Polynomial-Time Algorithms for Subgraph Isomorphism in Small Graph Classes of Perfect Graphs*

Matsuo Konagaya[1,a)] Yota Otachi[1,b)] Ryuhei Uehara[1,c)]

Abstract: Given two graphs, Subgraph Isomorphism is the problem of deciding whether the first graph (the *base graph*) contains a subgraph isomorphic to the second graph (the *pattern graph*). This problem is NP-complete for very restricted graph classes such as connected proper interval graphs. Only a few cases are known to be polynomial-time solvable even if we restrict the graphs to be perfect. For example, if both graphs are co-chain graphs, then the problem can be solved in linear time.
In this paper, we present a polynomial-time algorithm for the case where the base graphs are chordal graphs and the pattern graphs are co-chain graphs. We also present a linear-time algorithm for the case where the base graphs are trivially perfect graphs and the pattern graphs are threshold graphs. These results answer some of the open questions of Kijima et al. [*Discrete Math.* 312, pp. 3164–3173, 2012]. To present a complexity contrast, we then show that even if the base graphs are somewhat restricted perfect graphs, the problem of finding a pattern graph that is a chain graph, a co-chain graph, or a threshold graph is NP-complete.

1. Introduction

The problem Subgraph Isomorphism is a very general and extremely hard problem which asks, given two graphs, whether one graph (the *base graph*) contains a subgraph isomorphic to the other graph (the *pattern graph*). The problem generalizes many other problems such as Graph Isomorphism, Hamiltonian Path, Clique, and Bandwidth. Clearly, Subgraph Isomorphism is NP-complete in general. Furthermore, by slightly modifying known proofs [5], [8], it can be shown that Subgraph Isomorphism is NP-complete when G and H are disjoint unions of paths or of complete graphs. Therefore, it is NP-complete even for small graph classes of perfect graphs such as proper interval graphs, bipartite permutation graphs, and trivially perfect graphs, while Graph Isomorphism can be solved in polynomial time for them [4], [18]. For these graph classes, Kijima et al. [16] showed that even if both input graphs are connected and have the same number of vertices, the problem remains NP-complete. They call the problem with such restrictions Spanning Subgraph Isomorphism.

Kijima et al. [16] also found polynomial-time solvable cases of Subgraph Isomorphism in which both graphs are chain, co-chain, or threshold graphs. Since these classes are proper subclasses of the aforementioned hard classes, those results together give sharp contrasts of computational complexity of Subgraph Isomorphism. However, the complexity of more subtle cases, like the one where the base graphs are proper interval graphs and the pattern graphs are co-chain graphs, remained open.

1.1 Our results

In this paper, we study the open cases of Kijima et al. [16], and present polynomial-time algorithms for the following cases:

- the base graphs are chordal graphs and the pattern graphs are co-chain graphs,
- the base graphs are trivially perfect graphs and the pattern graphs are threshold graphs.

We also show that even if the pattern graphs are chain, co-chain, or threshold graphs and the base graphs are somewhat restricted perfect graphs, the problem remains NP-complete. The problem of finding a chain subgraph in a bipartite permutation graph, which is an open case of Kijima et al. [16], remains unsettled. See Tables 1 and 2 for the summary of our results.

1.2 Related results

Subgraph Isomorphism for trees can be solved in polynomial time [22], while it is NP-complete for connected outerplanar graphs [26]. Therefore, the problem is NP-complete even for connected graphs of bounded treewidth.

*Partially supported by JSPS KAKENHI Grant Numbers 23500013, 25730003, and by MEXT KAKENHI Grant Number 24106004.
[1] School of Information Science, Japan Advanced Institute of Science and Technology
[a)] matsu.cona@jaist.ac.jp
[b)] otachi@jaist.ac.jp
[c)] uehara@jaist.ac.jp

Table 1 NP-complete cases of Spanning Subgraph Isomorphism.

Base	Pattern	Complexity	Reference
Bipartite Permutation		NP-complete	[16]
Proper Interval		NP-complete	[16]
Trivially Perfect		NP-complete	[16]
Chain	Convex	NP-complete	[16]
Co-chain	Co-bipartite	NP-complete	[16]
Threshold	Split	NP-complete	[16]
Bipartite	Chain	NP-complete	This paper
Co-convex	Co-chain	NP-complete	This paper
Split	Threshold	NP-complete	This paper

Table 2 Polynomial-time solvable cases of Subgraph Isomorphism.

Base	Pattern	Complexity	Reference
Chain		$O(m+n)$	[16]
Co-chain		$O(m+n)$	[16]
Threshold		$O(m+n)$	[16]
Bipartite permutation	Chain	Open	
Chordal	Co-chain	$O(mn^2+n^3)$	This paper
Trivially perfect	Threshold	$O(m+n)$	This paper

On the other hand, it can be solved in polynomial time for 2-connected outerplanar graphs [17]. More generally, it is known that Subgraph Isomorphism for k-connected partial k-trees can be solved in polynomial time [11], [21]. Eppstein [7] gave a $k^{O(k)}n$-time algorithm for Subgraph Isomorphism on planar graphs, where k and n are the numbers of the vertices in the pattern graph and the base graph, respectively. Recently, Dorn [6] has improved the running time to $2^{O(k)}n$. For other general frameworks, especially for the parameterized ones, see the recent paper by Marx and Pilipczuk [19] and the references therein.

Another related problem is Induced Subgraph Isomorphism which asks whether the base graph has an induced subgraph isomorphic to the pattern graph. Damaschke [5] showed that Induced Subgraph Isomorphism on cographs is NP-complete. He also showed that Induced Subgraph Isomorphism is NP-complete for the disjoint unions of paths, and thus for proper interval graphs and bipartite permutation graphs. Marx and Schlotter [20] showed that Induced Subgraph Isomorphism on interval graphs is W[1]-hard when parameterized by the number of vertices in the pattern graph, but fixed-parameter tractable when parameterized by the numbers of vertices to be removed from the base graph. Heggernes et al. [14] showed that Induced Subgraph Isomorphism on proper interval graphs is NP-complete even if the base graph is connected. Heggernes et al. [15] have recently shown that Induced Subgraph Isomorphism on proper interval graphs and bipartite permutation graphs can be solved in polynomial time if the pattern graph is connected. Belmonte et al. [1] showed that Induced Subgraph Isomorphism on connected trivially perfect graphs is NP-complete. This result strengthens known results since every trivially perfect graph is an interval cograph. They also showed that the problem can be solved in polynomial time if the base graphs are trivially perfect graphs and the pattern graphs are threshold graphs.

2. Preliminaries

All graphs in this paper are finite, undirected, and simple. Let $G[U]$ denote the subgraph of $G = (V, E)$ induced by $U \subseteq V$. For a vertex $v \in V$, we denote by $G - v$ the graph obtained by removing v from G; that is, $G - v = G[V \setminus \{v\}]$. The *neighborhood* of a vertex v is the set $N(v) = \{u \in V \mid \{u, v\} \in E\}$. A vertex $v \in V$ is *universal* in G if $N(v) = V \setminus \{v\}$. A vertex $v \in V$ is *isolated* in G if $N(v) = \emptyset$. A set $I \subseteq V$ in $G = (V, E)$ is an *independent set* if for all $u, v \in I$, $(u, v) \notin E$. A set $S \subseteq V$ in $G = (V, E)$ is a *clique* if for all $u, v \in S$, $(u, v) \in E$. A pair (X, Y) of sets of vertices of a bipartite graph $H = (U, V; E)$ is a *biclique* if for all $x \in X$ and $y \in Y$, $(x, y) \in E$. A *component* of a graph G is an inclusion maximal connected subgraph of G. A component is *non-trivial* if it contains at least two vertices. The *complement* of a graph $G = (V, E)$ is the graph $\bar{G} = (V, \bar{E})$ such that $\{u, v\} \in \bar{E}$ if and only if $\{u, v\} \notin E$. The *disjoint union* of two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ is the graph $(V_G \cup V_H, E_G \cup E_H)$, where $V_G \cap V_H = \emptyset$. For a map $\eta\colon V \to V'$ and $S \subseteq V$, let $\eta(S)$ denote the set $\{\eta(s) \mid s \in S\}$.

2.1 Definitions of the problems

A graph $H = (V_H, E_H)$ is *subgraph-isomorphic* to a graph $G = (V_G, E_G)$ if there exists an injective map η from V_H to V_G such that $\{\eta(u), \eta(v)\} \in E_G$ holds for each $\{u, v\} \in E_H$. We call such a map η a *subgraph-isomorphism* from H to G. Graphs G and H are called the *base graph* and the *pattern graph*, respectively. The problems Subgraph Isomorphism and Spanning Subgraph Isomorphism are defined as follows:

Problem 2.1 Subgraph Isomorphism
Instance: A pair of graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$.
Question: Is H subgraph-isomorphic to G?

Problem 2.2 Spanning Subgraph Isomorphism
Instance: A pair of connected graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$, where $|V_G| = |V_H|$.
Question: Is H subgraph-isomorphic to G?

2.2 Graph classes

Here we introduce the graph classes we deal with in this paper. For their inclusion relations, see the standard textbooks in this field [3], [9], [25]. See Fig. 1 for the class hierarchy.

A bipartite graph $B = (X, Y; E)$ is a *chain* graph if the vertices of X can be ordered as $x_1, x_2, \ldots, x_{|X|}$ such that $N(x_1) \subseteq N(x_2) \subseteq \cdots \subseteq N(x_{|X|})$. A graph $G = (V, E)$ with $V = \{1, 2, \ldots, n\}$ is a *permutation graph* if there is a permutation π over V such that $\{i, j\} \in E$ if and only if $(i - j)(\pi(i) - \pi(j)) < 0$. A *bipartite permutation graph* is a permutation graph that is bipartite. A bipartite graph $H = (X, Y; E)$ is a *convex* graph if one of X and Y can be ordered such that the neighborhood of each vertex in the

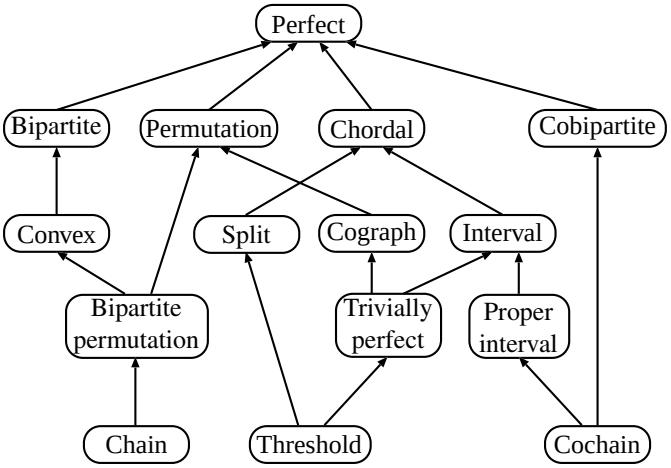

Fig. 1 Graph classes.

other side is consecutive in the ordering. It is known that a chain graph is a bipartite permutation graph, and that a bipartite permutation graph is a convex graph.

A graph is a *co-chain* graph if it is the complement of a chain graph. An *interval graph* is the intersection graph of a family of closed intervals of the real line. A *proper interval graph* is the intersection graph of a family of closed intervals of the real line where no interval is properly contained in another. A graph is *co-bipartite* if its complement is bipartite. In other words, co-bipartite graphs are exactly the graphs whose vertex sets can be partitioned into two cliques. From the definition, every co-chain graph is co-bipartite. It is known that every co-chain graph is a proper interval graph.

A graph is a *threshold* graph if there is a positive integer T (the *threshold*) and for every vertex v there is a positive integer $w(v)$ such that $\{u, v\}$ is an edge if and only if $w(u) + w(v) \geq T$. A graph is *trivially perfect* if the size of the maximum independent set is equal to the number of maximal cliques for every induced subgraph. It is known that a threshold graph is a trivially perfect graph, and that a trivially perfect graph is an interval graph.

A *split graph* is a graph whose vertex set can be partitioned into a clique and an independent set. A graph is chordal if every induced cycle is of length 3. Clearly, every threshold graph is a split graph, and every split graph is a chordal graph. It is known that every interval graph is a chordal graph. It is easy to see that any split graph (and thus any threshold graph) has at most one non-trivial component.

A graph is *perfect* if for any induced subgraph the chromatic number is equal to the size of a maximum clique. Graphs in all classes introduced in this section are known to be perfect.

3. Polynomial-Time Algorithms

In this section, we denote the number of the vertices and the edges in a base graph by n and m, respectively. For the input graphs G and H, we assume that $|V_G| \geq |V_H|$ and $|E_G| \geq |E_H|$, which can be checked in time $O(m + n)$.

3.1 Finding co-chain subgraphs in chordal graphs

It is known that co-chain graphs are precisely $\{I_3, C_4, C_5\}$-free graphs [13]; that is, graphs having no vertex subset that induces I_3, C_4, or C_5, where I_3 is the empty graph with three vertices and C_k is the cycle of k vertices. Using this characterization, we can show the following simple lemma.

Lemma 3.1 A graph is a co-chain graph if and only if it is a co-bipartite chordal graph.

Proof. To prove the if-part, let G be a co-bipartite chordal graph. Since G is co-bipartite, it cannot have I_3 as its induced subgraph. Since G is chordal, it does not have C_4 or C_5 as its induced subgraph. Therefore, G is $\{I_3, C_4, C_5\}$-free.

To prove the only-if-part, let G be a co-chain graph, and thus it is a co-bipartite graph. Suppose that G has an induced cycle C of length $k \geq 4$. Then k cannot be 4 or 5 since it does not have C_4 or C_5. If $k \geq 6$, then the first, third, and fifth vertices in the cycle form I_3.

Now we can solve the problem as follows.

Theorem 3.2 Subgraph Isomorphism is solvable in $O(mn^2 + n^3)$ time if the base graphs are chordal graphs and the pattern graphs are co-chain graphs.

Proof. Let $G = (V_G, E_G)$ be the base chordal graph and $H = (V_H, E_H)$ be the pattern co-chain graph. We assume that G is not complete, since otherwise the problem is trivial.

Algorithm: We enumerate all the maximal cliques $C_1, \ldots, C_k$ of G. For each pair (C_i, C_j), we check whether H is subgraph-isomorphic to $G[C_i \cup C_j]$. If H is subgraph-isomorphic to $G[C_i \cup C_j]$ for some i and j, then we output "yes." Otherwise, we output "no."

Correctness: It suffices to show that H is subgraph-isomorphic to G if and only if there are two maximal cliques C_i and C_j of G such that H is subgraph-isomorphic to $G[C_i \cup C_j]$. The if-part is obviously true. To prove the only-if-part, assume that there is a subgraph-isomorphism η from H to G. Observe that for any clique C of H, there is a maximal clique C' of G such that $\eta(C) \subseteq C'$. Thus, since H is co-bipartite, there are two maximal cliques C_i and C_j such that $\eta(V_H) \subseteq C_i \cup C_j$. That is, H is subgraph-isomorphic to $G[C_i \cup C_j]$.

Running time: It is known that a chordal graph of n vertices with m edges has at most n maximal cliques, and all the maximal cliques can be found in $O(m+n)$ time [2], [12]. Since $G[C_i \cup C_j]$ is a co-chain graph by Lemma 3.1, testing whether H is subgraph-isomorphic to $G[C_i \cup C_j]$ can be done in $O(m+n)$ time [16]. Since the number of pairs of maximal cliques is $O(n^2)$, the total running time is $O(mn^2 + n^3)$.

3.2 Finding threshold subgraphs in trivially perfect graphs

Here we present a linear-time algorithm for finding a threshold subgraph in a trivially perfect graph. To this end, we need the following lemmas.

Lemma 3.3 If a graph G has a universal vertex u_G, and a graph H has a universal vertex u_H, then H is subgraph-isomorphic to G if and only if $H-u_H$ is subgraph-isomorphic to $G-u_G$.

Proof. To prove the if-part, let η' be a subgraph-isomorphism from $H-u_H$ to $G-u_G$. Now we define $\eta\colon V_H \to V_G$ as follows:

$$\eta(w) = \begin{cases} u_G & \text{if } w = u_H, \\ \eta'(w) & \text{otherwise.} \end{cases}$$

Let $\{x,y\} \in E_H$. If $u_H \notin \{x,y\}$, then $\{\eta(x),\eta(y)\} = \{\eta'(x),\eta'(y)\} \in E_G$. Otherwise, we may assume that $x = u_H$ without loss of generality. Since u_G is universal in G, it follows that $\{\eta(x),\eta(y)\} = \{\eta(u_H),\eta(y)\} = \{u_G,\eta'(y)\} \in E_G$.

To prove the only-if-part, assume that η' is a subgraph-isomorphism from H to G. If there is no vertex $v \in V_H$ such that $\eta'(v) = u_G$, then we are done. Assume that $\eta'(v) = u_G$ for some vertex $v \in V_H$. Now we define $\eta\colon V_H \setminus \{u_H\} \to V_G \setminus \{u_G\}$ as follows:

$$\eta(w) = \begin{cases} \eta'(u_H) & \text{if } w = v, \\ \eta'(w) & \text{otherwise.} \end{cases}$$

Let $\{x,y\} \in E_H$. If $v \notin \{x,y\}$, then $\{\eta(x),\eta(y)\} = \{\eta'(x),\eta'(y)\} \in E_G$. Otherwise, we may assume without loss of generality that $v = x$. Since u_H is universal in H, it follows that $\{\eta(x),\eta(y)\} = \{\eta'(u_H),\eta'(y)\} \in E_G$.

A component of a graph is *maximum* if it contains the maximum number of vertices among all the components of the graph. If a split graph has a non-trivial component, then the component is the unique maximum component of the graph.

Lemma 3.4 A split graph H with a maximum component C_H is subgraph-isomorphic to a graph G if and only if $|V_H| \le |V_G|$ and there is a component C_G of G such that C_H is subgraph-isomorphic to C_G.

Proof. First we prove the only-if-part. Let η be a subgraph-isomorphism from H to G. We need $|V_H| \le |V_G|$ to have an injective map from V_H to V_G. Since C_H is connected, $G[\eta(V(C_H))]$ must be connected. Thus there is a component C_G such that $\eta(V(C_H)) \subseteq V(C_G)$. Then $\eta|_{V(C_H)}$, the map η restricted to $V(C_H)$, is a subgraph isomorphism from C_H to C_G.

To prove the if-part, let η' be a subgraph-isomorphism from C_H to C_G. Let $R_H = V_H \setminus V(C_H) = \{u_1,\dots,u_r\}$, and let $R_G = V_G \setminus \eta'(V(C_H)) = \{w_1,\dots,w_s\}$. Since $|V_H| \le |V_G|$ and $|V(C_H)| = |\eta'(V(C_H))|$, it holds that $r \le s$. Now we define $\eta\colon V_H \to V_G$ as follows:

$$\eta(v) = \begin{cases} w_i & \text{if } v = u_i \in R_H, \\ \eta'(v) & \text{otherwise.} \end{cases}$$

Since H is a split graph, any component of H other than C_H cannot have two or more vertices. Thus the vertices in R_H are isolated in H. Therefore, the map η is a subgraph-isomorphism from H to G.

The two lemmas above already allows us to have a polynomial-time algorithm. However, to achieve a linear running time, we need the following characterization of trivially perfect graphs.

A *rooted tree* is a directed tree with a unique in-degree 0 vertex, called the *root*. Intuitively, every edge is directed from the root to leaves in a directed tree. A *rooted forest* is the disjoint union of rooted trees. The *comparability graph* of a rooted forest is the graph that has the same vertex set as the rooted forest, and two vertices are adjacent in the graph if and only if one of the two is a descendant of the other in the forest. Yan et al. [28] showed that a graph is a trivially perfect graph if and only if it is the comparability graph of a rooted forest, and that such a rooted forest can be computed in linear time. We call such a rooted forest a *generating forest* of the trivially perfect graph. If a generating forest is actually a rooted tree, then we call it a *generating tree*.

Theorem 3.5 Subgraph Isomorphism is solvable in $O(m+n)$ time if the base graphs are trivially perfect graphs and the pattern graphs are threshold graphs.

Proof. Let $G = (V_G, E_G)$ be the base trivially perfect graph and $H = (V_H, E_H)$ be the pattern threshold graph.
Algorithm: The pseudocode of our algorithm can be found in Algorithm 1. We use the procedure SGI which takes a trivially perfect graph as the base graph and a threshold graph as the pattern graph, and conditionally answers whether the pattern graph is subgraph-isomorphic to the base graph. The procedure SGI requires that

- both the graphs are connected, and
- the base graph has at least as many vertices as the pattern graph.

To use this procedure, we first attach a universal vertex to both G and H. This guarantees that both graphs are connected. We call the new graphs G' and H', respectively. By Lemma 3.3, (G', H') is a yes-instance if and only if so is (G, H). After checking that $|V_{G'}| \ge |V_{H'}|$, we use the procedure SGI.

In SGI(G, H), let u_G and u_H be universal vertices of G and H, respectively. There are such vertices since G and H are connected trivially perfect graphs [27]. Let C_H be a maximum component of $H - u_H$. For each connected component C_G of $G - u_G$, we check whether C_H is subgraph-isomorphic to C_G, by recursively calling the procedure SGI itself. If at least one of the recursive calls returns "yes," then we return "yes." Otherwise we return "no."
Correctness: It suffices to prove the correctness of the procedure SGI. If $|V_H| = 1$, then H is subgraph-isomorphic to G since $|V_G| \ge |V_H|$ in SGI. By Lemmas 3.3 and 3.4, H is subgraph-isomorphic to G if and only if there is a component C_G of $G - u_G$ such that C_H is subgraph-isomorphic to C_G. (Recall that any threshold graph is a split graph.) The procedure just checks these conditions. Also, when SGI

Algorithm 1 Finding a threshold subgraph H in a trivially perfect graph G.

```
G' := G with a universal vertex
H' := H with a universal vertex
if |V_{G'}| ≥ |V_{H'}| then
    return SGI(G', H')
else
    return no

Require: G and H are connected, and |V_G| ≥ |V_H|
procedure SGI(G, H)
    if |V_H| = 1 then
        return yes
    u_G := a universal vertex of G
    u_H := a universal vertex of H
    C_H := a maximum component of H − u_H
    for all components C_G of G − u_G do
        if |V(C_G)| ≥ |V(H − u_H)| then
            if SGI(C_G, C_H) = yes then
                return yes
    return no
```

recursively calls itself, the parameters C_G and C_H satisfy its requirements; that is, C_G and C_H are connected, and $|V(C_G)| \geq |V(C_G)|$.

Running time: For each call of $\mathrm{SGI}(G, H)$, we need the following:

- universal vertices u_G and u_H of G and H, respectively,
- a maximum component C_H of $H - u_H$,
- the components C_G of $G - u_G$, and
- the numbers of the vertices of C_G and $H - u_H$.

We show that they can be computed efficiently by using generating forests. Basically we apply the algorithm to generating forests instead of graphs.

Before the very first call of $\mathrm{SGI}(G, H)$, we compute generating trees of G and H in linear time. Additionally, for each node in the generating trees, we store the number of its descendants. This can be done also in linear time in a bottom-up fashion.

At some call of $\mathrm{SGI}(G, H)$, assume that we have generating trees of G and H. It is easy to see that the root of the generating trees are universal vertices. Hence we can compute u_G and u_H in constant time. By removing these root nodes from the generating trees, we obtain generating forests of $G - u_G$ and $H - u_H$. Each component of the generating forests corresponds to a component of the corresponding graphs. Thus we can compute the components of $G - u_G$ and a maximum component of $H - u_H$, with their generating trees, in time proportional to the number of the children of u_G and u_H. The numbers of the vertices of C_G and $H - u_H$ can be computed easily in constant time, because we know the number of the descendants of each node in generating trees.

The recursive calls of SGI take only $O(n)$ time in total since it is proportional to the number of edges in the generating trees. Therefore, the total running time is $O(m + n)$.

4. NP-completeness

It is known that for perfect graphs, Clique can be solved in polynomial time [10]. Since co-chain graphs and threshold graphs are very close to complete graphs, one may ask whether the problem of finding co-chain graphs or threshold graphs can be solved in polynomial time for perfect graphs. In this section, we show that this is not the case. More precisely, we show that even the specialized problem Spanning Subgraph Isomorphism is NP-complete for the case where the base graphs are somewhat restricted perfect graphs and the pattern graphs are co-chain or threshold graphs.

It is known that Maximum Edge Biclique, the problem of finding a biclique with the maximum number of edges, is NP-complete for bipartite graphs [24]. This implies that Subgraph Isomorphism is NP-complete if the base graphs are connected bipartite graphs and the pattern graphs are connected chain graphs, because complete bipartite graphs are chain graphs. We sharpen this hardness result by showing that the problem is still NP-complete if we further restrict the pattern chain graphs to have the same number of vertices as the base graph. That is, we show that Spanning Subgraph Isomorphism is NP-complete when the base graphs are bipartite graphs and the pattern graphs are chain graphs.

Since the problem Spanning Subgraph Isomorphism is clearly in NP for any graph class, we only show its NP-hardness here. All the results in this section are based on the following theorem and lemma taken from Kijima et al. [16].

Theorem 4.1 (Kijima et al. [16]) Spanning Subgraph Isomorphism is NP-complete if

(1) the base graphs are chain graphs and the pattern graphs are convex graphs,
(2) the base graphs are co-chain graphs and the pattern graphs are co-bipartite graphs, or
(3) the base graphs are threshold graphs and the pattern graphs are split graphs.

Lemma 4.2 (Kijima et al. [16]) If $|V_H| = |V_G|$, then H is subgraph-isomorphic to G if and only if $\bar{G}$ is subgraph-isomorphic to $\bar{H}$.

For a graph class $\mathcal{C}$, let co-$\mathcal{C}$ denote the graph class $\{\bar{G} \mid G \in \mathcal{C}\}$. The next lemma basically shows that if $\mathcal{C}$ satisfies some property, then the hardness of Spanning Subgraph Isomorphism for $\mathcal{C}$ implies the hardness for co-$\mathcal{C}$.

Lemma 4.3 Let $\mathcal{C}$ and $\mathcal{D}$ be graph classes such that co-$\mathcal{C}$ and co-$\mathcal{D}$ are closed under universal vertex additions. If Spanning Subgraph Isomorphism is NP-complete when the base graphs belong to $\mathcal{C}$ and the pattern graphs belong to $\mathcal{D}$, then the problem is NP-complete also when the base graphs belong to co-$\mathcal{D}$ and the pattern graphs belong to co-$\mathcal{C}$.

Proof. Given two connected graphs $G \in \mathcal{C}$ and $H \in \mathcal{D}$ with $|V_G| = |V_H|$, it is NP-complete to decide whether H is subgraph-isomorphic to G. By Lemma 4.2, H is subgraph-isomorphic to G if and only if $\bar{G}$ is subgraph-isomorphic

to $\bar{H}$. By Lemma 3.3, $\bar{G}$ is subgraph-isomorphic to $\bar{H}$ if and only if $\bar{G}'$ is subgraph-isomorphic to $\bar{H}'$, where $\bar{G}'$ and $\bar{H}'$ are obtained from $\bar{G}$ and $\bar{H}$, respectively, by adding a universal vertex. Therefore, H is subgraph-isomorphic to G if and only if $\bar{G}'$ is subgraph-isomorphic to $\bar{H}'$. Clearly, $\bar{G}' \in$ co-$\mathcal{C}$ and $\bar{H}' \in$ co-$\mathcal{D}$, they are connected, and they have the same number of vertices. Thus the lemma holds.

A graph is a *co-convex* graph if its complement is a convex graph. Clearly co-convex graphs are closed under additions of universal vertices.

Corollary 4.4 Spanning Subgraph Isomorphism is NP-complete if

(1) the base graphs are co-convex graphs and the pattern graphs are co-chain graphs,
(2) the base graphs are bipartite graphs and the pattern graphs are chain graphs, or
(3) the base graphs are split graphs and the pattern graphs are threshold graphs.

Proof. The NP-completeness of the case (1) is a corollary to Theorem 4.1 (1) and Lemma 4.3. To prove (3), we need Theorem 4.1 (3), Lemma 4.3, and the well-known facts that threshold graphs and split graphs are self-complementary [9]. That is, the complement of a threshold graph is a threshold graph, and the complement of a split graph is a split graph.

For (2), we cannot directly apply the combination of Theorem 4.1 (2) and Lemma 4.3 since bipartite graphs and chain graphs are not closed under universal vertex additions. Fortunately, we can easily modify the proof of Theorem 4.1 (2) in Kijima et al. [16] so that the complements of the base graphs and the pattern graphs are also connected. Then, Lemma 4.2 implies the statement. Since it will be a repeat of a known proof with a tiny difference, we omit the detail.

5. Conclusion

We have studied (Spanning) Subgraph Isomorphism for classes of perfect graphs, and have shown sharp contrasts of its computational complexity. An interesting problem left unsettled is the complexity of Subgraph Isomorphism where the base graphs are bipartite permutation graphs and the pattern graphs are chain graphs. It is known that although the maximum edge biclique problem is NP-complete for general bipartite graphs [24], it can be solved in polynomial time for some super classes of bipartite permutation graphs (see [23]). Therefore, it might be possible to have a polynomial-time algorithm for Subgraph Isomorphism when the pattern graphs are chain graphs and the base graphs belong to an even larger class like convex graphs.

References

[1] R. Belmonte, P. Heggernes, and P. van 't Hof. Edge contractions in subclasses of chordal graphs. *Discrete Appl. Math.*, 160:999–1010, 2012.
[2] J. R. S. Blair and B. Peyton. An introduction to chordal graphs and clique trees. In A. George, J. R. Gilbert, and J. W. H. Liu, editors, *Graph Theory and Sparse Matrix Computation*, volume 56 of *The IMA Volumes in Mathematics and its Applications*, pages 1–29. Springer Verlag, 1993.
[3] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: A Survey*. SIAM, 1999.
[4] C. J. Colbourn. On testing isomorphism of permutation graphs. *Networks*, 11:13–21, 1981.
[5] P. Damaschke. Induced subgraph isomorphism for cographs is NP-complete. In *WG '90*, volume 487 of *Lecture Notes in Comput. Sci.*, pages 72–78, 1991.
[6] F. Dorn. Planar subgraph isomorphism revisited. In *STACS 2010*, volume 5 of *LIPIcs*, pages 263–274, 2010.
[7] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *J. Graph Algorithms Appl.*, 3:1–27, 1999.
[8] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.
[9] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*, volume 57 of *Annals of Discrete Mathematics*. North Holland, second edition, 2004.
[10] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.
[11] A. Gupta and N. Nishimura. The complexity of subgraph isomorphism for classes of partial k-trees. *Theoret. Comput. Sci.*, 164:287–298, 1996.
[12] P. Heggernes. Treewidth, partial k-trees, and chordal graphs. Partial curriculum in INF334 - Advanced algorithmical techniques, Department of Informatics, University of Bergen, Norway, 2005.
[13] P. Heggernes and D. Kratsch. Linear-time certifying recognition algorithms and forbidden induced subgraphs. *Nordic J. Comput.*, 14:87–108, 2007.
[14] P. Heggernes, D. Meister, and Y. Villanger. Induced subgraph isomorphism on interval and proper interval graphs. In *ISAAC 2010*, volume 6507 of *Lecture Notes in Comput. Sci.*, pages 399–409, 2010.
[15] P. Heggernes, P. van 't Hof, D. Meister, and Y. Villanger. Induced subgraph isomorphism on proper interval and bipartite permutation graphs. Submitted manuscript.
[16] S. Kijima, Y. Otachi, T. Saitoh, and T. Uno. Subgraph isomorphism in graph classes. *Discrete Math.*, 312:3164–3173, 2012.
[17] A. Lingas. Subgraph isomorphism for biconnected outerplanar graphs in cubic time. *Theoret. Comput. Sci.*, 63:295–302, 1989.
[18] G. S. Lueker and K. S. Booth. A linear time algorithm for deciding interval graph isomorphism. *J. ACM*, 26:183–195, 1979.
[19] D. Marx and M. Pilipczuk. Everything you always wanted to know about the parameterized complexity of subgraph isomorphism (but were afraid to ask). *CoRR*, abs/1307.2187, 2013.
[20] D. Marx and I. Schlotter. Cleaning interval graphs. *Algorithmica*, 65:275–316, 2013.
[21] J. Matoušek and R. Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Math.*, 108:343–364, 1992.
[22] D. W. Matula. Subtree isomorphism in $O(n^{5/2})$. In B. Alspach, P. Hell, and D. Miller, editors, *Algorithmic Aspects of Combinatorics*, volume 2 of *Annals of Discrete Mathematics*, pages 91–106. Elsevier, 1978.
[23] D. Nussbaum, S. Pu, J.-R. Sack, T. Uno, and H. Zarrabi-Zadeh. Finding maximum edge bicliques in convex bipartite graphs. *Algorithmica*, 64(2):311–325, 2012.
[24] R. Peeters. The maximum edge biclique problem is NP-complete. *Discrete Appl. Math.*, 131:651–654, 2003.
[25] J. P. Spinrad. *Efficient Graph Representations*, volume 19 of *Fields Institute monographs*. American Mathematical Society, 2003.
[26] M. M. Sysło. The subgraph isomorphism problem for outerplanar graphs. *Theoret. Comput. Sci.*, 17:91–97, 1982.
[27] E. S. Wolk. A note on "The comparability graph of a tree". *Proc. Amer. Math. Soc.*, 16:17–20, 1965.
[28] J.-H. Yan, J.-J. Chen, and G. J. Chang. Quasi-threshold graphs. *Discrete Appl. Math.*, 69(3):247–255, 1996.