On Computing Longest Paths in Small Graph Classes

Ryuhei Uehara[*] Yushi Uno[†]

July 28, 2005

Abstract

The longest path problem is to find a longest path in a given graph. While the graph classes in which the Hamiltonian path problem can be solved efficiently are widely investigated, few graph classes are known to be solved efficiently for the longest path problem. For a tree, a simple linear time algorithm for the longest path problem is known. We first generalize the algorithm, and show that the longest path problem can be solved efficiently for weighted trees, block graphs, and cacti. We next show that the longest path problem can be solved efficiently on some graph classes that have natural interval representations.
Keywords: efficient algorithms, graph classes, longest path problem.

1 Introduction

The Hamiltonian path problem is one of the most well known $\mathcal{NP}$-hard problem, and there are numerous applications of the problems [17]. For such an intractable problem, there are two major approaches; approximation algorithms [20, 2, 35] and algorithms with parameterized complexity analyses [15]. In both approaches, we have to change the decision problem to the optimization problem. Therefore the longest path problem is one of the basic problems from the viewpoint of combinatorial optimization. From the practical point of view, it is also very natural approach to try to find a longest path in a given graph, even if it does not have a Hamiltonian path. However, finding a longest path seems to be more difficult than determining whether the given graph has a Hamiltonian path or not. Even if a graph has a Hamiltonian path, it is impossible to find paths of length $n - n^{\epsilon}$ for any $\epsilon > 0$ unless $\mathcal{P} = \mathcal{NP}$ [21]. In general, the longest path problem does not belong to APX unless $\mathcal{P} = \mathcal{NP}$ [21], and the best known performance ratio of an approximation algorithm is $O(n(\log\log n/\log n)^2)$ [7] (see also [27, 1, 32, 36] for related results).

Recently, many new graph classes are proposed, and the complexity of hard problems on those graph classes are investigated [9, 18]. The classification of the graph classes by the difficulty in solving the Hamiltonian path problem gives us an insight for the longest path problem. If the Hamiltonian path problem is $\mathcal{NP}$-hard, the longest path problem is also intractable since the Hamiltonian path problem is the special case of the longest path problem. Such "hard" graph classes include chordal bipartite graphs, strongly chordal split graphs (and hence chordal graphs and split graphs)[28], undirected path graphs, double interval graphs, rectangle graphs [6], and circle graphs [13]. On the other hand, proper interval graphs are a "trivial" graph class in the sense that all connected proper interval graphs have a Hamiltonian path [5]. Thus we can find a longest path of length equal to one less than the number of vertices for any given connected proper interval graph. Between them, there are "non-trivial" graph classes; the Hamiltonian path problem is polynomial time solvable for circular arc graphs (and hence interval graphs) [14], and bipartite permutation graphs [33]. Moreover, such "non-trivial" graph classes include infinite number of graphs that have no Hamiltonian paths (e.g., trees). In this paper, we focus on the longest path problem for the "non-trivial" graph classes, and propose efficient algorithms for several graph classes.

[*]Department of Information Processing, School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), Ishikawa 923-1292, Japan. uehara@jaist.ac.jp

[†]Department of Mathematics and Information Sciences, Graduate School of Science, Osaka Prefecture University, Sakai 599-8531, Japan. uno@mi.s.osakafu-u.ac.jp

There are few polynomial time algorithms for finding a longest path in a graph; as far as the authors know, trees are the only graph class which is natural and non-trivial such that the longest path problem can be solved in polynomial time. The algorithm for trees was invented by W. Dijkstra around 1960. It runs in linear time, and the formal proof is given by R.W. Bulterman et al [11].

We first generalize the Dijkstra's algorithm and its correctness proof in [11], and show that the longest path problem can be solved efficiently for (vertex/edge) weighted trees, block graphs, and cacti.

Next, we focus on the graph classes which have natural interval representations. We consider some graph classes between "non-trivial" and "trivial" graph classes and show that the longest path problem can be solved efficiently on these classes.

The class of interval bigraphs has a natural bipartite analogy with the class of interval graphs. In the class, there are some subclasses; chain graphs $\subset$ bipartite permutation graphs $\subset$ biconvex graphs $\subset$ convex graphs $\subset$ interval bigraphs. We will show some relationship among these graph classes (though some of them seem to be folklore), and show a linear time algorithm for finding a longest path in a bipartite permutation graph. (For these graph classes, see, e.g., [37, 23] for chain graphs, [33, 10, 25] for bipartite permutation graphs, [9] for (bi)convex graphs, and [29, 19] for interval bigraphs.)

We then turn to the interval graphs and their subclasses. Although all longest paths of a connected interval graph have non-empty intersection [3, Corollary 2.2], there are no efficient algorithms for finding a specific longest path in an interval graph. Between the class of interval graphs and the class of proper interval graphs, we introduce a new graph class named "interval biconvex graphs." A interval biconvex graph G is an interval graph that corresponds to a biconvex graph G' such that G and G' have the same interval representations. That is, the class of interval biconvex graphs is the class that the class of biconvex graphs is a natural bipartite analogy. We also show the inclusion among graph classes (proper interval graphs $\cup$ threshold graphs) $\subset$ interval biconvex graphs $\subset$ interval graphs. That is, the class of interval biconvex graphs is a generalization of both of proper interval graphs and threshold graphs. Finally, we give an algorithm that finds a longest path for any interval biconvex graph in $O(n^3(m + n \log n))$ time.

2 Preliminaries

A graph $G = (V, E)$ consists of a finite set V of *vertices* and a collection E of 2-element subsets of V called *edges*. The *neighborhood* of a vertex v in a graph $G = (V, E)$ is the set $N_G(v) = \{u \in V \mid \{u, v\} \in E\}$, and the *degree* of a vertex v is $|N_G(v)|$ and is denoted by $\deg_G(v)$. For a subset U of V, we denote by $N_G(U)$ the set $\{v \in V \mid v \in N(u) \text{ for some } u \in U\}$. If no confusion can arise we will omit the index G. We denote the closed neighborhood $N(v) \cup \{v\}$ by $N[v]$. Given a graph $G = (V, E)$, its *complement* is defined by $\bar{E} = \{\{u, v\} \mid \{u, v\} \notin E\}$, and denoted by $\bar{G} = (V, \bar{E})$. A vertex set I is an *independent set* if $G[I]$ contains no edges, and then the graph $\bar{G}[I]$ is said to be *clique*.

For a graph $G = (V, E)$, a sequence of the vertices $v_0, v_1, \ldots, v_l$ is a *path*, denoted by $(v_0, v_1, \ldots, v_l)$, if $\{v_j, v_{j+1}\} \in E$ for each $0 \leq j < l$. The *length* of a path is the number of edges on the path. For two vertices u and v, the *distance* of the vertices, denoted by $d(u, v)$, is the minimum length of the paths joining u and v. A *cycle* is a path beginning and ending with the same vertex. A cycle of length i is denoted by C_i. An edge which joins two vertices of a cycle but is not itself an edge of the cycle is a *chord* of that cycle. A graph is *chordal* if every cycle of length at least 4 has a chord. Given a graph $G = (V, E)$, a vertex $v \in V$ is *simplicial* in G if $G[N(v)]$ is a clique in G. A graph $G = (V, E)$ is *bipartite* if V can be partitioned into two sets X and Y such that every edge joins a vertex in X and the other vertex in Y.

A graph $G = (V, E)$ is called a *threshold graph* when there exist nonnegative reals $w(v)$ for $v \in V$ and t such that $\sum_{v \in S} w(v) \leq t$ if and only if S is an independent set in G.

Tree-like graph classes: We here introduce some graph classes that have similar structure to trees. A graph G is a *block graph* if G is connected and every maximal 2-connected subgraph is a clique. Block graphs can be seen as graphs obtained from trees by replacing each edge by a clique, and the cliques have at most one vertex in common. A graph G is a *cactus* if every edge is part of at most one cycle in G. Similarly, cacti can be seen as graphs obtained from trees by replacing each edge by a cycle, and the cycles have at most one vertex in common. In other words, a cactus is a graph whose block is either a path or a cycle. See [9] for further details of the graph classes.

Interval graph representation and related classes: A graph (V, E) with $V = \{v_1, v_2, \ldots, v_n\}$ is an *interval graph* if there is a finite set of intervals $\mathcal{I} = \{I_{v_1}, I_{v_2}, \ldots, I_{v_n}\}$ 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 $0 < i, j \leq n$. We call the set $\mathcal{I}$ of intervals an *interval representation* of the graph. For each interval I, we denote by $R(I)$ and $L(I)$ the right and left endpoints of the interval, respectively (hence we have $L(I) \leq R(I)$ and $I = [L(I), R(I)]$).

A bipartite graph (X, Y, E) with $X = \{x_1, x_2, \ldots, x_{n_x}\}$ and $Y = \{y_1, y_2, \ldots, y_{n_y}\}$ is an *interval bigraph* if there are families of intervals $\mathcal{I}_X = \{I_{x_1}, I_{x_2}, \ldots, I_{x_{n_x}}\}$ and $\mathcal{I}_Y = \{I_{y_1}, I_{y_2}, \ldots, I_{y_{n_y}}\}$ such that $\{x_i, y_j\} \in E$ iff $I_{x_i} \cap I_{y_j} \neq \emptyset$ for each i and j with $1 \leq i \leq n_x$ and $1 \leq j \leq n_y$. We also call the families of intervals $(\mathcal{I}_X, \mathcal{I}_Y)$ an *interval representation* of the graph. If no confusion can arise we sometimes unify a vertex v and corresponding interval I_v, and denote as $N(I_v)$, $L(v)$, and so on.

For two intervals I and J, we write $I \prec J$ if $L(I) \leq L(J)$ and $R(I) \leq R(J)$. For any interval representation $\mathcal{I}$ and a point p, $N[p]$ denotes the set of intervals that contain the point p.

Given an interval (bi)graph, an interval representation is said to be *compact* if the following conditions hold;

1. for each interval I, $R(I)$ and $L(I)$ are integers, and
2. for each integer p, q with $N[p] \neq \emptyset$ and $N[q] \neq \emptyset$, $N[p] \setminus N[q] \neq \emptyset$ and $N[q] \setminus N[p] \neq \emptyset$.

Lemma 1 For any given interval graph $G = (V, E)$, there is a linear time algorithm that constructs a compact interval representation $\mathcal{I}$ such that

1. each simplicial vertex v is a point; that is, $L(I_v) = R(I_v)$,
2. $\cup_{I \in \mathcal{I}} I$ is contained in $[1, |V|]$, and
3. each integer point i with $N[i] \neq \emptyset$ corresponds to a distinct maximal clique of G.

Proof. Given an interval graph $G = (V, E)$, we can construct an $\mathcal{MPQ}$-tree, which is a kind of labeled $\mathcal{PQ}$-tree, in linear time [24]. From the $\mathcal{MPQ}$-tree, in a natural way, we can construct a compact interval representation in linear time. The claim can be proved by the induction of the number of nodes/sections in the $\mathcal{MPQ}$-tree, but it is straightforward and tedious, and omitted here. We show that the constructed compact interval representation satisfies the conditions.

Let v be any simplicial vertex in G. By the definition, $N(v)$ induces a clique. Thus, by the Helly property (see, e.g.,[4]), all vertices u in $N(v)$ share a common point. Therefore, I_v contains the point. If I_v is not an integer point, I_v should contain an interval $[i, i+1]$ for some integer i. Then, $N[i] = N[i+1]$, which contradicts the second condition of the definition. Thus we have the claim 1.

The maximal cliques of an interval graph G can be linearly ordered such that for each vertex v, the maximal cliques containing v occur consecutively [8]. Since the representation is compact, we have the claim 3 immediately. It is well known that chordal graphs, and hence interval graphs, have at most n maximal cliques. This fact with the claim 3 implies the claim 2. ∎

New graph class: An interval graph is *proper interval graph* if no two intervals I and J properly contain each other. That is, either $I \prec J$ or $J \prec I$ for each I and J.

Let $G = (X, Y, E)$ be a bipartite graph. An ordering $<$ of X in G has the *adjacency property* if for each vertex $y \in Y$, $N(y)$ consists of vertices that are consecutive in the ordering $<$ of X. A bipartite graph $G = (X, Y, E)$ is *biconvex* if there are orderings of X and Y that fulfill the adjacency property, and that is *convex* if there is an ordering of X or Y that fulfills the adjacency property.

If a vertex v corresponds to an integer point on a compact interval representation, the vertex v and the corresponding integer $L(I_v) = R(I_v)$ are sometimes unified; for example, "$u := v + 1$" means that "let u be a vertex with $L(I_u) = R(I_u) = L(I_v) + 1 = R(I_v) + 1$."

Given a biconvex graph $G = (X, Y, E)$ with $|X| = n_x$ and $|Y| = n_y$, a compact interval representation $\mathcal{I}(G)$ is constructed as follows: Let $x_1 < x_2 < \cdots < x_{n_x}$ and $y_1 < y_2 < \cdots < y_{n_y}$ be the orderings over X and Y that have adjacency property. Let x_i correspond to the integer point i with $1 \leq i \leq n_x$. For each j with $1 \leq j \leq n_y$, since each y_j contains consecutive xs, we can let y_j correspond to the interval $[L(y_j), R(y_j)]$ such that $L(y_j) = l$ and $R(y_j) = r$, where x_l and x_r are the minimum and maximum vertices in $N(y_j)$, respectively.

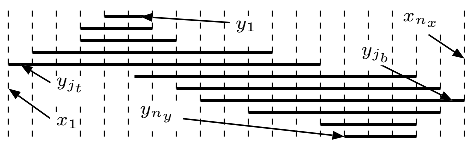

Figure 1: Bounds of a biconvex graph.

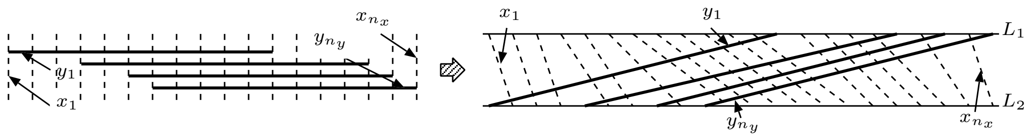

Figure 2: Proper biconvex graph = bipartite permutation graph.

Lemma 2 For the compact interval representation $\mathcal{I}(G)$ of a biconvex graph $G = (X, Y, E)$, there are two indices j_t and j_b such that (1) $L(y_1) \geq L(y_2) \geq \cdots \geq L(y_{j_t})$ and $L(y_{j_t}) \leq L(y_{j_t+1}) \leq \cdots \leq L(y_{n_y})$, and (2) $R(y_1) \leq R(y_2) \leq \cdots \leq R(y_{j_b})$ and $R(y_{j_b}) \geq R(y_{j_t+1}) \geq \cdots \geq R(y_{n_y})$ (Figure 1). Moreover, we can assume that $j_t \leq j_b$ without loss of generality.

Proof. We note that $x_1 \in N(y_{j_t})$ and $x_{n_x} \in N(y_{j_b})$ if and only if G is connected. If the two indices do not exist, we have three consecutive vertices, say, $y_i < y_j < y_k$ such that $L(y_i) > L(y_j)$ and $L(y_k) > L(y_j)$. Then there is a vertex x in X such that $y_i, y_k \in N(x)$ and $y_j \notin N(x)$. This contradicts the definition of biconvex graph. Thus there are two indices j_t and j_b. If $j_t > j_b$, reversing the ordering of X, we have $j_t < j_b$. ∎

A biconvex graph $G = (X, Y, E)$ is said to be a *chain graph* if it has a vertex ordering in Lemma 2 and $j_t = n_y$. That is, for the interval representation of $\mathcal{I}(G)$, a biconvex graph G is a chain graph if $J_1 \subseteq J_2 \subseteq \cdots \subseteq J_{n_y}$, where $J_1, J_2, \ldots, J_{n_y}$ are the corresponding intervals of $y_1, y_2, \ldots, y_{n_y}$.

A graph $G = (V, E)$ with $|V| = \{1, 2, \ldots, n\}$ is said to be a *permutation graph* if there is a permutation σ over $\{1, 2, \ldots, n\}$ such that $\{i, j\} \in E$ if and only if $(i - j)(\sigma^{-1}(i) - \sigma^{-1}(j)) < 0$. Intuitively, each vertex v in a permutation graph corresponds to a line v joining two points on two parallel lines L_1 and L_2. Then two vertices v and u are adjacent if and only if the corresponding lines v and u intersect. The ordering of vertices gives the ordering of the points on L_1, and the permutation of the ordering gives the ordering of the points on L_2. When a permutation graph is bipartite, it is said to be a *bipartite permutation graph*.

Here we can introduce a natural bipartite analogy of a proper interval graph; we say a biconvex graph $G = (X, Y, E)$ is *proper* if it has a vertex ordering in Lemma 2 and $J_1 \prec J_2 \prec \cdots \prec J_{n_y}$, where $J_1, J_2, \ldots, J_{n_y}$ are the corresponding intervals of $y_1, y_2, \ldots, y_{n_y}$ (and hence we also have $I_1 \prec I_2 \prec \cdots \prec I_{n_x}$, where $I_1, I_2, \ldots, I_{n_x}$ are the corresponding intervals of $x_1, x_2, \ldots, x_{n_x}$). We here show that the class of "proper" biconvex graphs is coincide with the class of bipartite permutation graphs.

Theorem 3 A bipartite graph $G = (X, Y, E)$ is proper biconvex if and only if G is a bipartite permutation.

Proof. We first assume that a bipartite graph $G = (X, Y, E)$ is proper biconvex graph with $X = \{x_1, x_2, \ldots, x_{n_x}\}$ and $Y = \{y_1, y_2, \ldots, y_{n_y}\}$. Then $y_1 \prec y_2 \prec \cdots \prec y_{n_y}$ by definition. From the interval representation, we construct the permutation σ that characterizes the bipartite permutation graph G as follows (see Figure 2). First, we put the lines corresponding to X between two parallel lines L_1 and L_2. For y_i, let x_j and x_k be the vertices in X such that $N(y_i) = \{x_j, x_{j+1}, \ldots, x_k\}$. Then we modify the line corresponding to y_i joining the point between x_k and x_{k+1} on L_1 and the point between x_{j-1} and x_j on L_2. The permutation σ can be obtained from the diagram immediately. From given bipartite permutation graph $G = (X, Y, E)$, the construction of the interval representation of the proper biconvex graph G is also straightforward. ∎

From the definitions, we have the following observation:

Observation 4 For any given biconvex graph $G = (X, Y, E)$ with two indices j_t and j_b in Lemma 2, we partition Y into $Y_1 := \{y_1, \ldots, y_{j_t}\}$, $Y_2 := \{y_{j_t+1}, \ldots, y_{j_b}\}$, and $Y_3 := \{y_{j_b+1}, \ldots, y_{n_2}\}$. Let X_i be the set of vertices in $N(Y_i)$ and E_i be the set of edges induced by $X_i \cup Y_i$. Then by Theorem 3, the graph $G_2 = (X_2, Y_2, E_2)$ is a bipartite permutation graph. From the definition, the graphs $G_1 = (X_1, Y_1, E_1)$ and $G_3 = (X_3, Y_3, E_3)$ are chain graphs.

That is, any biconvex graph consists of two chain graphs and one bipartite permutation graph.

Theorem 5 A chain graph $G = (X, Y, E)$ is a bipartite permutation graph.

Proof. By definition, there is an index i_c such that $N(x_1) \subseteq N(x_2) \subseteq \cdots \subseteq N(x_{i_c})$ and $N(x_{n_x}) \subseteq N(x_{n_x-1}) \subseteq \cdots \subseteq N(x_{i_c})$, where $n_x = |X|$. Since Y is linearly ordered in inclusion, for any pair of x_i and $x_{i'}$ with $1 \le i \le c$ and $c \le i' \le n_x$, we have $N(x_i) \subseteq N(x_{i'})$ or $N(x_{i'}) \subseteq N(x_i)$. Thus, we can linearly order both of X and Y; the ordering of Y is as it is, and the ordering of X is computed by the following algorithm:

```
Algorithm 1: REORDER
  Input  : A chain graph $G = (X, Y, E)$ with $|X| = n_x$ and $|Y| = n_y$, the orderings over $X$ and $Y$,
           and a compact interval representation $\mathcal{I}(G)$.
  Output: A linear ordering of $X$ in inclusion.
1 $\ell := 1$; $r := n_x$;
2 if $N(x_\ell) \subseteq N(x_r)$ then output $x_\ell$ and $\ell := \ell + 1$ else output $x_r$ and $r := r - 1$;
3 if $\ell < r$ then goto step 2 else output $x_\ell$ and halt.
```

The correctness of the algorithm is trivial. ∎

We note that the algorithm REORDER runs in $O(n_x)$ time, since $N(x_\ell) \subseteq N(x_r)$ if and only if $\min\{N(x_\ell)\} \ge \min\{N(x_r)\}$.

We here introduce new graph class between interval graphs and proper interval graphs. An interval graph $G = (V, E)$ is said to be *an interval biconvex graph* if V can be partitioned into two sets S and Y such that

1. each vertex $x \in S$ is simplicial in G, and
2. bipartite graph $G' := (S, Y, E')$ is a biconvex graph, where $E' := \{\{x, y\} \mid x \in S, y \in Y, \text{ and } \{x, y\} \in E\}$.

Intuitively, interval biconvex graphs G are interval graphs obtained from biconvex graphs G'; they have the same interval representations. However, we allow to some vertices in S to correspond to the same point:

Lemma 6 Let $G = (V = S \cup Y, E)$ be an interval biconvex graph. Let v be a (simplicial) vertex in S. Let $N_S[v]$ be the set $\{v\} \cup (N(v) \cap S)$ in G. Then $N_S[v]$ induces a clique, and for any two vertices v_1 and v_2 in $N_S[v]$, $N_G[v_1] = N_G[v_2]$.

Proof. Since v is simplicial, $N_S[v]$ induces a clique. Thus we show that $N[v_1] = N[v_2]$ for any v_1 and v_2 in $N_S[v]$. To derive a contradiction, we assume that there is a vertex $w \in N[v_1] \setminus N[v_2]$. Then, both of v_2 and w are in $N[v_1]$, and $\{w, v_2\} \notin E$, which contradicts v_1 is simplicial in G. ∎

Corollary 7 Let $G = (V = S \cup Y, E)$ be an interval biconvex graph, and $\mathcal{I}(G)$ be a compact interval representation of G. Then

1. for each vertex v in S, I_v is an integer point, and
2. for each vertices u and v in S with $N[u] = N[v]$, I_v and I_u are the same integer point.

We have the following proper inclusion:

Lemma 8 (Proper interval graphs $\cup$ threshold graphs) $\subset$ interval biconvex graphs $\subset$ interval graphs.

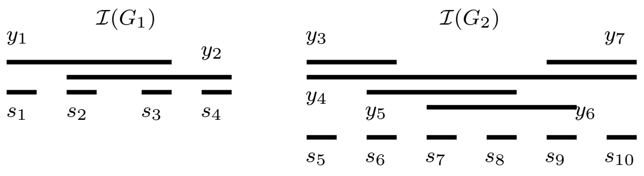

Figure 3: Graphs in the differences.

Proof. By their definitions, the following inclusion is straightforward; proper interval graphs $\subseteq$ interval biconvex graphs $\subseteq$ interval graphs. Thus we first show that any threshold graph $G = (V, E)$ with its threshold t is an interval biconvex graph. We assume that vertices are ordered with respect to their weights, that is, $w(v_1) \le w(v_2) \le \cdots \le w(v_n)$. Let i be the index that $w(v_i) \ge \frac{t}{2}$. We partition V into $V' := \{v_i, \ldots, v_n\}$ and $S := V \setminus V'$. Then it is easy to see that (1) $G[V']$ is a clique, (2) each vertex in S is simplicial, (3) for each vertex v in S, there is an index j such that $N(v) = \{v_1, \ldots, v_j\}$. Thus G is interval biconvex graph.

We also show that the inclusions are proper; the graph G_1 with an interval representation $\mathcal{I}(G_1)$ in Figure 3 is an interval biconvex graph. However, G_1 is not threshold graph; if $w(y_1) > w(y_2)$, s_4 should overlap with y_1, and vice versa. On the other hand, G_1 is not proper interval graph; if y_1, y_2, s_2, and s_3 have the same length, s_1 cannot overlap with y_1 without overlapping with s_2. On the other hand, the graph G_2 in Figure 3 is an interval graph, but it is not an interval biconvex graph; if y_4 has the same length to y_5 and y_6, y_4 cannot overlap both of s_5 and s_{10}. ∎

3 Algorithms for tree-like graphs

Given a finite tree T, a longest path can be found in linear time. A simple procedure is invented by E.W. Dijkstra around 1960, and its formal proof is given in [11]. In order to illustrate our algorithms for tree-like graphs, we first give a framework that allows a generalization of the Dijksta's algorithm by transforming G into another graph G'. For a given graph $G = (V, E)$, suppose that we can construct a new graph $G' = (V', E')$ satisfying the following three conditions:

1. $V \subseteq V'$,
2. for each pair u, v in V, $d_{G'}(u, v)$ equals the length of a longest path joining u and v in G, and
3. for each pair u, v in V, $d_{G'}(u, v)$ is given by the unique shortest path on G'.

Then the following algorithm computes the length of a longest path in G by using the graph G' in $O(|V| + |E| + |V'| + |E'|)$ time and space.

Algorithm 2: TR

```
Input  : Graphs $G = (V, E)$ and $G' = (V', E')$.
Output: The length of a longest path P in $G$.
1 pick up any vertex $u$ in $V$ of $G'$;
2 find a vertex $x$ in $V$ such that $d_{G'}(u, x)$ is largest;
3 find a vertex $y$ in $V$ such that $d_{G'}(x, y)$ is largest;
4 output the length of a shortest path joining $x$ and $y$ on $G'$.
```

Letting $G' = G$, we have the Dijkstra's algorithm for a tree.

Theorem 9 Algorithm TR computes the length of a longest path in G in linear time of G and G'.

Proof. In the proof in [11], the correctness of the Dijkstra's algorithm is based on the following three points;

- for any vertices u and v, the shortest path joining them gives the longest path joining them,
- for any vertices u, v and w, we have $d(u, v) \le d(u, w) + d(w, v)$, and

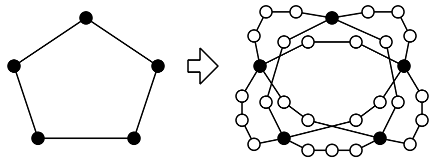

Figure 4: Gadgets for C_5.

- for any vertices u, v and w, we have $d(u,v) = d(u,w) + d(w,v)$ if and only if w is on the path joining u and v.

By condition of G', for any vertices u and v in V, the length of the shortest path joining u and v on G' is equal to the length of the longest path joining u and v on G. For any vertices u, v, and w in V, we also have $d_{G'}(u,v) \le d_{G'}(u,w) + d_{G'}(w,v)$. By condition of G', $d_{G'}(u,v)$ is given by the unique shortest path on G'. Thus, for any vertices u, v and w in V, we have $d_{G'}(u,v) = d_{G'}(u,w) + d_{G'}(w,v)$ if and only if w is on the shortest path joining u and v. Therefore, we can use the same argument in the proof of [11] to show the correctness of Algorithm TR. Since $d_{G'}(u,v)$ is given by the unique shortest path on G' for each pair $u, v \in V$, we can use the standard breadth first search to execute the algorithm in linear time. ∎

Hereafter, we show three graph classes such that Algorithm TR computes the length of a longest path with suitable reductions. We note that, in step 4, Algorithm TR outputs the length of a longest path. However, in the algorithms for the specified graph classes, it is easy to modify them to output the longest path itself of G.

Lemma 10 Let $G = (V,E)$ be a tree, and $w : V \cup E \to Z$ an integer weight function of vertices and edges. We define the length of a weighted path $(v_1, v_2, \ldots, v_k)$ by $\sum_{i=1}^{k} w(v_i) + \sum_{i=1}^{k-1} w(\{v_i, v_{i+1}\})$. Then the weighted longest path problem can be solved in $O(|V| + |E| + s)$ time and space, where $s = \sum_{x \in V \cup E} w(x)$.

Proof. We can use the Dijkstra's algorithm straightforwardly. ∎

Theorem 11 Let $G = (V,E)$ be a block graph. Then the longest path problem can be solved in $O(|V| + |E|)$ time and space.

Proof. Given block graph G, we construct a weighted tree G' as follows: For each maximal clique K of size $k > 2$, we first remove all edges in K, second put a vertex c of weight $k-3$, and third join c and each vertex in K by an edge of weight 1. We repeat the replacement for all maximal cliques of size > 2. The resulting graph G' is an integer weighted tree. Let u and v be any two vertices in V. Then the length of a longest path between u and v on G is equal to the weight of the path joining u and v on G'. Thus we can use Lemma 10. ∎

Theorem 12 Let $G = (V,E)$ be a cactus. Then the longest path problem can be solved in $O(|V|^2)$ time and space.

Proof. We first show a reduction in $O(|V|^3)$ time and space. Let C_k be any cycle of k vertices in G. We replace C_k by a gadget C'_k defined as follows (Figure 4): For each pair of vertices v_i and v_j on C_k, if $d_{C_k}(v_i, v_j) = h$, C'_k contains a path of length $k-h$ (with auxiliary vertices). We note that the length of the longest path joining v_i and v_j on G (or C_k) is $k-h$. We replace all cycles of G by the gadgets. The resulting graph G' has $O(|V|^3)$ vertices and edges, and the reduction can be done in $O(|V|^3)$ time and space. To show the correctness of Algorithm TR, we show that for each pair u, v in V, (1) $d_{G'}(u,v)$ equals the length of a longest path joining u and v in G, and (2) the shortest path joining u and v on G' is uniquely determined. Since G is a cactus, we show that for each pair v_i and v_j on C_k, $d_{G'}(v_i, v_j)$ equals the length of a longest path joining v_i and v_j in G. Without loss of generality, we assume that $i = 0$ and $0 < j \le \frac{k}{2}$. By the construction, we have $d_{G'}(v_0, v_j)$ is at most $k-j$ by the added path joining v_0 and v_j. We assume that there is a shorter path P in G' joining v_0 and v_j to derive a contradiction. Since G is a cactus, P contains at least one vertex v' on C_k with $v' \neq v_0, v_j$. From the construction,

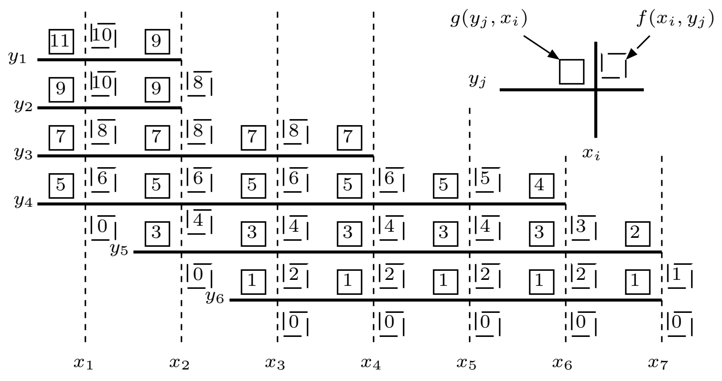

Figure 5: An example for $f(x_i, y_j)$ and $g(y_j, x_i)$

for any pair of vertices u and u' on C'_k, we have $\frac{k}{2} \le d_{G'}(u, u') \le k - 1$. Thus, $k - j \le k - 1$. On the other hand, since P contains three vertices v_0, v_j, and v' on C_k, the length of P is at least $2\frac{k}{2} = k$, which contradicts that the length of P is shorter than $k - j \le k - 1$. Thus, for each pair u, v in V, $d_{G'}(u, v)$ is equal to the length of a longest path joining u and v in G, and the shortest path joining u and v on G' is uniquely determined. In each gadget in Figure 4, we replace each path by the edge of weight equal to the length of the path. This reduction can be done in $O(|V|^2)$ time and space. ∎

4 Algorithms for biconvex graphs

In this section, we consider the longest path problem on subclasses of biconvex graphs. We assume that a biconvex graph $G = (X, Y, E)$ is given with its compact interval representation. For given any path P, we denote by P_X the set of vertices in P and X, and by P_Y the set of vertices in P and Y. We assume that the sets P_X and P_Y are ordered; when $P_X = \{x_{i_1}, x_{i_2}, \ldots\}$ and $P_Y = \{y_{j_1}, y_{j_2}, \ldots\}$, then $P = (x_{i_1}, y_{j_1}, x_{i_2}, y_{j_2}, \ldots)$ or $P = (y_{j_1}, x_{i_1}, y_{j_2}, x_{i_2}, \ldots)$.

We first assume that given biconvex graph is a bipartite permutation graph; in our context, we deal it as a "proper" biconvex graph as depicted in the left hand in Figure 2.

Lemma 13 For given bipartite permutation graph $G = (X, Y, E)$, there is a longest path P with $P_X = \{x_{i_1}, x_{i_2}, \ldots\}$ and $P_Y = \{y_{j_1}, y_{j_2}, \ldots\}$ such that $i_k < i_{k+1}$ and $j_k < j_{k+1}$ for each k.

Proof. Let P′ be any longest path $(x_{i'_1}, y_{j'_1}, x_{i'_2}, y_{j'_2}, \ldots)$ such that $x_i \in X$ and $y_j \in Y$ (the symmetric case that P′ starts from a vertex in Y is omitted). We construct P from P′ such that P contains the same vertices in P′ and satisfies the condition. The proof is done by the induction for the length of the path. The lemma holds when the length of the path is 2. Thus we assume that the length of the path is at least 3. We have two cases:
(1) The path ends two vertices $(x_{i'_k}, y_{j'_k})$ with $x_{i'_k} \in X$ and $y_{j'_k} \in Y$. Then, by the inductive hypothesis, there is a path $P'' = (x_{i_1}, y_{j_1}, x_{i_2}, y_{j_2}, \ldots, x_{i_k}, y_{j_k})$ satisfying the condition. If either $y_{j_k} \prec y_{j'_{k+1}}$ or $y_{j'_{k+1}} \prec y_{j_1}$, the path $(x_{i_1}, y_{j_1}, \ldots, x_{i_k}, y_{j'_{k+1}}, x_{i'_{k+1}})$ or $(x_{i'_{k+1}}, y_{j'_{k+1}}, x_{i_1}, y_{j_1}, \ldots, x_{i_k})$ satisfies the condition. Thus we suppose that $y_{j_1} \prec y_{j'_{k+1}} \prec y_{j_k}$. Then since the graph is proper, there is an index h such that $y_{j_h} \prec y_{j'_{k+1}} \prec y_{j_{h+1}}$. Then $x_{i_{h+1}} \in I_{y_{j'_{k+1}}}$, and $x_{i'_{k+1}} \in (I_{y_{j_h}} \cup I_{y_{j_{h+1}}})$. If $x_{i'_{k+1}} \in I_{y_{j_h}}$, the path $(x_{i_1}, y_{j_1}, x_{i_2}, y_{j_2}, \ldots, y_{j_h}, x_{i'_{k+1}}, y_{j'_{k+1}}, x_{i_{h+1}}, y_{j_{h+1}}, \ldots, x_{i_k}, y_{j_k})$ satisfies the condition. Otherwise, the path $(x_{i_1}, y_{j_1}, x_{i_2}, y_{j_2}, \ldots, y_{j_h}, x_{i_{h+1}}, y_{j'_{k+1}}, x_{i'_{k+1}}, y_{j_{h+1}}, \ldots, x_{i_k}, y_{j_k})$ satisfies the condition.
(2) The path ends two vertices $(y_{j'_k}, x_{i'_{k+1}})$ with $y_{j'_k} \in Y$ and $x_{i'_{k+1}} \in X$. Then, by the inductive hypothesis, there is a path $P'' = (x_{i_1}, y_{j_1}, x_{i_2}, y_{j_2}, \ldots, x_{i_k}, y_{j_k})$ satisfying the condition. Then we have three possible cases; $y_{j_k} \prec y_{j'_k}$, $y_{j'_k} \prec y_{j_1}$, and $y_{j_1} \prec y_{j'_k} \prec y_{j_k}$. Using the same argument in (1), we have the lemma. ∎

Theorem 14 For given bipartite permutation graph $G = (X, Y, E)$, a longest path can be found in $O(|X \cup Y| + |E|)$ time.

Proof. To compute the longest path satisfying the condition in Lemma 13, we use a standard dynamic programming. We define two functions $f(x_i, y_j)$ and $g(y_j, x_i)$ such that $f(x_i, y_j)$ gives the length of a longest path starting at the edge $\{x_i, y_j\}$ and $g(y_j, x_i)$ gives the length of a longest path starting at the edge $\{y_j, x_i\}$. We define $f(x_i, y_j) = 0$ if $\{x_i, y_j\} \notin E$ or y_j does not exist, and $g(y_j, x_i) = 0$ if $\{x_i, y_j\} \notin E$ or x_i does not exist. An example is depicted in Figure 5. Then, if $\{x_i, y_j\} \in E$,

$$\begin{aligned} f(x_i, y_j) &= \max\{f(x_i, y_{j+1}), g(y_j, x_{i+1}) + 1\} \text{ and} \\ g(y_j, x_i) &= \max\{f(x_i, y_{j+1}) + 1, g(y_j, x_{i+1})\}. \end{aligned}$$

Consequently, the length of a longest path is given by $\max\{f(x_1, y_1), g(y_1, x_1)\}$. Computing the longest path itself is also straightforward. In Figure 5, $f(x_1, y_1) = 10$ and $g(y_1, x_1) = 11$. Hence the length of a longest path is 11, which is given by $(y_1, x_1, y_2, x_2, y_3, x_3, y_4, x_4, y_5, x_5, y_6, x_6)$ or $(y_1, x_1, y_2, x_2, y_3, x_3, y_4, x_4, y_5, x_5, y_6, x_7)$. The computation of $\max\{f(x_1, y_1), g(y_1, x_1)\}$ can be done in a standard dynamic programming in linear time and space. ∎

By Theorems 5 and 14, we can find a longest path in a chain graph in linear time. On the other hand, by Observation 4 and Theorem 5, any biconvex graph G consists of three bipartite permutation graphs G_1, G_2, and G_3. We conjecture that the longest path among three longest paths in G_i with $i = 1, 2, 3$ has at least one third length of a longest path in G.

5 Algorithm for interval biconvex graphs

Let $G = (V = S \cup Y, E)$ be an interval biconvex graph with the set S of simplicial vertices. Let $x_1 < x_2 < \cdots < x_{n_x}$ and $y_1 < y_2 < \cdots < y_{n_y}$ be the orderings over S and Y that have adjacency property. We here denote by $I(x_i)$ the integer point I_{x_i}, that is, $I(x_i) = R(I_{x_i}) = L(I_{x_i})$. By Lemma 6, for each x and x' with $N[x] = N[x']$, $I(x) = I(x')$. For each j with $1 \leq j \leq n_y$, since each y_j contains consecutive xs, we can let y_j correspond to the interval $[L(y_j), R(y_j)]$ such that $L(y_j) = I(x_\ell)$ and $R(y_j) = I(x_r)$, where x_ℓ and x_r are the minimum and maximum vertices in $N(y_j)$, respectively. By definition and the proof of Lemma 2, we immediately have the following lemma:

Lemma 15 For the compact interval representation $\mathcal{I}(G)$ of the interval biconvex graph $G = (V = S \cup Y, E)$ with the set S of simplicial vertices, there are two indices j_t and j_b such that (0) $j_t \leq j_b$, (1) $L(y_1) \geq L(y_2) \geq \cdots \geq L(y_{j_t})$ and $L(y_{j_t}) \leq L(y_{j_t+1}) \leq \cdots \leq L(y_{n_y})$, and (2) $R(y_1) \leq R(y_2) \leq \cdots \leq R(y_{j_b})$ and $R(y_{j_b}) \geq R(y_{j_t+1}) \geq \cdots \geq R(y_{n_y})$.

The following proposition is also immediate (see also [5, Lemma 2]):

Proposition 16 Let $G = (V = S \cup Y, E)$ be a connected interval biconvex graph. Let $y_1 < y_2 < \cdots < y_{n_y}$ be the orderings over Y that have adjacency property. Then $\{y_i, y_{i+1}\} \in E$ for each $1 \leq i < n_y$. That is, $(y_1, y_2, \ldots, y_{n_y})$ is a path of G.

Hereafter, we denote by (X, y) a path $(x_i, x_{i+1}, \ldots, x_{i+j}, y)$ for a set $X = \{x_i, x_{i+1}, \ldots, x_{i+j}\}$ if $G[X]$ is a clique.

Lemma 17 For a connected interval biconvex graph $G = (V = S \cup Y, E)$, there is a longest path P such that
(1) the vertices in $Y \cap \mathsf{P}$ appear consecutively; that is, $Y \cap \mathsf{P}$ is $y_j, y_{j+1}, \ldots, y_{j+k-1}, y_{j+k}$ for some j and k, and those vertices appear according to the ordering over Y,
(2) the consecutive vertices in S on P correspond to the same integer point; that is, if P contains a subpath $(y_j, x_i, x_{i+1}, x_{i+2}, \ldots, x_{i+h}, y_{j+1})$, we have $I(x_i) = I(x_{i+1}) = \cdots = I(x_{i+h})$,
(3) the vertices in S on P appear according to the ordering over S; that is, if $S \cap \mathsf{P}$ is $x_{i_1}, x_{i_2}, \ldots, x_{i_h}$ in this order, $I(x_{i_1}) \leq I(x_{i_2}) \leq \cdots \leq I(x_{i_h})$, and
(4) P starts and ends at the vertices in S.

Let s and f be integers such that $N[s] \cap S$ and $N[f] \cap S$ give us the set of vertices satisfying the condition that P starts at $N[s]$ and ends at $N[f]$. Let y_s and y_f be the vertices such that P starts with $(N[s], y_s)$ and ends with $(y_f, N[f])$. Then
(5) y_s is the minimum vertex in Y with I_{y_s} contains s, and y_f is the maximum vertex in Y with I_{y_f} contains f.

Proof. (1) We assume that P contains y_{j_1}, y_{j_2}, and y_{j_3} in this ordering and $y_{j_1} < y_{j_3} < y_{j_2}$. Then we have five cases; (a) $I(y_{j_1}) \subseteq I(y_{j_3}) \subseteq I(y_{j_2})$, (b) $I(y_{j_1}) \subseteq I(y_{j_3})$ and $y_{j_3} \prec y_{j_2}$, (c) $y_{j_1} \prec y_{j_3} \prec y_{j_2}$, (d) $y_{j_1} \prec y_{j_3}$ and $I(y_{j_2}) \subseteq I(y_{j_3})$, or (e) $I(y_{j_2}) \subseteq I(y_{j_3}) \subseteq I(y_{j_1})$. We assume that they are minimal, that is, they are consecutive in $\mathsf{P} \cap Y$. Then, in any case, swapping y_{j_2} and y_{j_3} we have also a path. Repeating this process, we have a path such that the vertices in Y appear according to the ordering over Y. Let y_j and y_{j+k} be the first and last vertices in $\mathsf{P} \cap Y$, respectively. We next show that all vertices $y_{j+k'}$ with $0 < k' < k$ appear on P. We assume that some $y_{j+k'}$ with $0 < k' < k$ does not appear on P. Then, inserting it, we have a longer path, which contradicts that P is a longest path.
(2) Let s be any integer such that there is a vertex x in $N[s] \cap S \cap \mathsf{P}$. If there is a vertex x' such that $x' \in N[s] \cap S$ and x' does not appear in P, we have longer path by replacing a subpath (x) of P by (x, x'). Thus all vertices in $N[s] \cap S$ appear in P. It is clear that gathering all vertices in $N[s] \cap S \cap \mathsf{P}$ has no effects on the connectivity and length of P. Thus we can assume that all vertices in $N[s] \cap S$ are consecutive in P.
(3) By (1), all vertices in $Y \cap \mathsf{P}$ appear consecutively. Thus, using the same argument in (1), we can assume that all vertices in $S \cap \mathsf{P}$ also appear consecutively.
(4) To derive a contradiction, we assume that P starts at the vertex y_j in S. Then, by the definition of a compact interval representation, we have (a) $L(y_j) = R(y_{j'})$ for some $j' < j$, or (b) $N[L(y_j)] \cap S \neq \emptyset$. In the case (a), we have longer path by adding $y_{j'}$ at the top of P. On the other hand, in the case (b), we also have longer path by adding the vertices in $N[L(y_j)] \cap S$ at the top of P. Thus, P starts at the vertex in S. Using the same argument, we can show that P ends at the vertex in S.
(5) To derive a contradiction, we assume that P starts with $(N[s], y_s)$ and there is a vertex $y_{s'}$ such that $s' \neq s$, $I_{y_{s'}}$ contains s, and $y_{s'} \prec y_s$. By (1), $y_{s'}$ does not appear in P. However, in the case, we have longer path by replacing $(N[s], y_s)$ by $(N[s], y_{s'}, y_s)$, which is a contradiction. Thus y_s is the minimum vertex in Y with I_{y_s} contains s. Using the same argument, we can show that y_f is the maximum vertex in Y with I_{y_f} contains f. ∎

By Lemma 17, the outline of our algorithm is as follows:

0. for each integer s and f with $0 < s < f \leq n_x$, suppose $N[s] \cap S$ and $N[f] \cap S$ are the endpoints of a longest path;
1. let y_{j_s} be the smallest vertex with $L[y_{j_s}] \leq s \leq R[y_{j_s}]$, and let y_{j_f} be the largest vertex with $L[y_{j_f}] \leq f \leq R[y_{j_f}]$;
2. for each integer $i = s+1, s+2, \ldots, f-1$, determine where the vertices in $S \cap N[i]$ are inserted in the path $(N[s], y_{j_s}, y_{j_s+1}, \ldots, y_{j_f-1}, y_{j_f}, N[f])$.

Step 2 can be implemented using the polynomial time algorithm for the maximum weighted matching problem. We first construct a weighted bipartite graph $G' = (X', Y', E')$ with $X' = \{i \mid I(x) = i$ for some $x \in S$ with $s < i < f\}$ and $Y' = \{\{y_j, y_{j+1}\} \mid y_j, y_{j+1} \in Y$ and $j_s \leq j \leq j_f - 1\}$ as follows; E' contains a weighted edge $e = \{i, \{y_j, y_{j+1}\}\}$ if $N[i]$ contains both of y_j and y_{j+1}. The weight of the edge e is defined by $|S \cap N[i]|$, which is the number of vertices x in S with $I(x) = i$. A matching M of G' gives us a path of G as follows; if an edge $e = \{\{y_j, y_{j+1}\}, i\}$ is in M, the path of G contains $(y_j, N[i], y_{j+1})$. Thus, by Lemma 17, the maximum weighted matching of G' gives us a longest path of G. The detail of the algorithm can be described as follows:

Algorithm 3: IBG

```
    Input  : An interval biconvex graph $G = (V = S \cup Y, E)$ with $|S| = n_x$ and $|Y| = n_y$, and a
             compact interval representation $\mathcal{I}(G)$.
    Output: A longest path P.
1   construct the weighted bipartite graph $G' = (X', Y', E')$ for $G$;
2   for each integer $s$ and $f$ with $0 < s < f \leq n_x$ do
3       let $N[s]$ and $N[f]$ suppose the endpoints of a longest path;
4       let $y_{j_s}$ be the smallest vertex with $L[y_{j_s}] \leq N[s] \leq R[y_{j_s}]$, and let $y_{j_f}$ be the largest vertex
        with $L[y_{j_f}] \leq N[f] \leq R[y_{j_f}]$;
5       let $G''$ be the induced bipartite graph from $G'$ by the vertices in $S$ between $s$ and $f$ and the
        vertices in $Y$ between $y_{j_s}$ and $y_{j_f}$;
6       find a maximum weighted matching $M$ in $G''$;
7       compute the path given by $(N[s], y_{j_s}, y_{j_s+1}, \ldots, y_{j_f-1}, y_{j_f}, N[f])$ and the edges in $M$;
8   end
9   find the longest path among the paths generated in step 7.
```

Theorem 18 A longest path in a given interval biconvex graph $G = (V = S \cup Y, E)$ with $|V| = n$ and $|E| = m$ can be found in $O(n^3(m + n \log n))$ time.

Proof. The correctness of Algorithm IBG follows from Lemma 17. Thus we analyze the running time. For the weighted bipartite graph $G' = (X', Y', E')$ constructed in step 1, we have $|X'| = O(n_x)$, $|Y'| = n_y - 1$, and $|E'| = O(|E|)$. A maximum weighted matching can be found in $O(|V|(|E| + |V| \log |V|))$ time and $O(|E|)$ space for any given graph $G = (V, E)$ [16]. Hence step 6 takes $O(n(m + n \log n))$ time with $n = n_x + n_y$ and $m = |E|$. Therefore total running time is $O(n_x^2 n(m + n \log n)) = O(n^3(m + n \log n))$. ∎

Corollary 19 A longest path in a given threshold graph $G = (V, E)$ with $|V| = n$ and $|E| = m$ can be found in $O(n + m)$ time and space.

Proof.(Sketch.) By Lemma 8 and Theorem 18, a longest path in a threshold graph can be found in $O(n^3(m + n \log n))$ time. However, Algorithm IBG can be simplified to run linear time and space using the properties shown in the proof of Lemma 8 and that $G[S]$ is an independent set. Similar idea can be found in [26], and hence omitted here. ∎

6 Concluding remarks

The main open problems are the complexity of the longest path problem for interval graphs, convex graphs, and biconvex graphs.

The longest path problem for interval graphs can be reduced to the longest path problem for convex graphs in polynomial time by using a simple reduction by Damaschke in [28, Section 2] (see Appendix A). Thus the problem for convex graphs seems to be more difficult than interval graphs. Intuitively, however, a convex graph has similar structure to the interval graph that has the $\mathcal{MPQ}$-tree with only one $\mathcal{Q}$-node in the manner of [24]. Therefore, the complexity of the longest path problem for interval graphs and convex graphs seems to be essentially the same. The longest path problems for biconvex graphs seems to be easier than the other two graph classes.

It is also worth investigating for the complexity of the longest cycle problem instead of the longest path problem. It is known that a longest cycle can be found in threshold graphs in polynomial time [26]. The efficient algorithms for the Hamiltonian cycle problem for proper interval graphs [5, 30] and interval graphs [22] have been given. There are a few results for the Hamiltonian cycle problem on split graphs [12, 31]. Dynamic programming technique seems to be effective for some graph classes that have tree decomposition (e.g., partial k-trees).

7 Acknowledgment

The authors would like to thank Prof. Hiro Ito, who pointed out a flaw of the algorithm for ptolemaic graphs in an early paper [34].

References

[1] N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995.

[2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation.* Springer, 1999.

[3] P.N. Balister, E. Györi, J. Lehel, and R.H. Schelp. Longest paths in circular arc graphs. Technical report, University of Memphis, `http://www.msci.memphis.edu/preprint.html`, 2002.

[4] C. Berge. *Hypergraphs.* Elsevier, 1989.

[5] A.A. Bertossi. Finding Hamiltonian circuits in proper interval graphs. *Information Processing Letters*, 17(2):97–101, 1983.

[6] A.A. Bertossi and M.A. Bonuccelli. Hamiltonian circuits in interval graph generalizations. *Information Processing Letters*, 23:195–200, 1986.

[7] A. Björklund and T. Husfeldt. Finding a path of superlogarithmic length. *SIAM Journal on Computing*, 32(6):1395–1402, 2003.

[8] K.S. Booth and G.S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using *PQ*-tree algorithms. *Journal of Computer and System Sciences*, 13:335–379, 1976.

[9] A. Brandstädt, V.B. Le, and J.P. Spinrad. *Graph Classes: A Survey.* SIAM, 1999.

[10] A. Brandstädt, J. Spinrad, and L. Stewart. Bipartite permutation graphs are bipartite tolerance graphs. *Congressus Numerantium*, 58:165–174, 1987.

[11] R.W. Bulterman, F.W. van der Sommen, G. Zwaan, T. Verhoeff, A.J.M. van Gasteren, and W.H.J. Feijen. On computing a longest path in a tree. *Information Processing Letters*, 81:93–96, 2002.

[12] R.E. Burkard and P.L. Hammer. A note on Hamiltonian split graphs. *Journal of Combinatorial Theory, Series B*, 28:245–248, 1980.

[13] P. Damaschke. The Hamiltonian circuit problem for circle graphs is NP-complete. *Information Processing Letters*, 32:1–2, 1989.

[14] P. Damaschke. Paths in interval graphs and circular arc graphs. *Discrete Mathematics*, 112:49–64, 1993.

[15] R.G. Downey and M.R. Fellows. *Parameterized Complexity.* Springer, 1999.

[16] H.N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *Proc. 1st Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 434–443. ACM, 1990.

[17] M.R. Garey and D.S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness.* Freeman, 1979.

[18] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs,* 2nd edition. Annals of Discrete Mathematics 57. Elsevier, 2004.

[19] P. Hell and J. Huang. Interval bigraphs and circular arc graphs. *J. of Graph Theory*, to appear. available at `http://www.cs.sfu.ca/ pavol/intBig.ps`.

[20] D. Hochbaum (eds.). *Approximation Algorithms for NP-hard Problems.* PWS Publishing Company, 1995.

[21] D. Karger, R. Motwani, and G.D.S. Ramkumar. On Approximating the longest path in a graph. *Algorithmica*, 18:82–98, 1997.

[22] J.M. Keil. Finding Hamiltonian circuits in interval graphs. *Information Processing Letters*, 20(4):201–206, 1985.

[23] T. Kloks, D. Kratsch, and H. Müller. Bandwidth of chain graphs. *Information Processing Letters*, 68:313–315, 1998.

[24] N. Korte and R.H. Möhring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM Journal on Computing*, 18(1):68–81, 1989.

[25] T.-H. Lai and S.-S. Wei. Bipartite permutation graphs with applications to the minimum buffer size problem. *Discrete Applied Mathematics*, 74:33–55, 1997.

[26] N.V.R. Mahadev and U.N. Peled. Longest cycles in threshold graphs. *Discrete Mathematics*, 135:169–176, 1994.

[27] B. Monien. How to find long paths efficiently. *Annals of Discrete Mathematics*, 25:239–254, 1985.

[28] H. Müller. Hamiltonian circuit in chordal bipartite graphs. *Disc. Math.*, 156:291–298, 1996.

[29] H. Müller. Recognizing interval digraphs and interval bigraphs in polynomial time. *Disc. Appl. Math.*, 78:189–205, 1997. Erratum is available at `http://www.comp.leeds.ac.uk/hm/pub/node1.html`.

[30] B.S. Panda and S.K. Das. A linear time recognition algorithm for proper interval graphs. *Information Processing Letters*, 87:153–161, 2003.

[31] J. Peemöller. Necessary conditions for Hamiltonian split graph. *Discrete Mathematics*, 54:39–47, 1985.

[32] M.G. Scutellà. An approximation algorithm for computing longest paths. *European Journal of Operational Research*, 148(3):584–590, 2003.

[33] J. Spinrad, A. Brandstädt, and L. Stewart. Bipartite permutation graphs. *Discrete Applied Mathematics*, 18:279–292, 1987.

[34] R. Uehara and Y. Uno. Efficient algorithms for the longest path problem. In *15th Annual International Symposium on Algorithms and Computation (ISAAC 2004)*, pages 871–883. Lecture Notes in Computer Science Vol.3341, Springer-Verlag, 2004.

[35] V.V. Vazirani. *Approximation Algorithms.* Springer, 2001.

[36] S. Vishwanathan. An approximation algorithm for finding a long path in Hamiltonian graphs. In *Proc. 11th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 680–685. ACM, 2000.

[37] M. Yannakakis. Node-deletion problems on bipartite graphs. *SIAM Journal on Computing*, 10(2):310–327, 1981.

A Related results

Using a simple reduction by Damaschke in [28, Section 2], we have the following two theorems:

Theorem 20 The longest path problem for interval graphs can be reduced to the longest path problem for convex graphs in $O(n^2)$ time.

Proof. Given connected interval graph $G = (V, E)$ with $|V| = n$, we construct a corresponding convex graph $G' = (V, W, E')$ such that a longest path in G gives a longest path in G' or vice versa. For G, we first construct a compact interval representation $\mathcal{I}$. Then the set W is defined as follows; for each integer i in $[1, n]$, W contains $|N[i]| + 1$ vertices corresponding to the point i. The ordering of W is defined naturally; $w < w'$ if $I(w) < I(w')$. The set E' is the set of edges $\{v, w\}$ if and only if I_v contains the point $I(w)$. The resulting bipartite graph is clearly convex graph, and the reduction can be done in $O(n^2)$ time. Thus it is sufficient to show that a longest path in G corresponds to a longest path in G'. Let $\mathsf{P} = (v_1, v_2, \ldots, v_k)$ be a path in G. Then, by definition of compact interval representation, $I_{v_i} \cap I_{v_{i+1}}$ contains at least one integer point for each $i = 1, \ldots, k-1$. Moreover, each integer point i corresponds to $|N[i]| + 1$ ws. Thus, on G', we can construct a path $\mathsf{P}' = (w_1, v_1, w_2, v_2, \ldots, w_k, v_k, w_{k+1})$ of length $2k + 1$. It is easy to see that P is a longest path in G if and only if P' is a longest path in G'. ∎

Theorem 21 We can solve the Hamiltonian path problem for biconvex graphs $G = (X, Y, E)$ in $O(n^2)$ time, where $n = |X \cup Y|$.

Proof. Let $G = (X, Y, E)$ be a given biconvex graph such that X and Y are ordered with respect to the adjacency property. By Lemma 1, each $x \in X$ on the compact interval representation $\mathcal{I}(G)$ corresponds to an integer point.

When $|X| = |Y|$, we can use the same idea in [28, Section 2] which is as follows: From G, we construct the interval graph $G' = (X \cup Y, E \cup E')$ where $E' := \{\{y_1, y_2\} \mid N(y_1) \cap N(y_2) \neq \emptyset\}$. We let $m = |E \cup E'|$. Then, since two vertices in X cannot be consecutive on a path of G', G has a Hamiltonian path if and only if G' has a Hamiltonian path.

However, G can have a Hamiltonian path when $||X| - |Y|| \leq 1$. Thus we have to consider two other cases. We first assume that $|X| - |Y| = 1$. Two vertices in X cannot consecutive even in G'. Thus if G' has a Hamiltonian path, it starts from a vertex in X and ends at a vertex in X. Hence the reduction above again works; G has a Hamiltonian path if and only if G' has a Hamiltonian path.

If $|Y| - |X| = 1$, we can swap X and Y and the problem is reduced to the case $|X| - |Y| = 1$. ∎

Corollary 22 Let $G = (X, Y, E)$ be a convex graph such that an ordering $<$ of X has the adjacency property. Then the Hamiltonian path problem for G can be solved in $O(n^2)$ time if $|X| = |Y|$ or $|X| - |Y| = 1$.

Remark: The Hamiltonian path problem for the convex graph $G = (X, Y, E)$ with $|Y| - |X| = 1$ is open. It seems that the case is related to the problem named "Hamiltonian path with fixed end" (1HP) proposed by Damaschke in [14], which is also open. If 1HP has a polynomial time algorithm, the Hamiltonian path problem for the convex graph $G = (X, Y, E)$ with $|Y| - |X| = 1$ also can be solved in polynomial time.