Theoretical Computer Science 411 (2010) 3859–3866

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Reconstruction of interval graphs[☆]

Masashi Kiyomi [a,*], Toshiki Saitoh [b], Ryuhei Uehara [a]

[a] *School of Information Science, JAIST, Asahidai 1-1, Nomi, Ishikawa 923-1292, Japan*

[b] *ERATO MINATO Discrete Structure Manipulation System Project, JST, North 14, West 9, Sapporo, Hokkaido 060-0814, Japan*

A R T I C L E I N F O

Article history:
Received 4 January 2010
Received in revised form 23 June 2010
Accepted 3 July 2010
Communicated by O. Watanabe

Keywords:
The graph reconstruction conjecture
Interval graphs
Polynomial time algorithm

A B S T R A C T

The graph reconstruction conjecture is a long-standing open problem in graph theory. There are many algorithmic studies related to it, such as DECK CHECKING, LEGITIMATE DECK, PREIMAGE CONSTRUCTION, and PREIMAGE COUNTING. We study these algorithmic problems by limiting the graph class to interval graphs. Since we can solve GRAPH ISOMORPHISM for interval graphs in polynomial time, DECK CHECKING for interval graphs is easily done in polynomial time. Since the number of interval graphs that can be obtained from an interval graph by adding a vertex and edges incident to it can be exponentially large, developing polynomial time algorithms for LEGITIMATE DECK, PREIMAGE CONSTRUCTION, and PREIMAGE COUNTING on interval graphs is not trivial. We present that these three problems are solvable in polynomial time on interval graphs.

1. Introduction

Given a simple graph $G = (V, E)$, we call the multi-set $\{G - v \mid v \in V\}$ the *deck* of G where $G - v$ is a graph obtained from G by removing vertex v and incident edges. The graph reconstruction conjecture by Ulam and Kelly[1] is that, for any multi-set D of graphs with at least two vertices, there is at most one graph whose deck is D. We call a graph whose deck is D a preimage of D. No counter-example is known for this conjecture, and there are many mathematical results about this conjecture. For example, trees, regular graphs, and disconnected graphs are reconstructible (i.e. the conjecture is true for these classes) [7]. Almost all graphs are reconstructible from three well-chosen graphs in its deck [1]. Rimscha showed that interval graphs are recognizable in the sense that, by looking at the deck of G, one can decide whether or not G belongs to interval graphs [14]. In the same paper, Rimscha also showed that many subclasses of perfect graphs, including perfect graphs themselves, are recognizable, and some of the subclasses including unit interval graphs are reconstructible. There are many good surveys about this conjecture. See for example [2,5].

Besides these mathematical results, there are some algorithmic results. We enumerate the algorithmic problems that we address in this paper.

- Given a graph G and a multi-set of graphs D, check whether D is a deck of G (DECK CHECKING).
- Given a multi-set of graphs D, determine whether there is a graph whose deck is D (LEGITIMATE DECK).
- Given a multi-set of graphs D, construct a graph whose deck is D (PREIMAGE CONSTRUCTION).
- Given a multi-set of graphs D, compute the number of (pairwise non-isomorphic) graphs whose decks are D (PREIMAGE COUNTING).

[☆] A preliminary version of this article was presented at COCOON 2009 [8].

[*] Corresponding author. Tel.: +81 761 51 1207.
E-mail addresses: mkiyomi@jaist.ac.jp (M. Kiyomi), t-saitoh@erato.ist.hokudai.ac.jp (T. Saitoh), uehara@jaist.ac.jp (R. Uehara).

[1] Determining the first person who proposed the graph reconstruction conjecture is difficult, actually. See [5] for the detail.

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.07.006

Kratsch and Hemaspaandra showed that these problems are solvable in polynomial time for graphs of bounded degree, partial k-trees for any fixed k, and graphs of bounded genus, in particular for planar graphs [11]. In the same paper they proved many graph isomorphism(GI)-related complexity results. Hemaspaandra et al. extended the results [6]. Note that the fact that graph class $\mathcal{C}$ is reconstructible does not mean that there is a polynomial time algorithm of PREIMAGE CONSTRUCTION for $\mathcal{C}$. For example, though disconnected graphs are reconstructible, it seems very hard to develop a polynomial time PREIMAGE CONSTRUCTION algorithm for disconnected graphs, since we do not know any polynomial time isomorphism algorithm for them. It is likely that a polynomial time PREIMAGE CONSTRUCTION algorithm on some graph class will gives some hint to a proof of the graph reconstruction conjecture on the class. Interval graph is one of the large major graph class for which we have an isomorphism algorithm, and thus we can consider reconstruction algorithms without bothering about graph isomorphism.

There is a linear time algorithm for determining if given two interval graphs are isomorphic [13]. Thus, developing a polynomial time algorithm for DECK CHECKING for interval graphs is easy.

Theorem 1. *There is an $O(n(n + m))$ time algorithm of DECK CHECKING for n-vertex m-edge graph and its deck (or a deck candidate) that consists of interval graphs.*

We will give the proof in Section 3. Note that we require only graphs in the deck candidate to be interval graphs, and we do not require the preimage candidate to be an interval graph. Thus this result is not straightforward from [11].

LEGITIMATE DECK, PREIMAGE CONSTRUCTION, and PREIMAGE COUNTING for interval graphs are solvable by almost the same algorithm. In order to develop such an algorithm, we show that, given a set of n interval graphs D, there are at most $O(n^2)$ graphs (preimages) whose decks are D. Further, we can construct such $O(n^2)$ preimage candidates. Our algorithm checks these $O(n^2)$ candidates one by one whether its deck is D with DECK CHECKING algorithm. Our algorithm constructs n preimage candidates from $O(n)$ different interval representations of each interval graph in D by inserting an interval to them. The key is that the number of preimage candidates is $O(n^2)$, while a naive algorithm which inserts an interval to an interval representation may construct $\Omega(2^n)$ candidates. (Consider the case that $\Theta(n)$ intervals terminate at some point t, and we insert a new left endpoint to t. The number of the ways of insertions is $\Theta(2^n)$, since there are $\Theta(n)$ choices whether the new interval intersects the old ones. Further, there may be many, say $\Theta(2^n)$, different compact interval representations for an interval graph. Therefore, the number of preimage candidates will be very huge, if we construct the candidates from all of them.)

The following is our main theorem.

Theorem 2. *There are $O(n^3(n + m))$ time algorithms for LEGITIMATE DECK and PREIMAGE CONSTRUCTION, and there is an $O(n^4(n + m))$ time algorithm for PREIMAGE COUNTING, for n interval graphs.*

Note that m is the number of edges in the preimage. Kelly's lemma [7] shows that we can compute m from the deck.

We state terminologies in Section 2, then explain about interval graphs in Section 3. In Section 3, we introduce many small lemmas for those who unfamiliar to interval graphs. Most of these lemmas may be well known and/or basic for those who familiar to interval graphs and the concepts of PQ-tree [3] and MPQ-tree [10]. However, these lemmas play important roles in this paper. Then, we show that the number of preimage candidates is $O(n^2)$, and we present our algorithm in Section 4. Finally, we make some remarks in Section 5.

2. Terminology

Graphs in this paper are all simple and undirected, unless explicitly stated. We denote by $N_G(v)$ the neighbor set of vertex v in graph G, and we denote by $N_G[v]$ the closed neighbor set of vertex v in graph G. "Closed" means that $N_G[v]$ contains v itself, i.e. $N_G[v] = N_G(v) \cup \{v\}$ holds. We denote by $\deg_G(v)$ the degree of vertex v in graph G. We omit the subscript G, when there is no confusion about the base graph. The sum of degrees of all vertices in graph G is denoted by degsum(G). Note that degsum(G) is equal to twice the number of edges in G. We denote by $\tilde{G}$ the graph obtained by adding one universal vertex to the graph G such that the vertex connects to every vertex in G. Note that $\tilde{G}$ is always connected. Given two graphs G_1 and G_2, we define the disjoint union $G_1 \dot\cup G_2$ of G_1 and G_2 as $(V_1 \dot\cup V_2, E_1 \dot\cup E_2)$ such that (V_1, E_1) is isomorphic to G_1, and (V_2, E_2) is isomorphic to G_2, where $\dot\cup$ means the disjoint union.

3. Interval Graphs

A graph $G = (V, E)$ with $V = \{v_1, v_2, \ldots, v_n\}$ is an *interval graph* iff there is a multi-set $\mathcal{I} = \{I_{v_1}, I_{v_2}, \ldots, I_{v_n}\}$ of closed intervals on the real line such that $\{v_i, v_j\} \in E$ if and only if $I_{v_i} \cap I_{v_j} \neq \emptyset$ for each i and j with $1 \leq i, j \leq n$. We call the multi-set $\mathcal{I}$ an *interval representation* of G. An interval graph may have infinitely many interval representations. We use a tractable one called compact interval representation among them.

3.1. Compact representation and basic lemmas

Definition 1 (*[15]*)**.** An interval representation $\mathcal{I}$ of an interval graph $G = (V, E)$ is *compact* iff

- coordinates of endpoints of intervals in $\mathcal{I}$ are finite non-negative integers (We denote by K the largest coordinates of endpoints for convenience. We sometimes call K the length of $\mathcal{I}$),

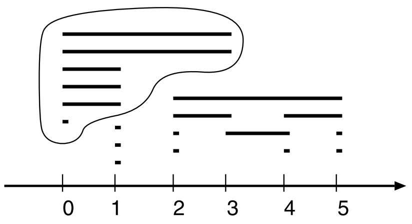

Fig. 1. A compact interval representation of an interval graph. Every interval graph has at least one compact interval representation. Vertices corresponding to the enclosed intervals are end-vertex set.

- there exists at least one endpoint whose coordinate is k for every integer $k \in [0, K]$, and
- interval multi-set $\mathcal{I}_k = \{I \in \mathcal{I} \mid k \in I\}$ differs from $\mathcal{I}_l = \{I \in \mathcal{I} \mid l \in I\}$, and they do not include each other, for every distinct integers $k, l \in [0, K]$.

We show an example of a compact interval representation of an interval graph in Fig. 1. Note that there may still be many compact interval representations of an interval graph. However, compact interval representations have some good properties.

Lemma 3. *Let $\mathcal{I}$ and $\mathcal{J}$ be compact interval representations of an interval graph $G = (V, E)$, and let K_1 be the length of $\mathcal{I}$, and let K_2 be the length of $\mathcal{J}$. Then, the following holds.*

$$\{\{I \in \mathcal{I} \mid 0 \in I\}, \{I \in \mathcal{I} \mid 1 \in I\}, \ldots, \{I \in \mathcal{I} \mid K_1 \in I\}\} = \{\{I \in \mathcal{J} \mid 0 \in I\}, \{I \in \mathcal{J} \mid 1 \in I\}, \ldots, \{I \in \mathcal{J} \mid K_2 \in I\}\}.$$

Proof. We denote by $\bar{\mathcal{I}}$ the set of multi-set of intervals $\{\{I \in \mathcal{I} \mid 0 \in I\}, \{I \in \mathcal{I} \mid 1 \in I\}, \ldots, \{I \in \mathcal{I} \mid K_1 \in I\}\}$, and we denote by $\bar{\mathcal{J}}$ the set of multi-set of intervals $\{\{I \in \mathcal{J} \mid 0 \in I\}, \{I \in \mathcal{J} \mid 1 \in I\}, \ldots, \{I \in \mathcal{J} \mid K_2 \in I\}\}$. The vertices represented by the multi-set of intervals $\mathcal{I}_i = \{I \in \mathcal{I} \mid i \in I\}$ correspond to a clique in G. Assume that $\mathcal{I}_i$ never appears in $\bar{\mathcal{J}}$ for some i. Since $\mathcal{I}_i$ represents a clique C, there must be a set of intervals representing a clique C' containing C in $\bar{\mathcal{J}}$ (otherwise, clique C cannot be represented in $\mathcal{J}$). Then for the same reason, $\bar{\mathcal{I}}$ must contain a set of intervals representing a clique containing C'. This contradicts the compactness of $\mathcal{I}$. $\square$

From the proof of Lemma 3, the following lemmas are straightforward.

Lemma 4. *Let $\mathcal{I}$ be a compact interval representation of an interval graph $G = (V, E)$, and let K be the length of $\mathcal{I}$. Then, $\{I \in \mathcal{I} \mid i \in I\}$ for each $i \in \{0, \ldots, K\}$ corresponds to each maximal clique of G.*

Lemma 5. *The length of a compact interval representation of an n-vertex interval graph is at most n.*

Note that the number of maximal cliques in an n-vertex interval graph is at most n (see [4]).

Lemma 6. *All the compact interval representations of an interval graph have the same length.*

Lemma 7. *Intervals in different compact interval representations corresponding to an identical vertex have the same length.*

From Lemma 7, lengths of intervals corresponding to a vertex that corresponds to an interval of length zero in some interval representation are always (i.e. in any interval representation) zero. Such a vertex is called *simplicial*.

3.2. PQ-trees and MPQ-trees

In this subsection, we explain PQ-trees and MPQ-trees. We can get the other perspective for the lemmas in the previous subsection with (M)PQ-trees, and it may be good for the understanding.

PQ-tree was introduced by Booth and Lueker [3]. We can use it for recognizing interval graphs. A PQ-tree is a rooted tree T with two types of internal nodes, P- and Q-nodes. Each leaf of T is labeled with each maximal clique of the interval graph G. The *frontier* of a PQ-tree T is the permutation of the maximal cliques obtained by the ordering of the leaves of T from left to right. The definition that PQ-tree T corresponds to interval graph G is given as follows.

Definition 2. PQ-tree T corresponds to interval graph G, if and only if, for every PQ-tree T' obtained from T by applying the following rules (1) and (2) a finite number of times, there is an interval representation of G in which the maximal cliques of G appear in the same order of the frontier of T'.

(1) arbitrarily permute the successor nodes of a P-node, or
(2) reverse the order of the successor nodes of a Q-node.

See for example Fig. 2. Booth and Lueker developed a linear time algorithm that either constructs a PQ-tree for G, or states that G is not an interval graph.

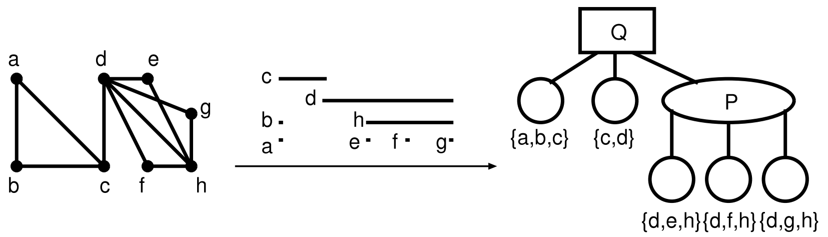

Fig. 2. An example of an interval graph, its interval representation, and its PQ-tree.

```
boolean function deck-checking(graph $G = (V, E)$) {
  Let $G'$ be an empty graph.
  for each vertex $v \in V$ {   $G' := G' \dot{\cup} (\tilde{G} - v)$.   }
  if $G'$ is isomorphic to $\tilde{G}'_1 \dot{\cup} \tilde{G}'_2 \dot{\cup} \ldots \dot{\cup} \tilde{G}'_n$
    return True   else   return False.
}
```

Fig. 3. The deck checking algorithm.

MPQ-tree, which stands for *modified PQ-tree*, is developed by Korte and Möhring to simplify the algorithm for the PQ-tree [10]. The MPQ-tree T^* assigns sets of vertices (or intervals from the view of interval representation) to the nodes of a PQ-tree T representing an interval graph $G = (V, E)$.

Lemma 4 says that each $\{I \in \mathcal{I} \mid i \in I\}$ corresponds to a maximal clique. Therefore, it corresponds to a leaf of a PQ-tree. Thus, Lemma 3 says that in every PQ-tree of an interval graph G, the set of leaves are identical. Lemma 5 says that in every PQ-tree of an interval graph $G = (V, E)$, the number of leaves are at most $|V|$. Lemma 6 says that in every PQ-tree of an interval graph G, the number of leaves are the same. And, Lemma 7 says that in every PQ-tree of an interval graph G, the number of leaves that a vertex v belongs are the same.

3.3. Deck checking

Now we prove Theorem 1. Our main algorithm enumerates the preimage candidates, and checks whether each candidate is really a preimage of the input deck. Thus, the theorem is one of the basic parts of our algorithm. We first show a lemma to prove Theorem 1.

Lemma 8. *Given an interval graph G which can be connected or disconnected, $\tilde{G}$ is always a connected interval graph.*

Proof. It is obvious that $\tilde{G}$ is connected. Consider a compact interval representation $\mathcal{I}$ of G. Let K be the length of $\mathcal{I}$. Then, $\mathcal{I} \cup \{[0, K]\}$ is an interval representation of $\tilde{G}$. Therefore, $\tilde{G}$ is a connected interval graph. □

Proof of Theorem 1. Let $G = (V, E)$ be a graph, where V is $\{1, 2, \ldots, n\}$, and $|E|$ is equal to m. Let $G_i (i \in V)$ be a graph obtained by removing vertex i from G. Suppose that $G_1, G_2, \ldots, G_n$ are interval graphs. It is clear that $\{G'_1, G'_2, \ldots, G'_n\}$ is a deck of G if and only if the multi-set $\{\tilde{G}_1, \tilde{G}_2, \ldots, \tilde{G}_n\}$ is equal to the multi-set $\{\tilde{G}'_1, \tilde{G}'_2, \ldots, \tilde{G}'_n\}$. Hence, we can determine whether or not the given multi-set $D = \{G'_1, G'_2, \ldots, G'_n\}$ is a deck of the input graph G by checking whether or not $\tilde{G}_1 \dot{\cup} \tilde{G}_2 \dot{\cup} \ldots \dot{\cup} \tilde{G}_n$ is isomorphic to $\tilde{G}'_1 \dot{\cup} \tilde{G}'_2 \dot{\cup} \ldots \dot{\cup} \tilde{G}'_n$. Since the disjoint union of two interval graphs is an interval graph, we can use well-known linear time isomorphism algorithm [13] for this checking. We describe the algorithm in Fig. 3. Since the number of vertices of $\tilde{G}_1 \dot{\cup} \ldots \dot{\cup} \tilde{G}_n$ is $O(n^2)$, and since the number of edges of $\tilde{G}_1 \dot{\cup} \ldots \dot{\cup} \tilde{G}_n$ is $O(mn + n^2)$, the time complexity of this algorithm is $O(n(n + m))$. □

Note that we cannot use $G_1 \dot{\cup} G_2 \dot{\cup} \ldots \dot{\cup} G_n$ instead of $\tilde{G}_1 \dot{\cup} \tilde{G}_2 \dot{\cup} \ldots \dot{\cup} \tilde{G}_n$. If some of $G_1, G_2, \ldots, G_n$ are disconnected, $G_1 \dot{\cup} G_2 \dot{\cup} \ldots \dot{\cup} G_n$ and $G'_1 \dot{\cup} G'_2 \dot{\cup} \ldots \dot{\cup} G'_n$ can be isomorphic, even if $\{G_1, G_2, \ldots, G_n\}$ and $\{G'_1, G'_2, \ldots, G'_n\}$ are not identical. See for example Fig. 4.

3.4. Non-interval graph preimage case

Our algorithm described in the next section outputs preimages that are interval graphs. However, it is possible that a non-interval graph has a deck that consists of interval graphs, though it is exceptional. Since considering this case all the time in the main algorithm makes it complex, we attempt to get done with this special case.

First of this subsection, we introduce a famous theorem below.

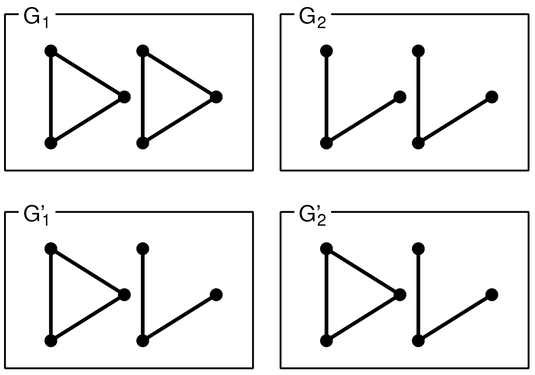

Fig. 4. $G_1 \dot{\cup} G_2$ is isomorphic to $G'_1 \dot{\cup} G'_2$, though the multi-sets $\{G_1, G_2\}$ and $\{G'_1, G'_2\}$ are different.

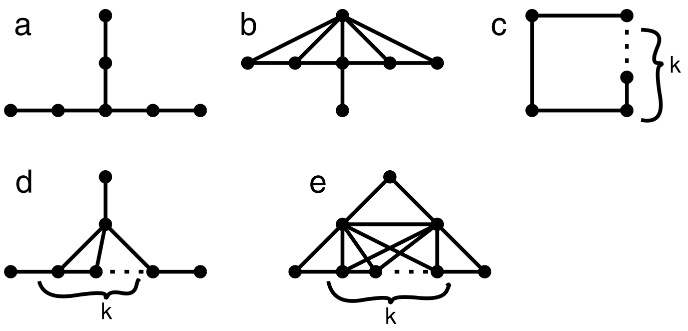

Fig. 5. The forbidden graphs of interval graphs. The part described k contains k vertices ($k \geq 1$). Thus, (c) is a chordless cycle of more than three vertices, (d) has more than five vertices, and (e) has more than five vertices.

Theorem 9 (*Lekkerkerker and Boland [12]*). *Graph G is an interval graph if and only if G has no graph described in Fig. 5 as an induced subgraph.*

Note that we can easily prove that all the members in a deck of an interval graph are interval graphs.

Theorem 10. *If n interval graphs $G_1, G_2, \ldots, G_n$ have a preimage G that is not an interval graph, we can reconstruct G from $G_1, G_2, \ldots, G_n$ in $O(n^2)$ time.*

Proof. Assume that $G_1, G_2, \ldots, G_n$ are the deck of G, and G is not an interval graph. Then, G must be one of the graphs described in Fig. 5, since any graph that is obtained by removing a vertex from G is an interval graph (containing none in Fig. 5). It is clear that $G_1, G_2, \ldots, G_n$ have the same number of vertices, $n - 1$, and the number of vertices in G is n. Since the number of graphs of size n in Fig. 5 is $O(1)$, we can check if one of them is a preimage of the input graphs in polynomial time with DECK CHECKING algorithm. The time complexity is $O(n(n + m))$ from Theorem 1, where m is the number of edges of a preimage. Since the numbers of edges in (a), (b), (c), (d), and (e) are $O(n)$, the time complexity is definitely $O(n^2)$. □

Therefore, we concentrate on an algorithm that tries to reconstruct an interval graph whose deck is the set of the input graphs in the remaining sections.

4. Main algorithm

4.1. Connected preimage case

It is possible that there is no connected preimage interval graph but a disconnected preimage interval graph of given n interval graphs. We consider this case later. Here in this subsection, we consider an algorithm for determining whether or not there are connected preimage interval graphs, and if any, returns them.

First we define end-vertex set. The end-vertex set is intuitively a set of vertices whose corresponding intervals are at the left end in at least one interval representation. Our algorithm adds a vertex adjacent to all the vertices in an end-vertex set of an interval graph in the input deck. This enables us to avoid exponential times' constructions of preimage candidates.

Definition 3. For an interval graph $G = (V, E)$, we call a vertex subset $S \subset V$ an *end-vertex set* iff, in some compact interval representation of G, all the coordinates of the left endpoints of intervals corresponding to vertices in S are 0, and S is maximal among such vertex subsets with respect to the interval representation.

See Fig. 1 for example. It is clear from the definition of compact interval representations that an end-vertex set contains at least one simplicial vertex.

We show some lemmas about end-vertex sets. We can estimate that the number of essentially different preimage candidates is $O(n^2)$ by these lemmas.

Lemma 11. *Let S be an end-vertex set of an interval graph $G = (V, E)$. If two vertices v and w in S have the same degree, then $N[v]$ is equal to $N[w]$.*

Proof. Since v and w are in S, on some compact representation of G, the interval corresponding to v is $I_v = [0, k_v]$, and the interval corresponding to w is $I_w = [0, k_w]$ for some k_v and k_w. Assume that k_v is not equal to k_w. We can assume that k_v is greater than k_w without loss of generality. Then, $N[w] \subsetneq N[v]$ holds due to the definition of a compact representation. This contradicts the fact that v and w have the same degree (see Fig. 1 for the better understanding). □

Lemma 12. *A connected interval graph has at most $O(n)$ end-vertex sets.*

Proof. An end-vertex set of an interval graph G is in the form $\{I \in \mathcal{I} \mid 0 \in I\}$ for some interval representation $\mathcal{I}$ of G. Thus, from Lemmas 3 and 5, there are at most $O(n)$ end-vertex sets for G. □

Now we refer the well-known lemma about the degree sequence.

Lemma 13 (*Kelly's Lemma [7]*)**.** *We can calculate the degree sequence of a preimage of the input n graphs in $O(n)$ time, if we know the number of edges in each input graph.*

Proof. Let $G_1, G_2, \ldots, G_n$ be the input graphs. Assume that graph G has a deck $\{G_1, G_2, \ldots, G_n\}$. There are vertices $v_1, v_2, \ldots, v_n$ such that G_i is obtained by removing v_i from G for each i in $\{1, 2, \ldots, n\}$. Thus,

$$\mathrm{degsum}(G_i) = \mathrm{degsum}(G) - 2\deg_G(v_i)$$

holds for each $i \in \{1, 2, \ldots, n\}$. Hence, we have

$$\mathrm{degsum}(G) = \frac{\sum_{i=1}^{n} \mathrm{degsum}(G_i)}{n-2}.$$

Therefore, we can easily calculate the degree sequence of G, i.e., $(\mathrm{degsum}(G) - \mathrm{degsum}(G_1))/2, (\mathrm{degsum}(G) - \mathrm{degsum}(G_2))/2, \ldots, (\mathrm{degsum}(G) - \mathrm{degsum}(G_n))/2$.

We can calculate $\mathrm{degsum}(G_i)$ in constant time, provided we know the number m_i of edges in G_i, for $\mathrm{degsum}(G_i)$ is equal to $2m_i$. Thus, the time complexity to calculate $\mathrm{degsum}(G)$ is $O(n)$, and the total time complexity to obtain the degree sequence of G is also $O(n)$. □

Now we present an algorithm for reconstructing a connected interval graph. Suppose that an n-vertex connected interval graph G has a deck of interval graphs $\{G_1, G_2, \ldots, G_n\}$. Let $\mathcal{I}$ be a compact interval representation of G. There must be an index $i \in \{1, \ldots, n\}$ such that G_i is obtained by removing a simplicial vertex s in the end-vertex set S corresponding to $\mathcal{I}$. We show that we can reconstruct G from G_i. Moreover, we can check whether or not G_j is G_i for every $j \in \{1, \ldots, n\}$. Therefore, we can reconstruct G by checking if G_j is the desired G_i for every $j \in \{1, \ldots, n\}$.

There are two cases about the number of simplicial vertices in S.

(i) S has only one simplicial vertex s.
(ii) S has at least two simplicial vertices.

In the case (ii), $S \setminus \{s\} = N_G(s)$ is still an end-vertex set of G_i. In the case (i), $S \setminus \{s\}$ is contained by vertices corresponding to $\{I \in \mathcal{I} \mid 1 \in I\}$ which is an end-vertex set of G_i. In both the cases, we thus have to add a simplicial vertex to an end-vertex set of G_i in order to reconstruct G. We denote the end-vertex set of G_i by $\tilde{S}$ ($S \setminus \{s\} \subset \tilde{S}$ holds). Since there are $O(n)$ end-vertex sets of G_i by Lemma 12, checking if each end-vertex set S' in G_i is $\tilde{S}$ takes $O(n)$ times iterations. Checking whether an end-vertex set S' is $\tilde{S}$ is simple. Since if S' is $\tilde{S}$, we can reconstruct G, we simply try to reconstruct G. If we can reconstruct G, S' is $\tilde{S}$, and we of course obtain G. Otherwise, S' is not $\tilde{S}$.

Now we explain how to reconstruct G from S' that is a candidate of $\tilde{S}$. Let $\mathcal{I}'$ be an interval representation of G_i whose corresponding end-vertex set is S'. Note that $\mathcal{I}'$ is easily obtained in $O(n+m)$ time by using the data structure MPQ-tree [10]. (Construct an MPQ-tree of G_i in linear time, and then, apply the operation (1) and (2) described in Section 3.2 to move the leaf corresponding to S' to the left end. From the MPQ-tree, we can easily obtain the corresponding compact interval representation.) Since $S \setminus \{s\} \subset \tilde{S}$ holds, $S \setminus \{s\} \subset S'$ holds, if S' is $\tilde{S}$. Hence, we can obtain an interval representation of G by extending intervals corresponding to vertices in $S \setminus \{s\} \subset S'$ to the left by one and adding an interval $[-1, -1]$. However, how to specify $S \setminus \{s\} \subset S'$? If we know $S \setminus \{s\}$, we can obtain G, since G has the interval representation obtained from $\tilde{\mathcal{I}}$ by extending intervals corresponding to vertices in $S \setminus \{s\}$ to the left by one and adding an interval $[-1, -1]$. In order to specify $S \setminus \{s\}$ in the polynomial time, we show the following lemma.

```
for each $G_i (i = 1, 2, \ldots, n)$ {
  for each end-vertex set $S'$ of $G_i$ {

    Let $\mathcal{I}'$ be an interval representation of $G_i$
     whose corresponding end-vertex set is $S'$.

    Compute the degree sequence $(d_1, \ldots, d_l)$ of $S \setminus \{s\}$.
    Let $S^*$ be a subset of $S'$ whose degree sequence in $G_i$ is $(d_1 - 1, \ldots, d_l - 1)$.
    If there does not exist such $S^*$, go to the next iteration.
    Let $G^*$ be an interval graph
     whose interval representation is obtained from $\mathcal{I}'$
     by extending interval corresponding to vertices in $S^*$ to the left by one
     and adding an interval $[-1, -1]$.

    if deck-checking($G^*$) = True
      output $G^*$.
  }
}
return No if the algorithm has output no graph.
```

Fig. 6. The algorithm for reconstructing connected interval graphs.

Lemma 14. *Let G be an interval graph. Let S be an end-vertex set of G, and let $\mathcal{I}$ be an compact interval representation of G whose corresponding end-vertex set is S. Let S_1 and S_2 be subsets of S such that the degree sequence of vertices in S_1 and the degree sequence of vertices in S_2 are the same. Let G_1 be a graph whose interval representation is obtained from $\mathcal{I}$ by extending interval corresponding to S_1 to the left by one and adding an interval $[-1, -1]$, and let G_2 be a graph whose interval representation is obtained by $\mathcal{I}$ extending interval corresponding to S_2 to the left by one and adding an interval $[-1, -1]$. Then, G_1 is isomorphic to G_2.*

Proof. The neighbor sets of S_1 and S_2 are the same due to Lemma 11. Hence, G_1 and G_2 are isomorphic. □

Since we know the degree sequence of G_i, and we can know the degree sequence of G by Lemma 13, we can know the degree sequence of $S \setminus \{s\}$. We denote the degree sequence by $(d_1, d_2, \ldots, d_l)$. Now we can specify $S \setminus \{s\} \subset S'$; $S \setminus \{s\}$ is the subset of S' such that whose degree sequence in G_i is $(d_1 - 1, d_2 - 1, \ldots, d_l - 1)$. Note that there may be exponentially many subsets of S' whose degree sequences in G_i are $(d_1 - 1, d_2 - 1, \ldots, d_l - 1)$. However, Lemma 14 guarantees that any of such subsets can be $S \setminus \{s\}$, i.e. all the graphs reconstructed under the assumption that some subset of S' whose degree sequence is $(d_1 - 1, d_2 - 1, \ldots, d_l - 1)$ are $S \setminus \{s\}$ are isomorphic to each other. Therefore, we can specify $S \setminus \{s\}$ in S'. To be more precise, if we can find such $S \setminus \{s\}$, S' is the desired $\tilde{S}$, and we can thus reconstruct G. The whole algorithm is described in Fig. 6.

Now we consider the time complexity of this algorithm. For each G_i, calculating an MPQ-tree of G_i in $O(n + m)$ time helps us to list each S' and $\mathcal{I}'$ in $O(n)$ time. Computing the degree sequence $(d_1, d_2, \ldots, d_l)$ takes $O(n \log n)$ time, since we can obtain $\{d_1, d_2, \ldots, d_l\}$ in $O(n)$ time from Lemma 13, and we can sort them in $O(n \log n)$ time. Since obtaining S^* needs also sorting, it requires $O(n \log n)$ time. It is clear that reconstructing an interval graph from its interval representation takes $O(n + m)$ time, if the endpoints of intervals are sorted. DECK CHECKING algorithm costs $O(n(n + m))$. Therefore, the total time complexity of this algorithm is $O(n((n+m)+n(n+m+n \log n+n(n+m)))) = O(n^3(n+m))$. Note that we have to check every output preimage is not isomorphic to each other for PREIMAGE COUNTING. Since the number of output preimage may be $O(n^2)$, we need $O(n^4(n + m))$ time for this checking. If the graph reconstruction conjecture is true, the time complexity of this checking can be omitted.

Theorem 15. *There is a polynomial time algorithm that lists up connected interval graphs that are preimages of the input n interval graphs. The time complexity for outputting one connected interval graph is $O(n^3(n + m))$, and that for outputting all is $O(n^4(n + m))$.*

4.2. Disconnected preimage case

Consider the case that the input graphs $G_1, G_2, \ldots, G_n$ have a disconnected preimage G. Then, from the argument in the Theorem 10, G must be an interval graph. Further, it is proven that the graph reconstruction conjecture is true in this case [7] (note that this fact does not imply that the reconstruction can be done in polynomial time). Lemma 8 and the fact that $\{G_1, G_2, \ldots, G_n\}$ is a deck of G if and only if $\{\tilde{G_1}, \tilde{G_2}, \ldots, \tilde{G_n}\} \cup \{G\}$ is a deck of $\tilde{G}$ simplify our algorithm in this case.

Since we can know the degree sequence of G by Lemma 13, we can know the degree sequence of $\tilde{G}$ by Lemma 13. Thus, we can obtain $\tilde{G}$ by the algorithm described in the previous subsection. Note that we do not know G, thus, in fact, we cannot use the algorithm itself. However, we can omit the case that G_i in the algorithm is G, since every interval graph has at least two end-vertex sets and so does $\tilde{G}$. Further, we can omit checking if G is in the deck of $\tilde{G}$. If the new algorithm (omitting checking if G is in the deck of $\tilde{G}$) returns some $\tilde{G}$, we can construct G from it. Then now we can check if G is a preimage of $G_1, \ldots, G_n$. Therefore, we have the theorem below.

Theorem 16. *There is a polynomial time algorithm that outputs a disconnected interval graph that is the preimage of the input n interval graphs, if there exists. The time complexity of the algorithm is* $O(n^3(n+m))$.

Therefore, we have the main theorem (Theorem 2) from Theorems 10, 15 and 16.

5. Concluding remarks

The algorithms which we described do not help directly the proof of the graph reconstruction conjecture on interval graphs. The conjecture on interval graphs remains to be open. Our proof shows that there are at most $\mathrm{O}(n^2)$ preimages for a deck of interval graphs. We hope that this fact will shed light on a proof of the graph reconstruction conjecture on interval graphs.

The complexities of graph reconstruction problems are strongly related to those of graph isomorphism problems. To develop a polynomial time algorithm for a graph reconstruction problem restricting inputs to be in GI-hard graph class seems very hard. Since graph isomorphisms of circular-arc graphs are polynomial time solvable, circular-arc graph reconstruction problem may be a good challenge. Recently, we developed a polynomial time reconstruction algorithm for permutation graphs [9].

References

[1] B. Bollobás, Almost every graph has reconstruction number three, Journal of Graph Theory 14 (1990) 1–4.
[2] J.A. Bondy, A graph reconstructor's manual, in: Surveys in Combinatorics, in: London Mathematical Society Lecture Note Series, vol. 166, 1991, pp. 221–252.
[3] K.S. Booth, G.S. Lueker, Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms, Journal of Computer and System Sciences 13 (1976) 335–379.
[4] D.R. Fulkerson, O.A. Gross, Incidence matrices and interval graphs, Pacific Journal of Mathematics 15 (1965) 835–855.
[5] F. Harary, A survey of the reconstruction conjecture, in: Graphs and Combinatorics, in: Lecture Notes in Mathematics, vol. 406, 1974, pp. 18–28.
[6] E. Hemaspaandra, L. Hemaspaandra, S. Radziszowski, R. Tripathi, Complexity results in graph reconstruction, Discrete Applied Mathematics 152 (2007) 103–118.
[7] P.J. Kelly, A congruence theorem for trees, Pacific Journal of Mathematics 7 (1957) 961–968.
[8] M. Kiyomi, T. Saitoh, R. Uehara, Reconstruction of interval graphs, in: COCOON 2009, in: Lecture Notes in Computer Science, vol. 5609, 2009, pp. 106–115.
[9] M. Kiyomi, T. Saitoh, R. Uehara, Reconstruction algorithm for permutation graphs, in: WALCOM 2010, in: Lecture Notes in Computer Science, vol. 5942, 2010, pp. 125–135.
[10] N. Korte, R.H. Möhring, An incremental linear-time algorithm for recognizing interval graphs, SIAM Journal on Computing 18 (1989) 68–81.
[11] D. Kratsch, L.A. Hemaspaandra, On the complexity of graph reconstruction, Mathematical Systems Theory 27 (1994) 257–273.
[12] C.G. Leckerkerker, J.Ch. Boland, Representation of a finite graph by a set of intervals on the real line, Fundamenta Mathematicae 51 (1962) 45–64.
[13] G.S. Lueker, K.S. Booth, A linear time algorithm for deciding interval graph isomorphism, Journal of the ACM 26 (1979) 183–195.
[14] M. von Rimscha, Reconstructibility and perfect graphs, Discrete Mathematics 47 (1983) 283–291.
[15] R. Uehara, Y. Uno, On computing longest paths in small graph classes, International Journal of Foundation of Computer Science 18 (2007) 911–930.