks whether the map $\phi : g_i \mapsto h_i$ is an isomorphism or not. If this map turns out to be an isomorphism then the algorithm gives ϕ as the answer to the problem. Otherwise, if the map is not an isomorphism, Lemma 2 asserts that there is no isomorphism between the given graphs at all. That is this case is impossible by the very setup of the problem. Thus the algorithm A gives the right answer for the set of inputs $P_{isom} \cap P_{oblique}$ and $A(G, H) = id$ on the complement set $\overline{P_{isom} \cap P_{oblique}}$, which may be the wrong answer (but also may be right answer occasionally). ■

6. Generic algorithms for GI and SGI and their failure probability

Theorem 4. *Endow the set* P_{all} *with the ensemble of distributions* (D_n)*, where* D_n *is the uniform distribution on the set* P_{all}^n *of pairs of* n*-vertex graphs. Then* $P_{isom} \cap P_{oblique}$ *is asymptotically almost certain in* P_{isom}*.*

Proof. Let $\mathcal{G}_n^o$ denote the set of oblique graphs on vertices $1, 2, ..., n$. We rely on the result from [2,13] which says that the property of a graph $G \in \mathcal{G}_n$ to be oblique holds asymptotically almost surely, i.e.

$$\lim_{n\to\infty} \frac{|\mathcal{G}_n^o|}{|\mathcal{G}_n|} = 1, \tag{18}$$

or, equivalently,

$$\lim_{n\to\infty} \frac{|\mathcal{S}_n|}{|\mathcal{G}_n|} = 0, \tag{19}$$

where $\mathcal{S}_n = \mathcal{G}_n - \mathcal{G}_n^o$.
We have to prove that

$$\lim_{n\to\infty} \frac{\left|P_{isom}^n \cap P_{oblique}^n\right|}{|P_{isom}^n|} = 1, \tag{20}$$

or, equivalently,

$$\lim_{n\to\infty} \frac{|P_{isom}^n \cap (\mathcal{S}_n \times \mathcal{S}_n)|}{|P_{isom}^n|} = 0. \tag{21}$$

Consider the natural projection map onto the first coordinate $pr_1 : P_{isom}^n \to \mathcal{G}_n$. What is the cardinality of the inverse image (=fiber over G) of a graph G under pr_1? Each isomorphism class of $G \in \mathcal{G}_n$ consists of at most $n!$ elements, so $\left|pr_1^{-1}G\right| \leqslant n!$ for every $G \in \mathcal{G}_n$, hence

$$|P_{isom}^n| \geqslant n! \, |\mathcal{G}_n| \, . \tag{22}$$

In case G is oblique, we can say more: $\left|pr_1^{-1}G\right| = n!$. Indeed, by Lemma 2, $Aut\,(G) = 1$ and thus the orbit S_nG consists of $n!$ graphs and the set (G, S_nG) is the inverse image of G under pr_1. Thus each fiber of the map $pr_1 : P_{isom}^n \cap (\mathcal{G}_n^o \times \mathcal{G}_n^o) \to$ $\to \mathcal{G}_n^o$ has cardinality $n!$. Hence $|P_{isom}^n \cap (\mathcal{G}_n^o \times \mathcal{G}_n^o)| = n! \, |\mathcal{G}_n^o|$ and so

$$|P_{isom}^n| \geqslant n! \, |\mathcal{G}_n^o| \, . \tag{23}$$

Considering the fibers of pr_1 over the set $\mathcal{S}_n$, we conclude that

$$|P_{isom}^n \cap (\mathcal{S}_n \times \mathcal{S}_n)| \leqslant n! \, |\mathcal{S}_n| \, . \tag{24}$$

Finally, combining inequalities 23 and 24 and the result 19, we obtain

$$\frac{|P_{isom}^n \cap (\mathcal{S}_n \times \mathcal{S}_n)|}{|P_{isom}^n|} \leqslant \frac{n! \, |\mathcal{S}_n|}{n! \, |\mathcal{G}_n^o|} = \frac{|\mathcal{S}_n|}{|\mathcal{G}_n^o|} = \frac{|\mathcal{S}_n|}{|\mathcal{G}_n| - |\mathcal{S}_n|} = \tag{25}$$

$$= \frac{|\mathcal{S}_n|}{|\mathcal{G}_n|} \left(\frac{1}{1 - \frac{|\mathcal{S}_n|}{|\mathcal{G}_n|}} \right) \to 0, \ n \to \infty. \tag{26}$$

■

Remark 2. It is shown in [2] a linear time, high probability canonical labeling algorithm for $G(n,p)$ graphs for $p = \omega(\ln^4 n/n \ln\ln n)$ and $p \leqslant 1/2$. Here, high probability means probability at least $1 - O(n^{-c})$ for every $c > 0$. It follows that $P_{isom} \cap P_{oblique}$ is asymptotically almost certain with the rate of convergence at least $1 - O(n^{-c})$ for every $c > 0$.

Here we prove the main theorem in the following more precise formulation.

Theorem 5. *Any algorithm A, constructed in Section 5. to solve the problem $SGI\,(P_{isom} \cap P_{oblique}, R)$, solves also the problem $SGI\,(P_{isom}, R)$ for asymptotically almost all inputs $P_{isom} \cap P_{oblique}$. In other words, the failure probability of the algorithm A tends to zero as n tends to infinity.*

7. Applications to probabilistic algorithms

Recall that the input set of the problem $SGI\,(P_{isom}, R)$ is $I^2 \times I$, the promise set is

$$P_{isom} = \{(G,H) : \exists n \text{ such that } G, H \in \mathcal{G}_n,\ G \simeq H\}\,, \tag{27}$$

and the relation is

$$R = \left\{(G,H,\phi) : \exists n \text{ such that } G, H\ \in \mathcal{G}_n, \phi \in S_n \subseteq I_{n^2} \text{ and } \phi : G \simeq H\right\}. \tag{28}$$

A.N. Rybalov considered a search graph isomorphism problem with particularly small promise set $P_\gamma \subseteq P_{isom}$ [12]. Namely, fix an infinite sequence of graphs $\gamma = (G_n)_{n\in\mathbb{N}}$, such that $G_n \in \mathcal{G}_n$ for all n and define

$$P_\gamma = \{(G, G_n) : n \in \mathbb{N}, \text{ and } G, G_n \in \mathcal{G}_n,\ G \simeq G_n\}\,. \tag{29}$$

A.N. Rybalov studied the problems of the type $SGI\,(P_\gamma, R)$ which clearly are all the restrictions of the candid search problem $SGI\,(P_{isom}, R)$.

The main result of A.N.Rybalov is the following.

Theorem 6. *[12] If there exists a polynomial generic algorithm for $SGI\,(P_\gamma, R)$, then there exists a polynomial probabilistic algorithm computing $SGI\,(P_\gamma, R)$ for all inputs.*

From this and from our Theorem 5 we derive the following

Theorem 7. *If the sequence of graphs $\gamma = (G_n)_{n\in\mathbb{N}}$ consists of oblique graphs, then the problem $SGI\,(P_\gamma, R)$ is polynomial-time solvable and there exists a polynomial probabilistic algorithm computing $SGI\,(P_\gamma, R)$ for all inputs.*

Acknowledgements

The author is thankful to A.N.Rybalov for stating the SGI.

References

1. Bogdanov A., Trevisan L. Average-case complexity // Found. Trends Theor. Comput. Sci. October 2006. V. 2(1). P. 1–106.
2. Czajka T., Pandurangan G. Improved random graph isomorphism // J. of Discrete Algorithms. March 2008. V. 6(1). P. 85–92.
3. Epstein D.B.A., Cannon J.W., Holt D.F., Levy S.V.F., Paterson M.S., Thurston W.P. Word processing and group theory. Jones and Bartlet Publishers, 1992.
4. Even S., Selman A.L., Yacobi Y. The complexity of promise problems with applications to public-key cryptography // Inform. and Control. 1984. V. 61(2). P. 159–173.
5. Farrugia A. Dually vertex-oblique graphs // Discrete Mathematics. 2007. V. 307(11–12). P. 1323–1331. The Fourth Caracow Conference on Graph Theory Czorsztyn 2002.
6. Goldreich O. On promise problems: a survey. In Theoretical computer science // Lecture Notes in Comput. Sci. Springer, Berlin, 2006. V. 3895. P. 254–290.
7. Goldreich O. Computational complexity – a conceptual perspective. Cambridge University Press, 2008.
8. Goldreich O. P, NP, and NP-Completeness: The Basics of Complexity Theory. Cambridge University Press, 2010.
9. Hoffmann C.M. Group-theoretic algorithms and graph isomorphism // Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1982. V. 136.
10. Longpré L. Selman A.L. Hard promise problems and nonuniform complexity. In STACS 90 (Rouen, 1990) // Lecture Notes in Comput. Sci. Springer, Berlin, 1990. V. 415. P. 216–226.
11. Myasnikov A., Shpilrain V., Ushakov A. Group-based Cryptography. Advanced Courses in Mathematics CRM Barcelona. Birkhauser Verlag, 2008.
12. Rybalov A. On the generic complexity of the searching graph isomorphism problem. Omsk State Technical University, 2015.
13. Schreyer J. Almost every graph is vertex-oblique // Discrete Math. 2007. V. 307(7-8). P. 983–989.
14. Schreyer J., Walther H., Mel'nikov L.S. Vertex-oblique graphs // Discrete Mathematics. 2007. V. 307(11–12). P. 1538 – 1544. The Fourth Caracow Conference on Graph Theory Czorsztyn 2002.

ПРОБЛЕМЫ ПОИСКА С ПОСУЛОМ И ИЗОМОРФИЗМ ГРАФОВ

Г.А. Носков
д.ф.-м.н., с.н.с., e-mail: g.noskov@googlemail.com

Институт Математики им. С.Л. Соболева СОРАН

Аннотация. Проблема изоморфизма графов рассматривается с точки зрения теории «проблем с посулом», развитой Ивеном, Зелманом и Якоби [4]. Изучается «тау-инвариант» графов и с его помощью решается проблема поиска изоморфизма для асимптотически почти всех графов.

Ключевые слова: проблема разрешимости, проблема поиска с посулом, изоморфизм графов.