IPSJ SIG Technical Report

Vol.2022-AL-186 No.7
2022/1/28

Algorithms for Happy Set Problem on Interval Graphs and Permutation Graphs

Hiroshi Eto[1,a)] Takehiro Ito[1,b)] Eiji Miyano[2,c)] Akira Suzuki[1,d)] Yuma Tamura[1,e)]

Abstract: In this technical report, we investigate the complexity of the Maximum Happy Set problem on subclasses of co-comparability graphs. For a graph G and its vertex subset S, a vertex $v \in S$ is happy if all v's neighbors in G are contained in S. Given a graph G and a non-negative integer k, Maximum Happy Set is the problem of finding a vertex subset S of G such that $|S| = k$ and the number of happy vertices in S is maximized. In this technical report, we first show that Maximum Happy Set is NP-hard even for co-bipartite graphs. We then give an algorithm for n-vertex interval graphs whose running time is $O(k^2n^2)$; this improves the best known running time $O(kn^8)$ for interval graphs. We also design an algorithm for n-vertex permutation graphs whose running time is $O(k^3n^2)$. These two algorithmic results provide a nice contrast to the fact that Maximum Happy Set remains NP-hard for chordal graphs, comparability graphs, and co-comparability graphs.

Keywords: Graph Algorithm, Happy Set Problem, Interval Graph, Permutation Graph

1. Introduction

Easley and Kleinberg [7] said that *homophily* is one of the most basic notions governing the structure of social networks. Homophily is the principle that we are likely to associate with people who are similar in characteristics, such as their ages, their occupations and their interests. Motivated from homophily of social networks, Zhang and Li [12] formulated two graph coloring problems, and recently Asahiro et al. [1] introduced another formulation on graphs. In this technical report, we study the latter formulation, defined as follows.

For a graph $G = (V, E)$ and a subset $S \subseteq V$, a vertex $v \in S$ is *happy* if all its neighbors in G are contained in S. Given an undirected graph $G = (V, E)$ and a non-negative integer k, Maximum Happy Set is the problem of finding a subset $S \subseteq V$ such that $|S| = k$ and the number of happy vertices in S is maximized.

1.1 Known results

Although Maximum Happy Set was proposed recently,[*1] it has been already studied from various viewpoints such as polynomial-time solvability, approximability, and fixed-parameter tractability.

Polynomial-time solvability: Maximum Happy Set is NP-hard even for bipartite graphs [2], cubic graphs [2], and split graphs [1]. On the other hand, the problem is solvable in $O(k^2n)$ time for block graphs [2], and solvable in $O(kn^8)$ time for interval graphs [2], where n is the number of vertices in a graph.

Approximability: Maximum Happy Set admits a polynomial-time approximation algorithm whose approximation ratio depends on the maximum degree of a graph [2].

Fixed-parameter tractability: Maximum Happy Set is W[1]-hard when parameterized by k even on split graphs [1], and hence it is very unlikely that the problem admits a fixed-parameter algorithm even when restricted to split graphs and parameterized by k. On the other hand, the problem admits fixed-parameter algorithms when parameterized by graph structural parameters such as tree-width, clique-width, neighborhood diversity, and twin-cover number of a graph [1].

1.2 Our contributions

In this technical report, we further investigate the polynomial-time solvability of Maximum Happy Set, by focusing on subclasses of co-comparability graphs. In particular, we consider co-bipartite graphs, interval graphs, and permutation graphs.

We first show that Maximum Happy Set is NP-hard even for co-bipartite graphs. As far as we know, this is the first intractability result of Maximum Happy Set on subclasses of co-comparability graphs. We thus need to focus on other subclasses of co-comparability graphs, in order to seek polynomial-time solvable cases, as below.

We then give a polynomial-time algorithm for interval graphs. Recall that the polynomial-time solvability for interval graphs is already known [2]. However, our algorithm runs in $O(k^2n^2)$ time for n-vertex interval graphs, which improves the best known run-

[1] Graduate School of Information Sciences, Tohoku University
[2] School of Computer Science and Systems Engineering, Kyushu Institute of Technology
[a)] hiroshi.eto.b4@tohoku.ac.jp
[b)] takehiro@tohoku.ac.jp
[c)] miyano@ai.kyutech.ac.jp
[d)] akira@dc.tohoku.ac.jp
[e)] tamura@tohoku.ac.jp

[*1] We note that the graph coloring problem introduced by Zhang and Li [12] is called a similar name, Maximum Happy Vertices, but it is a different problem from ours.

ning time $O(kn^8)$ [2].

We finally give an algorithm for n-vertex permutation graphs which runs in $O(k^3n^2)$ time. This is a new polynomial-time solvable case, and gives a nice contrast to the known fact that Maximum Happy Set is NP-hard for comparability graphs and co-comparability graphs. We note that if k is a constant, then both algorithms for interval graphs and permutation graphs run in $O(n^2)$ time.

In this technical report, we briefly summarize the complexity and the algorithmic results for Maximum Happy Set shown in [8]. See [8] for further details.

Technical highlight: Both our polynomial-time algorithms for interval graphs and permutation graphs employ basically the same technique, that is, a dynamic programming approach based on graph representation models. Details and formal definitions will be given later, but we here explain the key point. Given an n-vertex graph $G = (V, E)$, we define a subgraph $G_i = (V_i, E_i)$ for each integer $i = 1, 2, \ldots, n$, depending on a representation model for G. Then, we wish to compute a partial solution $S_i = S^* \cap V_i$ for each G_i, where S^* is an optimal solution of G. Note that S_i is not always optimal for G_i, and hence it is not enough to compute an optimal solution of G_i. The key of our algorithms is that partial solutions S_i of G_i can be characterized by only two vertices that are *not* contained in S^*, when G is an interval graph or a permutation graph. This efficient characterization of partial solutions leads to improving the running time for interval graphs.

1.3 Contrasts to related results

Our initial motivation was to develop a polynomial-time algorithm for Maximum Happy Set on co-comparability graphs, because it is known that several classical problems are tractable for co-comparability graphs even if they are NP-hard on perfect graphs. Such examples include Minimum Dominating Set [11], Hamiltonian Cycle [6], and Minimum Feedback Vertex Set [4]. Our result of NP-hardness for co-comparability graphs gives an interesting contrast to these complexity examples.

The Densest k-Subgraph problem [9], which has been studied for more than two decades in the field of graph theory, can be seen as an edge variant of Maximum Happy Set: given an undirected graph $G = (V, E)$ and a non-negative integer k, the task of the problem is to find a vertex subset $S \subseteq V$ of size exactly k such that the number of edges whose both endpoints are contained in S is maximized. Interestingly, the complexity of Densest k-Subgraph remains open for interval graphs, permutation graphs, and planar graphs. Although results on Maximum Happy Set cannot be converted directly to Densest k-Subgraph, our complexity results in this technical report may give new insights to Densest k-Subgraph.

2. Preliminaries

Let $G = (V, E)$ be a graph; we denote by $V(G)$ and $E(G)$ the vertex set and the edge set of G, respectively. We assume that all graphs in this technical report are simple, undirected, and unweighted. For a vertex v of G, we denote by $N_G(v)$ and $N_G[v]$ the open and closed neighborhood of v in G, respectively, that is, $N_G(v) = \{w \in V(G) : vw \in E(G)\}$ and $N_G[v] = N_G(v) \cup \{v\}$. For a vertex subset $V' \subseteq V$, we denote by $G - V'$ the subgraph of G obtained by deleting all the vertices in V' and their incident edges. We shall often write $G - v$ instead of $G - \{v\}$ for a vertex $v \in V$.

For a graph $G = (V, E)$ and its vertex subset $S \subseteq V$, we say that a vertex $v \in V$ is *happy with respect to S on G* if $N_G[v] \subseteq S$; otherwise v is *unhappy with respect to S on G*. We denote by $H(G; S)$ the set of happy vertices with respect to S on G. We note that $H(G; \emptyset) = \emptyset$. Given a graph $G = (V, E)$ and a non-negative integer k, Maximum Happy Set is the problem of finding a vertex subset $S \subseteq V$ such that $|S| = k$ and the size of $H(G; S)$ is maximized. For simplicity, our algorithms in this technical report only compute the maximum value of $|H(G; S)|$. However, one can easily modify the algorithms so that they find an actual subset S in the same time complexity.

3. NP-hardness for co-bipartite graphs

A graph is *co-bipartite* if it is the complement of a bipartite graph. In other words, a co-bipartite graph is a graph whose vertex set can be partitioned into two cliques. We give the following hardness result, whose proof is omitted from this technical report.

Theorem 1. Maximum Happy Set *is NP-hard for co-bipartite graphs.*

4. Polynomial-time algorithm for interval graphs

A graph $G = (V, E)$ with vertices $v_1, v_2, \ldots, v_n$ is called an *interval graph* if, for some family $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$ of intervals on the real line, there is a one-to-one correspondence between V and $\mathcal{I}$ such that $v_iv_j \in E$ if and only if I_i intersects I_j for each $i, j \in \{1, 2, \ldots, n\}$. Such a family $\mathcal{I}$ of intervals is called an *interval representation* of G. In this section, we give a polynomial-time algorithm for Maximum Happy Set on interval graphs.

Theorem 2. *Given an n-vertex interval graph G and a non-negative integer k,* Maximum Happy Set *is solvable in $O(k^2n^2)$ time.*

Before the detailed description of our algorithm, we give the following simple but useful lemma.

Lemma 1. *Let $G = (V, E)$ be a graph and let V', S be subsets of V. Then, it holds that $H(G; S) \setminus V' \subseteq H(G - V'; S \setminus V')$. Moreover, if $V' \subseteq S$, then it holds that $H(G; S) \setminus V' = H(G - V'; S \setminus V')$.*

Notice that Lemma 1 is applicable to general graphs as well as interval graphs.

To explain our algorithm, we need several assumptions and notations. Given an interval graph G, an interval representation $\mathcal{I}$ of G can be constructed in linear time [3], [5], [10]. Therefore, we may assume without loss of generality that an interval graph G and its interval representation $\mathcal{I}$ are both given. In the remainder of this section, we do not distinguish between vertices of G and intervals of $\mathcal{I}$, that is, we regard I_i as not only an interval of $\mathcal{I}$ but also a vertex of G. We denote by $\mathsf{left}(I_i)$ and $\mathsf{right}(I_i)$ the left endpoint and the right endpoint of an interval $I_i \in \mathcal{I}$, respectively. It is easy to see that an interval representation $\mathcal{I}$ can be transformed into another one without changing

G so that distinct integers between 1 and $2n$ are assigned to the endpoints $\mathsf{left}(I_i)$ and $\mathsf{right}(I_i)$ of every interval I_i. Moreover, we assume that intervals of $\mathcal{I}$ are sorted in increasing order of the right endpoints, that is, $\mathsf{right}(I_i) < \mathsf{right}(I_j)$ for any integers i, j such that $1 \le i < j \le n$. We then add dummy intervals I_0 and I_{n+1} with $\mathsf{left}(I_0) = -1, \mathsf{right}(I_0) = 0, \mathsf{left}(I_{n+1}) = 2n+1$ and $\mathsf{right}(I_{n+1}) = 2n+2$ into $\mathcal{I}$. Note that, the dummy intervals I_0 and I_{n+1} correspond to the isolated vertices of G. The addition of I_0 and I_{n+1} is not essential for proving Theorem 2, but this simplifies the description of our algorithm. In the remainder of this section, we assume that G has I_0 and I_{n+1}. Let G_i be the subgraph of G induced by a vertex set $\mathcal{I}_i = \{I_0, I_1, \ldots, I_i\}$. We also define $\mathcal{I}_i^+ = N_{G_i}[I_i]$ and $\mathcal{I}_i^- = \mathcal{I}_i \setminus \mathcal{I}_i^+$.

We describe the idea of our algorithm. Let S^* be a subset of $V(G) \setminus \{I_0, I_{n+1}\}$ such that S^* maximizes $|H(G; S^*)|$ among all subsets of $V(G) \setminus \{I_0, I_{n+1}\}$ of size k. Since I_0 and I_{n+1} are the isolated vertices on G, S^* is also the optimal solution of the original graph that has no dummy vertices I_0 and I_{n+1}. In order to find S^*, we wish to compute a partial solution $S_i = S^* \cap V(G_i)$ for each $i = 0, 1, \ldots, n+1$ by means of dynamic programming. Since $G = G_{n+1}$, we have $S^* = S_{n+1}$. Notice that a partial solution S_i is not always optimal for G_i, because a happy vertex $I_{i'} \in \mathcal{I}_i$ with respect to S_i on G_i may be unhappy with respect to S^* on G. This implies that it is not enough to find only an optimal solution of G_i. To correctly compute S_i, we guess integers r, u, k' that satisfy the following three conditions for S^*:

- the interval I_r has the smallest left endpoint among all intervals in $V(G) \setminus (V(G_i) \cup S^*)$;
- the interval I_u has the largest right endpoint among all intervals in $V(G_i) \setminus S^*$, that is, $I_u \notin S^*$ and $I_{i'} \in S^*$ for every i' with $u < i' \le i$; and
- $|S_i| = k'$.

We say that a quadruple (i, r, u, k') is *compatible with* S^* if i, r, u, k' satisfy the above three conditions. Clearly, if (i, r, u, k') is compatible with S^*, then $0 \le u \le i < r \le n+1$ holds. For integers i and r, we denote by $G_{i,r}$ the subgraph of G induced by $V(G_i) \cup \{I_r\}$. We then obtain the following lemma.

Lemma 2. *Let S^* be a subset of $V(G) \setminus \{I_0, I_{n+1}\}$ such that S^* maximizes $|H(G; S^*)|$ among all subsets of $V(G) \setminus \{I_0, I_{n+1}\}$ of size k, and let $S_i = S^* \cap V(G_i)$ for an integer i with $0 \le i \le n$. For integers r, u, k' with $0 \le u \le i < r \le n+1$ and $k' \le k$, suppose that a quadruple (i, r, u, k') is compatible with S^*. Then, S_i maximizes $|H(G_{i,r}; S_i)|$ among all subsets $S_i \subseteq V(G_{i,r}) \setminus \{I_0, I_r, I_u\}$ of size k' such that $I_{i'} \in S_i$ for every i' with $u < i' \le i$.*

Lemma 2 suggests that, for the sake of computing S_i for each i, it suffices to guess integers r, u, k' and compute S maximizes $|H(G_{i,r}; S)|$ among all subsets $S \subseteq V(G_{i,r}) \setminus \{I_0, I_r, I_u\}$ of size k' such that $I_{i'} \in S$ for every i' with $u < i' \le i$. In fact, to compute the size of S^*, our algorithm uses the following two main functions $f_{\mathrm{in}}(G_{i,r}; k')$, $f_{\mathrm{out}}(G_{i,r}; k')$ and the subfunction $f'_{\mathrm{in}}(G_{i,r}; j, k')$, where i, r, j, k' are integers such that $0 \le i < r \le n+1$, $0 \le j \le \min\{k', i\} - 1$ and $0 \le k' \le k$;

- $f_{\mathrm{in}}(G_{i,r}; k')$ returns the maximum of $|H(G_{i,r}; S)|$ among all subsets $S \subseteq V(G_{i,r}) \setminus \{I_0, I_r\}$ such that $I_i \in S$ and $|S| = k'$;
- $f_{\mathrm{out}}(G_{i,r}; k')$ returns the maximum of $|H(G_{i,r}; S)|$ among all subsets $S \subseteq V(G_{i,r}) \setminus \{I_0, I_i, I_r\}$ such that $|S| = k'$; and
- $f'_{\mathrm{in}}(G_{i,r}; j, k')$ returns the maximum of $|H(G_{i,r}; S)|$ among all subsets $S \subseteq V(G_{i,r}) \setminus \{I_0, I_{i-j-1}, I_r\}$ such that $\{I_i \ldots, I_{i-j}\} \subseteq S$ and $|S| = k'$.

We let $f_{\mathrm{in}}(G_{i,r}; k') = -\infty$, $f_{\mathrm{out}}(G_{i,r}; k') = -\infty$ and $f'_{\mathrm{in}}(G_{i,r}; j, k') = -\infty$ if there exists no subset S that satisfies all the prescribed conditions for f_{in}, f_{out} and f'_{in}, respectively. We remark that $f_{\mathrm{out}}(G_{i,r}; k')$ corresponds to the case where $u = i$ and $f'_{\mathrm{in}}(G_{i,r}; j, k')$ corresponds to the case where $u = i - j - 1$ on Lemma 2. The main function $f_{\mathrm{in}}(G_{i,r}; k')$ is used to improve the running time of our algorithm. We also remark that the integer j must be less than k' and i because $j \ge k'$ violates $|S| = k'$ and $j \ge i$ violates $I_0 \notin S$. We will compute values $f_{\mathrm{in}}(G_{i,r}; k')$, $f_{\mathrm{out}}(G_{i,r}; k')$ and $f'_{\mathrm{in}}(G_{i,r}; j, k')$ by means of dynamic programming. By taking the maximum of $f_{\mathrm{in}}(G_{n,n+1}; k)$ and $f_{\mathrm{out}}(G_{n,n+1}; k)$, we obtain the maximum size of $H(G; S)$ such that $S \subseteq V(G) \setminus \{I_0, I_{n+1}\}$ and $|S| = k$.

4.1 The computation of $f_{\mathrm{in}}(G_{i,r}; k')$

If $i = 0$, then we have $f_{\mathrm{in}}(G_{i,r}; k') = -\infty$ for any r and k' because there is no subset $S \subseteq V(G_{i,r}) \setminus \{I_0, I_r\}$ such that $I_i \in S$. Similarly, if $k' = 0$, then we have $f_{\mathrm{in}}(G_{i,r}; k') = -\infty$ for any i and r. Suppose that $i > 0$ and $k' > 0$. We then compute $f_{\mathrm{in}}(G_{i,r}; k')$ from $f'_{\mathrm{in}}(G_{i,r}; j, k')$ under the assumption that $f'_{\mathrm{in}}(G_{i,r}; j, k')$ has already been computed for each j with $0 \le j \le \min\{k', i\} - 1$. Obviously, we have

$$f_{\mathrm{in}}(G_{i,r}; k') = \max_{0 \le j \le \min\{k', i\} - 1} f'_{\mathrm{in}}(G_{i,r}; j, k').$$

We explain how to compute $f'_{\mathrm{in}}(G_{i,r}; j, k')$ for each quadruple (i, r, j, k'). We assume that the main function f_{out} and the subfunction f'_{in} have been already computed in accordance with the lexicographical order of (i, r, j, k'). We consider the two subcases: (I) $j = 0$ and (II) $j > 0$.

Case (I): $j = 0$

In this case, $f'_{\mathrm{in}}(G_{i,r}; j, k')$ returns the maximum of $|H(G_{i,r}; S)|$ such that $S \subseteq V(G_{i,r}) \setminus \{I_0, I_{i-1}, I_r\}$, $I_i \in S$ and $|S| = k'$. From Lemma 1, it holds that $H(G_{i,r}; S) \setminus \{I_i\} = H(G_{i-1,r}; S \setminus \{I_i\})$. We thus compute $f'_{\mathrm{in}}(G_{i,r}; j, k')$ from $f_{\mathrm{out}}(G_{i-1,r}; k'-1)$ by deciding whether the vertex I_i is happy with respect to S on $G_{i,r}$. Clearly, if I_i is adjacent to the vertex I_{i-1} or I_r, then I_i is unhappy. Conversely, if I_i is adjacent to neither I_{i-1} nor I_r, then I_i is the isolated vertex on $G_{i,r}$ from the assumption that the intervals of $\mathcal{I}$ are sorted in increasing order of the right endpoints. Thus, I_i is happy and we have $f'_{\mathrm{in}}(G_{i,r}; j, k')$ in this case as follows:

$$f'_{\mathrm{in}}(G_{i,r}; j, k') = \begin{cases} f_{\mathrm{out}}(G_{i-1,r}; k'-1) & \text{if } I_iI_{i-1} \in E(G_{i,r}) \text{ or } I_iI_r \in E(G_{i,r}), \\ f_{\mathrm{out}}(G_{i-1,r}; k'-1) + 1 & \text{otherwise.} \end{cases}$$

Case (II): $j > 0$

This case means that $f'_{\mathrm{in}}(G_{i,r}; j, k')$ returns the maximum of $|H(G_{i,r}; S)|$ such that $S \subseteq V(G_{i,r}) \setminus \{I_0, I_{i-j-1}, I_r\}$, $\{I_i, \ldots, I_{i-j}\} \subseteq S$ and $|S| = k'$. In particular, $I_{i-1} \in S$ holds. We thus take a value

$f'_{\text{in}}(G_{i-1,r}; j-1, k'-1)$ to compute $f'_{\text{in}}(G_{i,r}; j, k')$. We then determine whether the vertex I_i is happy with respect to S on $G_{i,r}$. If I_i is adjacent to I_{i-j-1} or I_r on $G_{i,r}$, then I_i is unhappy because $I_{i-j-1}, I_r \notin S$. If I_i is adjacent to neither I_{i-j-1} nor I_r on $G_{i,r}$, then $I_{i-j-1} \in \mathcal{I}_i^-$. This implies that $\mathcal{I}_i^+ \subseteq \{I_i, \ldots, I_{i-j}\} \subseteq S$ and hence I_i is happy. Therefore, it suffices to check whether I_i is adjacent to I_{i-j-1} or I_r on $G_{i,r}$, and we have $f'_{\text{in}}(G_{i,r}; j, k')$ in this case as follows:

$$f'_{\text{in}}(G_{i,r}; j, k') = \begin{cases} f'_{\text{in}}(G_{i-1,r}; j-1, k'-1) & \text{if } I_i I_{i-j-1} \in E(G_{i,r}) \text{ or } I_i I_r \in E(G_{i,r}), \\ f'_{\text{in}}(G_{i-1,r}; j-1, k'-1) + 1 & \text{otherwise.} \end{cases}$$

4.2 The computation of $f_{\text{out}}(G_{i,r}; k')$

Let S be a subset of $V(G_{i,r}) \setminus \{I_0, I_i, I_r\}$ such that S maximizes $|H(G_{i,r}; S)|$ among all subsets $S \subseteq V(G_{i,r}) \setminus \{I_0, I_i, I_r\}$ with $|S| = k'$. Then, all vertices in $\mathcal{I}_i^+$ and I_r are unhappy with respect to S on $G_{i,r}$. However, some vertices in $\mathcal{I}_i^+ \setminus \{I_i\}$ may be contained in S because they can be used to make vertices in $\mathcal{I}_i^-$ happy. We thus consider which vertices in $\mathcal{I}_i^+ \setminus \{I_i\}$ are contained in S. In the naive way, we enumerate all subsets of $\mathcal{I}_i^+ \setminus \{I_i\}$ of size at most k'; it takes superpolynomial time in general. The following lemma provides us that the number of subsets of $\mathcal{I}_i^+ \setminus \{I_i\}$ to be enumerated is at most k'.

Lemma 3. *Let $G_{i,r}$ be an interval graph and suppose that there exist intervals $I_x, I_y \in \mathcal{I}_i^+ \setminus \{I_i\}$ such that $\mathsf{left}(I_x) < \mathsf{left}(I_y)$ for integers x, y. Then, for any subset $S \subseteq V(G_{i,r})$ such that $I_i, I_x, I_r \notin S$ and $I_y \in S$, it holds that $H(G_{i,r}; S) \subseteq H(G_{i,r}; S \cup \{I_x\} \setminus \{I_y\})$.*

Suppose that $|S \cap \mathcal{I}_i^+| = p$ for an integer p. We note that p is not greater than k' and $|\mathcal{I}_i^+| - 1$ because $p > k'$ violates $|S| = k'$ and $p > |\mathcal{I}_i^+| - 1$ violates $I_i \notin S$. We denote by $\mathcal{I}_i^p$ the set produced by picking the first p intervals in increasing order of the left endpoints of intervals in $\mathcal{I}_i^+ \setminus \{I_i\}$, that is, $\mathsf{left}(I_x) < \mathsf{left}(I_y)$ for any $I_x \in \mathcal{I}_i^p$ and any $I_y \in \mathcal{I}_i^+ \setminus (\mathcal{I}_i^p \cup \{I_i\})$. By applying Lemma 3 to S iteratively, we can obtain a subset $S' \subseteq V(G_{i,r}) \setminus \{I_0, I_i, I_r\}$ with $|S'| = k'$ such that $S' \cap \mathcal{I}_i^+ = \mathcal{I}_i^p$ and $H(G_{i,r}; S) \subseteq H(G_{i,r}; S')$. From the maximality of $|H(G_{i,r}; S)|$, S' also maximizes $|H(G_{i,r}; S')|$ among all subsets of $V(G_{i,r}) \setminus \{I_0, I_i, I_r\}$ of size k'. Thus, without enumerating all subsets of $\mathcal{I}_i^+ \setminus \{I_i\}$ of size at most k', it suffices to guess exactly p vertices in $\mathcal{I}_i^+ \setminus \{I_i\}$ are contained in S and assume that $S \cap \mathcal{I}_i^+ = \mathcal{I}_i^p$.

We next give another lemma that plays a central role in the computation of $f_{\text{out}}(G_{i,r}; k')$. Let i' be an integer such that the interval $I_{i'}$ has the largest right endpoint among all intervals in $\mathcal{I}_i^-$.

Lemma 4. *Let S be a subset of $V(G_{i,r}) \setminus \{I_i, I_r\}$ and let $S' = S \cap \mathcal{I}_i^+$. In addition, let r' be an integer such that the interval $I_{r'}$ has the smallest left endpoint among all intervals in $\mathcal{I}_i^+ \cup \{I_r\} \setminus S'$. Then, $H(G_{i,r}; S) = H(G_{i',r'}; S \setminus S')$.*

We have prepared for computing $f_{\text{out}}(G_{i,r}; k')$ for a triple (i, r, k') of integers such that $0 \leq i < r \leq n+1$ and $0 \leq k' \leq k$. If $i = 0$, the graph $G_{i,r}$ consists of the two isolated vertices I_0 and I_r. Only $S = \emptyset$ satisfies the prescribed conditions for $f_{\text{out}}(G_{i,r}; k')$. Thus, for any integer $r > 0$, we have $f_{\text{out}}(G_{i,r}; k') = 0$ if $k' = 0$; otherwise $f_{\text{out}}(G_{i,r}; k') = -\infty$.

Suppose that $i > 0$. Let S be a subset of $V(G_{i,r}) \setminus \{I_0, I_i, I_r\}$ such that S maximizes $|H(G_{i,r}; S)|$ among all subsets $S \subseteq V(G_{i,r}) \setminus \{I_0, I_i, I_r\}$ with $|S| = k'$. As mentioned before, if $|S \cap \mathcal{I}_i^+| = p$ for an integer p with $0 \leq p \leq \min\{k', |\mathcal{I}_i^+| - 1\}$, we can assume that $S \cap \mathcal{I}_i^+ = \mathcal{I}_i^p$ from Lemma 3. Let r' be an integer such that the interval $I_{r'}$ has the smallest left endpoint among intervals in $\mathcal{I}_i^+ \cup \{I_r\} \setminus \mathcal{I}_i^p$. For an optimal solution S^* of G, if the quadruple (i, r, i, k') is compatible with S^*, then the quadruple $(i', r', u, k'-p)$ is also compatible with S^* for some integer u with $0 \leq u \leq i'$. Therefore, by setting $S' = \mathcal{I}_i^p$ on Lemma 4, we compute $f_{\text{out}}(G_{i,r}; k')$ as follows:

$$f_{\text{out}}(G_{i,r}; k') = \max_{0 \leq p \leq \min\{k', |\mathcal{I}_i^+| - 1\}} \{f_{\text{in}}(G_{i',r'}; k'-p), f_{\text{out}}(G_{i',r'}; k'-p)\}.$$

The total running time of our algorithm is $O(k^2 n^2)$, as claimed in Theorem 2.

5. Polynomial-time algorithm for permutation graphs

Consider two horizontal parallel lines on the plane and a permutation π between integers 1 and n, where the upper line has distinct n points labeled $1, 2, \ldots, n$, and the lower line has distinct n points labeled $\pi(1), \pi(2), \ldots, \pi(n)$ in the order from left to right, respectively. For each integer $i = 1, 2, \ldots, n$, let L_i be a line segment from a point i on the upper line to a point i on the lower line. A graph $G = (V, E)$ with vertices $v_1, v_2, \ldots, v_n$ is called a *permutation graph* if there exists a permutation π between 1 and n such that for any two integers i and j, $v_i v_j \in E$ if and only if L_i intersects L_j. A family $\mathcal{L}$ of line segments corresponding to G is called a *line representation* of G. We give a polynomial-time algorithm for Maximum Happy Set on permutation graphs by the same algorithmic approach as interval graphs. The details are omitted from this technical report.

Theorem 3. *Given an n-vertex permutation graph G and a non-negative integer k,* Maximum Happy Set *is solvable in $O(k^3 n^2)$ time.*

6. Conclusion

In this technical report, we studied the complexity of Maximum Happy Set on subclasses of co-comparability graphs; co-bipartite graphs, interval graphs and permutation graphs. We showed that Maximum Happy Set remains NP-hard even for co-bipartite graphs. We then gave polynomial-time algorithms for interval graphs and permutation graphs which run in $O(k^2 n^2)$ time and $O(k^3 n^2)$ time, respectively. Especially, our algorithm for interval graphs improved the best known running time $O(kn^8)$. Our polynomial-time algorithms employ basically the same technique. We believe that the technique is applicable to Maximum Happy Set on other graph classes.

The complexity of Maximum Happy Set has been studied for various graph classes. However, the (in)tractability of Maximum Happy Set on planar graphs remains open. We note that the complexity of the edge variant of Maximum Happy Set is also unknown for planar graphs.

Acknowledgments

This work is partially supported by JSPS KAKENHI Grant Numbers JP18H04091, JP19K11814, JP20H05793, JP20H05794, JP20K11666, JP21K11755 and JP21K21302, Japan.

References

[1] Asahiro, Y., Eto, H., Hanaka, T., Lin, G., Miyano, E. and Terabaru, I.: Parameterized Algorithms for the Happy Set Problem, *WALCOM: Algorithms and Computation* (Rahman, M. S., Sadakane, K. and Sung, W.-K., eds.), Cham, Springer International Publishing, pp. 323–328 (2020).

[2] Asahiro, Y., Eto, H., Hanaka, T., Lin, G., Miyano, E. and Terabaru, I.: Complexity and approximability of the happy set problem, *Theoretical Computer Science*, Vol. 866, pp. 123–144 (online), DOI: 10.1016/j.tcs.2021.03.023 (2021).

[3] Booth, K. S. and Lueker, G. S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms, *Journal of Computer and System Sciences*, Vol. 13, No. 3, pp. 335–379 (online), DOI: https://doi.org/10.1016/S0022-0000(76)80045-1 (1976).

[4] Coorg, S. R. and Rangan, C. P.: Feedback vertex set on cocomparability graphs, *Networks*, Vol. 26, No. 2, pp. 101–111 (online), DOI: https://doi.org/10.1002/net.3230260205 (1995).

[5] Corneil, D. G., Olariu, S. and Stewart, L.: The LBFS Structure and Recognition of Interval Graphs, *SIAM Journal on Discrete Mathematics*, Vol. 23, No. 4, pp. 1905–1953 (online), DOI: 10.1137/S0895480100373455 (2010).

[6] Deogun, J. S. and Steiner, G.: Polynomial Algorithms for Hamiltonian Cycle in Cocomparability Graphs, *SIAM J. Comput.*, Vol. 23, No. 3, pp. 520–552 (online), DOI: 10.1137/S0097539791200375 (1994).

[7] Easley, D. and Kleinberg, J.: *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*, Cambridge University Press (2010).

[8] Eto, H., Ito, T., Miyano, E., Suzuki, A. and Tamura, Y.: Happy Set Problem on Subclasses of Co-comparability Graphs, *WALCOM: Algorithms and Computation* (to appear).

[9] Feige, U., Kortsarz, G. and Peleg, D.: The Dense k-Subgraph Problem, *Algorithmica*, Vol. 29, No. 3, pp. 410–421 (online), DOI: 10.1007/s004530010050 (2001).

[10] Hsu, W.-L. and Ma, T.-H.: Fast and Simple Algorithms for Recognizing Chordal Comparability Graphs and Interval Graphs, *SIAM Journal on Computing*, Vol. 28, No. 3, pp. 1004–1020 (online), DOI: 10.1137/S0097539792224814 (1998).

[11] Kratsch, D. and Stewart, L.: Domination on Cocomparability Graphs, *SIAM Journal on Discrete Mathematics*, Vol. 6, No. 3, pp. 400–417 (online), DOI: 10.1137/0406032 (1993).

[12] Zhang, P. and Li, A.: Algorithmic aspects of homophyly of networks, *Theoretical Computer Science*, Vol. 593, pp. 117–131 (online), DOI: https://doi.org/10.1016/j.tcs.2015.06.003 (2015).