ON THE MAXIMUM CONNECTED INTERVAL SUBGRAPH OF BLOCK GRAPHS AND CHAIN GRAPHS

*Sheng-Lung Peng**, *Hsiao-An Chao*, *Ruay-Shiung Chang*

Department of Computer Science and information engineering,
National Dong Hwa University, Hualien, Taiwan, R.O.C.
Email:{lung, m8721011, rschang}@csie.ndhu.edu.tw

Abstract

The maximum connected interval subgraph problem on a graph G is the problem of finding an induced subgraph H of G such that H is a connected interval graph with the maximum number of vertices. It has been shown that this problem is NP-hard and have no constant ratio approximation on general graphs. In this paper, we propose linear-time algorithms for solving this problem on block graphs and chain graphs.

1. Introduction

Let G be a finite and simple graph with vertex set $V(G)$ and edge set $E(G)$. A graph G is an *interval graph* if there exists a one-to-one correspondence between $V(G)$ and a family F of intervals of the real line such that two vertices in $V(G)$ are adjacent if and only if their corresponding intervals in F overlap. Interval graphs have many applications, among them scheduling, seriation in archeology, medical diagnosis, behavioral psychology, circuit design and most recently the Human Genome Project [4–6,9,14].

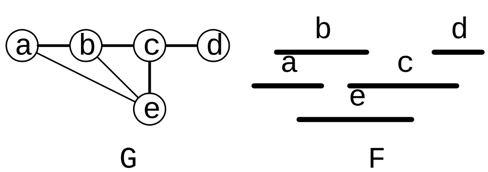


Figure 1. An interval graph and its corresponding interval family.

Recently, many researchers focus on finding an interval supergraph of the input graph with different requirements [2,3,10,8,15]. In this paper, we deal with the problem of finding an interval subgraph of the input graph with the maximum size. The maximum interval subgraph problem ($MISP$ for short) on G is the problem of finding a maximum interval subgraph of G by deleting the minimum number of vertices of G. A more generalized problem is referred as the node deletion problem [11,12,17]. Similarly, the maximum connected interval subgraph problem ($MCISP$ for short) on G is the problem of finding a maximum connected interval subgraph of G by deleting the minimum number of vertices of G. Note that the solution of MISP may not contain the solution of MCISP for a graph. For example, please see Figure 2.

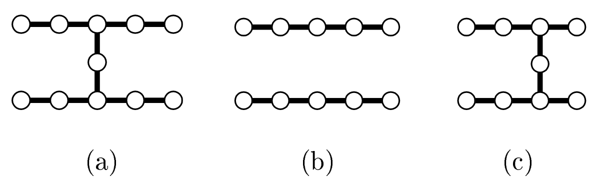


Figure 2. (a) A graph G. (b) A solution of MISP. (c) A solution of MCISP.

It has been shown that MCISP is NP-hard on general graphs [16]. Furthermore, it has also been shown that it is impossible to be approximated with ratio $n^{1-\epsilon}$, for every $\epsilon > 0$, in polynomial time unless $P = NP$, where n is the number of vertices in the input graph [13]. So far, to our knowledge, there is no result for MCISP on any special graphs.

In this paper, we solve this problem on block graphs and chain graphs. On a block graph, we show that this problem can be reduced to the problem of finding a maximum caterpillar (with a constraint) in its block tree. On a chain graph, we show that this problem is equivalent to the problem of finding a maximum caterpillar of this graph. As a result, we propose linear-time algorithms for solving this problem on above graphs.

*corresponding author: lung@csie.ndhu.edu.tw

The rest of this paper is organized as follows. In Section 2, we give the definitions and notation used in this paper. The main results on block graphs and chain graphs are presented in Sections 3 and 4, respectively. Finally, we give the concluding remarks in the last section.

2. Definitions and Notation

Let G be a connected, simple and finite graph. Let $V(G)$ and $E(G)$ be its vertex and edge sets, respectively. Let $n = |V(G)|$ and $m = |E(G)|$. Let $N(v) = \{w \mid (v, w) \in V(E)\}$ denote the neighborhood of vertex v. For a vertex set $W \subseteq V(G)$, let $G[W]$ denote the subgraph of G induced by W. For two vertex sets X and Y, $X \backslash Y = \{v \in X \mid v \notin Y\}$. In the following, "subgraph" means "induced subgraph."

A *clique* in a graph G is a complete subgraph of G. A clique also refers to a set of vertices whose induced subgraph is complete when there is no confusion in the description. An *independent* set I in G is a vertex subset of $V(G)$ in which no two vertices are adjacent. For a graph G, a sequence of vertices $\langle v_1, v_2, ..., v_r\rangle$ is a *path* if $(v_i, v_{i+1}) \in E(G)$, $1 \leq i \leq r-1$.

For a tree T, a vertex with degree 1 is called a *leaf* and a non-leaf vertex is called an *internal vertex*. A tree T is called a *caterpillar* if $V(T)$ can be partitioned into two sets B and H such that vertices in B induce a path and vertices in H induce an independent set. The path is also called the *backbone* of this caterpillar. We call a vertex in B (respectively, H) a *backbone vertex* (respectively, *hair vertex*).

3. Block Graphs

In this section, we discuss MCISP on block graphs. For any graph G, a vertex v is called a *cut vertex* if deleting v increases the number of connected components. A *block* is a maximal connected subgraph of G without any cut vertex. A graph is called a *block graph* if and only if its blocks are complete graphs and the intersection of two blocks is either empty or a cut vertex [7]. Note that trees form a subclass of block graphs. In the following, G is treated as a connected block graph.

Suppose G has m blocks $B_1, B_2, \ldots, B_m$ and n cut vertices $v_1, v_2, \ldots, v_n$. The *block cut vertex structure* T_G is the tree with vertex set $V(T_G) = \{B_1, B_2, \ldots, B_m, v_1, v_2, \ldots, v_n\}$ and edge set $E(T_G) = \{(B_i, v_j) \mid 1 \leq i \leq m, 1 \leq j \leq n, v_j \in B_i\}$. Vertices B_i, $1 \leq i \leq m$, are called *block vertices* and vertices v_j, $1 \leq j \leq n$, are called *cut vertices*. For convenience, T_G is also called the *block tree* of G. For example, Figure 4 shows the block tree of the graph G depicted in Figure

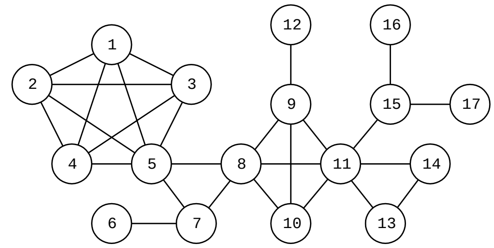
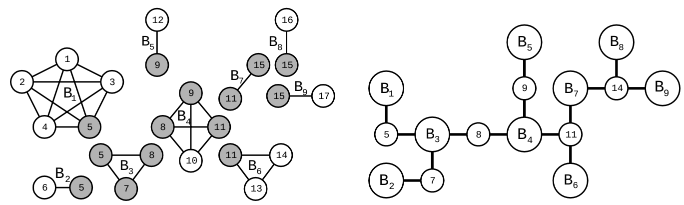


Figure 3. A block graph G with 17 vertices.

3. The block tree of a block graph can be constructed in linear time by a depth first search [1]. For a block vertex u in a block tree, let B_u denote its corresponding block in G. Note that every leaf of a block tree is a block vertex.



(a) (b)

Figure 4. (a) The 9 blocks of G where a dark vertex denotes a cut vertex. (b) The block tree of G.

For a block graph G, we assume that T_G is a caterpillar. By the definition, T_G can be partitioned into two sets B and H. Without loss of generality, let $B = \langle v_{10}, v_{20}, \ldots, v_{p0}\rangle$ be the backbone of T_G, where $v_{10}, v_{30}, \ldots, v_{p0}$ are block vertices and $v_{20}, v_{40}, \ldots, v_{(p-1)0}$ are cut vertices. Note that every vertex in H must be a block vertex. Therefore, it is impossible to have a hair vertex who is adjacent to a block vertex in B since T_G is a caterpillar. Let d_i be the number of the neighbors of v_{i0} in H for $i \in \{1, \ldots, p\}$. Let $v_{i1}, v_{i2}, \ldots, v_{id_i}$ be the block vertices which are the neighbors of v_{i0} in H. For a vertex u in G, we construct an interval I_u as follows. If u is not a cut vertex and belongs to block $B_{v_{ij}}$, let $I_u = (l_{ij}, l_{ij} + 0.5)$ where

$$l_{ij} = \begin{cases} 1 & if\ i = 1 \\ l_{(i-1)d_{i-1}} + 1 & if\ i \neq 1, j = 0 \\ l_{i(j-1)} + 1 & if\ i \neq 1, j > 0 \end{cases}$$

Otherwise, u is a cut vertex. In this case, assume that $u = v_{i0}$ for some i. Then $I_u = (l_{(i-1)0}, l_{(i+1)0} + 0.5)$.

Let $F = \{I_u \mid u \in G\}$. It is not hard to see that there is a one-to-one correspondence between $V(G)$ and F such that u and w are adjacent if and only if their corresponding intervals I_u and I_w intersect. That is, G is a connected interval graph.

On the other hand, if T_G is not a caterpillar, then G is impossible to be an interval graph. Hence, we have the following theorem.

Theorem 3.1 *A block graph G is a connected interval graph if and only if T_G is a caterpillar.*

According to Theorem 3.1, MCISP on a block graph G is equivalent to the problem of finding the maximum subgraph G' of G such that $T_{G'}$ is a caterpillar. In other words, our problem can be reduced to the problem of finding a maximum caterpillar in T_G such that every hair vertex is a block vertex.

In the following, we treat T_G as a rooted tree. A maximum caterpillar of a rooted block tree is also a maximum caterpillar of its underlying unrooted block tree. For a rooted tree T and a vertex u, let $T[u]$ denote the subtree of T rooted at u. For convenience, we use $G[T_G[u]]$ to denote the block graph with block tree $T_G[u]$. Therefore, if C is a caterpillar of T_G, then $G[C]$ is the block graph with C being its block tree. For a rooted caterpillar, we define the following two types on its backbone vertices u.

Type I: u has at most one child who is also a backbone vertex.

Type II: u has exactly two children who are also backbone vertices.

Note that for a rooted caterpillar, there is at most one backbone vertex which is Type II.

For a vertex u in T_G, let $type1(u)$ (respectively, $type2(u)$) denote $|V(G[C])|$ if there is a maximal caterpillar C in $T_G[u]$ with u being a Type I (respectively, Type II) vertex. In the case that u cannot be a Type II vertex, we let $type2(u) = 0$. Algorithm MCISB (Figure 5) first computes ($type1(u)$,$type2(u)$) for every vertex u in T_G from leaves to the root and then finds the $\max_{v \in V(T_G)}\{type1(v), type2(v)\}$ which is the number of vertices of the maximum connected interval subgraph of G.

By the definitions, for a leaf u in T_G, $type1(u) = |B_u|$ and $type2(u) = 0$. Furthermore, for an internal vertex u, if u has just one child, then $type2(u) = 0$. Otherwise, we have the following lemmas for an internal block vertex.

Algorithm: MCISB;
Input: A rooted block tree T_G with root u;
Output: The number of vertices of the maximum connected interval subgraph of G;

initially, every vertex v in T_G, $label(v) = (0, 0)$;
$label(u) = compute\_block\_tree\_label(u)$;
find a vertex v such that either $type1(v)$ or $type2(v)$ is maximum;
Output$= \max\{type1(v), type2(v)\}$;

Function: $compute\_block\_tree\_label$(vertex:u)

if u is a leaf **then** $label(u) = (|B_u|, 0)$;
else /* u is not a leaf */
  for all vertices $v_i, 1 \le i \le d$, the d children of u, **do**
    $label(v_i) = compute\_block\_tree\_label(v_i)$;
  next i
  if u is a block vertex **then**
    let $(a_1, b_1), (a_2, b_2), ..., (a_d, b_d)$ be the labels of $v_1, v_2, ..., v_d$, with $a_1 \ge a_2 \ge ... \ge a_d$;
    $type1(u) = a_1 - 1 + |B_u|$;
    if $d = 1$ **then** $type2(u) = 0$
    else $type2(u) = a_1 + a_2 - 2 + |B_u|$;
  else / * u is a cut vertex */
    let $(a_1, b_1), (a_2, b_2), ..., (a_d, b_d)$ be the labels of $v_1, v_2, ..., v_d$, with $a_1 - |B_{v_1}| \ge a_2 - |B_{v_2}| \ge ... \ge a_d - |B_{v_d}|$;
    $type1(u) = a_1 + \sum_{i \in \{2,3,\dots,d\}} (|B_{v_i}| - 1)$;
    if $d = 1$ **then** $type2(u) = 0$
    else $type2(u) = a_1 + a_2 - 1 + \sum_{v \in N(u) \setminus \{v_1, v_2\}} (|B_v| - 1)$;
  end if
  $label(u) = (type1(u), type2(u))$;
end if
return $label(u)$;

Figure 5. Algorithm MCISB.

Lemma 3.1 *If u is an internal block vertex of a block tree T_G, $(a_1, b_1), (a_2, b_2), \dots, (a_d, b_d)$ are the labels of $v_1, v_2, \dots, v_d$, the d children of u, where $a_1 \ge a_2 \ge \dots \ge a_d$, then $type1(u) = a_1 - 1 + |B_u|$.*

Proof. By the definition, $type1(u) = |V(G[C])|$ where C is the maximal caterpillar in $T_G[u]$ including u as a Type I vertex. If u is a block vertex, then $V(G[C])$ consists of two sets, $V(G[C_i])$ and $V(B_u)$ for some i, where $G[C_i]$ is the maximum caterpillar, in $T_G[v_i]$ including v_i as a Type I vertex. Therefore,

$$\begin{aligned} type1(u) &= \max\{|V(G[C_i]) \cup V(B_u)|\} \\ &= \max\{|V(G[C_i])| - 1 + |V(B_u)|\} \\ &= \max\{a_i - 1\} + |B_u| \\ &= a_1 - 1 + |B_u| \end{aligned}$$

□

Lemma 3.2 *If u is an internal block vertex of a block tree T_G, $(a_1, b_1), (a_2, b_2), \dots, (a_d, b_d)$ are the labels of*

$v_1, v_2, \ldots, v_d$, *the* d *children of* u, *where* $a_1 \geq a_2 \geq \ldots \geq a_d$, $d > 1$, *then* $type2(u) = a_1 + a_2 - 2 + |B_u|$.

Proof. By the definition, $type2(u) = |V(G[C])|$ where C is the maximum caterpillar in $T_G[u]$ which includes u as a Type II vertex. Because there are more than one child, the backbone of C must have a subpath $\langle v_i, u, v_j \rangle$, where v_i and v_j are two children of u. If u is a block vertex, $V(G[C])$ consists of three sets, $V(G[C_i])$, $V(G[C_j])$ and $V(B_u)$ for some i and j, where C_i (respectively, C_j) is the maximum caterpillar in $T_G[v_i]$ (respectively, $T_G[v_j]$ including v_i (respectively, v_j) as a Type I vertex. Therefore,

$$\begin{aligned} type2(u) &= \max\{|V(G[C_i]) \cup V(G[C_j]) \cup V(B_u)|\} \\ &= \max\{a_i - 1 + a_j - 1\} + |B_u| \\ &= a_1 + a_2 - 2 + |B_u| \end{aligned}$$

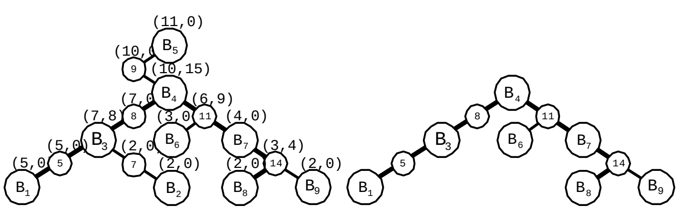
□

Using a similar argument, we have the following lemmas for a cut vertex.

Lemma 3.3 *If* u *is a cut vertex of a block tree* T_G, $(a_1, b_1), (a_2, b_2), \ldots, (a_d, b_d)$ *are the labels of* $v_1, v_2, \ldots, v_d$, *the* d *children of* u, *where* $a_1 - |B_{v_1}| \geq a_2 - |B_{v_2}| \geq \cdots \geq a_d - |B_{v_d}|$, $d > 1$, *then* $type1(u) = a_1 + \sum_{u \in \{v_2, v_3, \ldots, v_d\}} (|B_u| - 1)$

Lemma 3.4 *If* u *is a cut vertex of a block tree* T_G, $(a_1, b_1), (a_2, b_2), \ldots, (a_d, b_d)$ *are the labels of* $v_1, v_2, \ldots, v_d$, *the* d *children of* u, *where* $a_1 - |B_{v_1}| \geq a_2 - |B_{v_2}| \geq \cdots \geq a_d - |B_{v_d}|$, $d > 1$, *then* $type2(u) = a_1 + a_2 - 1 + \sum_{v \in N(u) \setminus \{v_1, v_2\}} (|B_v| - 1)$.

An example for Algorithm MCISB is presented in Figure 6.



(a) (b)

Figure 6. (a) An example of the labeling on a block tree. (b) The maximum caterpillar.

Lemma 3.5 *For a block graph* G, *Algorithm MCISB correctly computes the number of vertices of the maximum connected interval subgraph of* G *in linear time.*

Proof. The correctness of Algorithm MCISB is directly from Lemmas 3.1, 3.2, 3.3 and 3.4. It is not hard to see that the Step 1 of MCISB takes $O(n)$ time. Since the label of each vertex u in T_G can be computed according to the labels of its children, it takes $O(\deg(u))$ time. Therefore, totally, it takes $O(n)$ time to compute all the labels. That is, the Step 2 of MCISB takes $O(n)$ time. By using a standard search algorithm, Step 3 of MCISB can be done in $O(n)$ time. Hence, we have that the time complexity of MCISB is $O(n)$. □

Once the labels of vertices in T_G are computed, we can use them to identify the maximum caterpillar C. First, we find the vertex v with the maximum in $\{type1(v), type2(v)\}$. Then, by a backtracking traversal on $T_G[v]$, we can identify C. Finally, $G[C]$ can be found. This procedure also runs in linear time. Conclusively, we have the following theorem.

Theorem 3.2 *A maximum connected interval subgraph of a block graph can be computed in linear time.*

4. Chain Graphs

In this section, we consider MCISP on chain graphs. Let $G = (X, Y, E)$ be a connected bipartite graph with $X = \{x_1, x_2, \ldots, x_p\}$ and $Y = \{y_1, y_2, \ldots, y_q\}$. The graph G is called a *chain graph* if and only if the neighborhoods of the vertices of X form a *chain*, i.e., the neighborhoods of the vertices of X can be ordered such that $N(x_1) \supseteq N(x_2) \supseteq \cdots \supseteq N(x_p)$. It is not hard to see that the neighborhoods of vertices in Y also forms a chain $(N(y_1) \subseteq N(y_2) \subseteq \cdots \subseteq N(y_q))$ [17]. For example, see Figure 7. By definition, $N(x_1) = Y$ and $N(y_q) = X$.

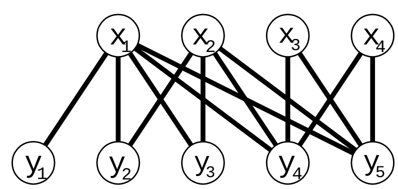


Figure 7. A chain graph.

If a chain graph has at most three vertices, the maximum connected interval subgraph is this chain graph. In the following, we only consider the connected chain graphs with more than three vertices.

Theorem 4.1 *A chain graph* G *is a connected interval graph if and only if* G *is a caterpillar with at most two internal vertices.*

Proof. First, it is not hard to see that if G is not a caterpillar, then G is not an interval graph. Therefore, we consider that G is a caterpillar. If $G = (X, Y, E)$ has at most two internal vertices, it is also not hard to check that G is an interval graph. If G contains more than two internal vertices, then at least two vertices appear in one partite. Without loss of generality, we assume that x_i and x_j are internal vertices in X and $N(x_i) \supseteq N(x_j)$ and $\{y_k, y_l\} \subseteq N(x_j)$. Then the vertices $\{x_i, y_k, x_j, y_l\}$ forms a 4-cycle. This contradicts that G is a caterpillar. Similarly, if at least two internal vertices appear in Y, then we also get a contradiction. Hence, this lemma holds. $\square$

According to Theorem 4.1, our problem on a chain graph G becomes to the problem of finding the maximum caterpillar of G with at most two internal vertices.

Lemma 4.1 *For a chain graph $G = (X, Y, E)$ with $X = \{x_1, x_2, \ldots, x_p\}$ and $Y = \{y_1, y_2, \ldots, y_q\}$, there exists a maximum connected interval subgraph which contains the edge (x_1, y_q).*

Proof. Let $G_1 = (X_1, Y_1, E_1)$ be a maximum connected interval subgraph of G. By Theorem 4.1, G_1 is a caterpillar with at most two internal vertices. We have the following cases.

Case 1: G_1 has two internal vertices. Without loss of generality, let these two vertices be x_i and y_j for some i and j. If $i \neq 1$, then let $G_2 = (X_2, Y_2, E_2)$ where $X_2 = (X_1 \backslash \{x_i\}) \cup \{x_1\}$, $Y_2 = Y_1$ and $E_2 = \{(x, y) \mid x \in X_2, y \in Y_2, (x, y) \in E\}$. It is not hard to check that G_2 is a caterpillar and $|V(G_1)| = |V(G_2)|$. Therefore, G_2 is also a solution of MCISP. Similarly, if $j \neq q$, then we can obtain a graph $G_3 = (X_3, Y_3, E_3)$ from G_2 with $X_3 = X_2$, $Y_3 = (Y_2 \backslash \{y_j\}) \cup \{y_q\}$ and $E_3 = \{(x, y) \mid x \in X_3, y \in Y_3, (x, y) \in E\}$. It can be checked that G_3 is a caterpillar and $|V(G_3)| = |V(G_2)|$. That is, G_3 is also a solution of MCISP with desired condition.

Case 2: G_1 has one internal vertex. Without loss of generality, suppose that x_i is the internal vertex for some i. Then $X_1 = \{x_i\}$ and $Y_1 = Y$. Let $G_2 = (\{x_1\}, Y, \{(x_1, y) \mid y \in Y\})$. Since $N(x_i) \subseteq N(x_1)$, $|V(G_2)| \geq |V(G_1)|$. That is, G_2 is also a solution of MCISP which contains the edge (x_1, y_q).

Note that it is impossible for G_1 to have no internal vertex since $|V(G)| = p + q > 3$. Hence, this lemma holds. $\square$

A solution of the graph depicted in Figure 7 is presented in Figure 8. By using Lemma 4.1, we have an algorithm to solve the MCISP on chain graphs (Figure 9).

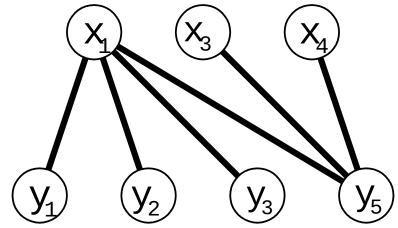


Figure 8. A maximum connected interval subgraph of the chain graph depicted in Figure 7.

Algorithm: MCISC;
Input: a chain graph $G = (X, Y, E)$ with $X = \{x_1, x_2, \ldots, x_p\}$ and $Y = \{y_1, \ldots, y_q\}$;
Output: a maximum connected interval subgraph of G;

```
add a dummy vertex x_{p+1} in X and let N(x_{p+1}) = {y_q};
for i = 2 to p + 1
    w_i = (i - 2) + (|N(x_i)| - 1);
next i
let w_k = min_{2<=i<=p+1} {w_i};
if k > 2 then S = {x_2, ..., x_{k-1}} ∪ (N(x_k)\{y_q})
else S = N(x_2)\{y_q};
Output G[V(G) \ S];
```

Figure 9. Algorithm MCISC.

Lemma 4.2 *For a chain graph $G = (X, Y, E)$ with $X = \{x_1, x_2, \ldots, x_p\}$ and $Y = \{y_1, y_2, \ldots, y_q\}$, the number of vertices in a maximum connected interval subgraph of G is $p + q + 3 - \min_{2 \leq i \leq p+1}\{i + |N(x_i)|\}$ where x_{p+1} is a dummy vertex and $N(x_{p+1}) = \{y_q\}$.*

Proof. Let $\ddot{G} = (\ddot{X}, \ddot{Y}, \ddot{E})$ be a maximum connected interval subgraph of G satisfying Lemma 4.1. That is, $(x_1, y_q) \in \ddot{E}$. Let $\ddot{X} = \{x_1, x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_r}\}$ where $1 < \pi_1 < \cdots < \pi_r \leq p$. Let $\hat{X} = \ddot{X} \backslash \{x_1\}$ and $\hat{Y} = \ddot{Y} \backslash \{y_q\}$. Because $\ddot{G}$ is an interval graph, there is no edge between $\hat{X}$ and $\hat{Y}$. Therefore, $N(x_{\pi_i}) \cap \hat{Y} = \emptyset$, for $i \in \{1, \ldots, r\}$. Since $N(x_{\pi_1}) \supseteq N(x_{\pi_2}) \supseteq \cdots \supseteq N(x_{\pi_r})$, we only consider $N(x_{\pi_1}) \cap \hat{Y} = \emptyset$. We have the following equations:

$$
\begin{array}{rcl}
\hat{Y} \subseteq (Y \backslash N(x_{\pi_1})) & \Rightarrow & |\hat{Y}| \leq q - |N(x_{\pi_1})| \\
1 < \pi_1 < \cdots < \pi_r \leq p & \Rightarrow & \pi_1 + r - 1 \leq p \\
|V(\ddot{G})| & = & |\hat{X} \cup \hat{Y} \cup \{x_1, y_q\}| \\
& = & |\hat{X}| + |\hat{Y}| + 2 \\
& \leq & r + q - |N(x_{\pi_1})| + 2 \\
& \leq & p - \pi_1 + 1 + q - |N(x_{\pi_1})| + 2 \\
& \leq & p + q + 3 - \pi_1 - |N(x_{\pi_1})|
\end{array}
$$

It is not hard to see that the maximum occurs in $\pi_1 + |N(x_{\pi_1})| = \min_{2 \leq i \leq p+1}\{i + |N(x_i))|\}$. This completes our proof. $\square$

The correctness of Algorithm MCISC is due to Lemma

4.2. It is not hard to see that the time complexity of MCISC is linear. Conclusively, we have the following theorem.

Theorem 4.2 *A maximum connected interval subgraph of a chain graph can be computed in linear time.*

5. Concluding Remarks

In this paper, we have proposed linear-time algorithms for the maximum connected interval subgraph problem on block graphs and chain graphs.

It has been shown that both the maximum interval subgraph problem and the maximum connected interval subgraph problem are NP-hard and have no constant ratio approximation. For the maximum interval subgraph problem, Yannakakis showed that it remains NP-hard on bipartite graphs [17]. Therefore, it is interesting to decide whether the maximum connected interval subgraph problem on bipartite graphs is NP-hard or not. Besides, it is also interesting to study the maximum connected interval subgraph problem on other special graphs such as permutation graphs and chordal graphs (or even the split graphs). On the other hand, it is still unknown that the maximum interval subgraph problem is solvable on block graphs and chain graphs.

REFERENCES

1. A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, M.A., 1974.
2. H.L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209:1–45, 1998.
3. H.L. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21:358–402, 1996.
4. A. Brandstadt, V.B. Le, and J.P. Spinrad. *Craph Classes: A Survey.* SIAM, 1999.
5. P.W. Goldberg, M.C. Golumbic, H. Kaplan, and R. Shamir. Four strikes against physical mapping of dna. *Journal of Computational Biology*, 2:139–152, 1995.
6. M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, New York, 1980.
7. F. Harary. A characterization of block graphs. *Canadian Mathematic Bull*, 6:1–6, 1982.
8. H. Kaplan, R. Shamir, and R.E. Tarjan. Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. *SIAM Journal on Computing*, 28:1906–1922, 1999.
9. R.M. Karp. Mapping the genome: some combinatorial problems arising in molecular biology. *25th Ann. Sympos. on Theory and Computing*, pages 278–285, 1993.
10. L.M. Kirousis and C.H. Papadimitriou. interval graph and searching. *Discrete Mathematics*, 55:181–184, 1985.
11. M.S. Krishnamoorthy and N. Deo. Node-deletion NP-complete problems. *SIAM Journal on Computing*, 8:619–625, 1979.
12. J.M. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20:219–230, 1980.
13. C. Lund and M. Yannakakis. The approximation of maximum subgraph problems. *Proc. 20th International Colloquium on Automata, Languages and Programming*, pages 40–51, 1993.
14. I. Pe'er and R. Shamir. Realizing interval graphs with side and distance constraints. *SIAM Journal on Discrete Mathematics*, 10:662–687, 1997.
15. S.L. Peng, M.T. Ko, C.W. Ho, T. s. Hsu, and C.Y. Tang. Graph searching on some subclasses of chordal graphs. *Algorithmica*, 27:395–426, 2000.
16. M. Yannakakis. The effect of a connectivity requirement on the complexity of maximum subgraph problems. *Journal of the Association for Computing Machinery*, 26:618–630, 1979.
17. M. Yannakakis. Node-deletion problems on bipartite graphs. *SIAM Journal on Computing*, 10:310–327, 1981.