Computing directed pathwidth in $O(1.89^n)$ time

Kenta Kitsunai[1,a)] Yasuaki Kobayashi[1,b)] Keita Komuro[1,c)] Hisao Tamaki[1,d)]
Toshihiro Tano[1,e)]

Abstract: We give an algorithm for computing the directed pathwidth of a digraph with n vertices in $O(1.89^n)$ time. This is the first algorithm with running time better than the straightforward $O^*(2^n)$. As a special case, it computes the pathwidth of an undirected graph in the same amount of time, improving on the algorithm due to Suchan and Villanger which runs in $O(1.9657^n)$ time.

Keywords: Directed pathwidth, Exact exponential algorithm, Graph algorithm, Pathwidth

1. Introduction

The *pathwidth* [2], [16] of an undirected graph G is defined as follows. A *path-decomposition* G is a sequence $\{X_i\}$, $1 \leq i \leq t$, of vertex sets of G that satisfies the following three conditions:

(1) $\bigcup_{1 \leq i \leq t} X_i = V(G)$,
(2) for each edge $\{u, v\}$ of G, there is some i, $1 \leq i \leq t$ such that $u, v \in X_i$, and
(3) for each $v \in V(G)$, the set of indices i such that $v \in X_i$ is contiguous, i.e., is of the form $\{i \mid a \leq i \leq b\}$.

The *width* of a path-decomposition $\{X_i\}$, $1 \leq i \leq t$, is $\max_{1 \leq i \leq t} |X_i| - 1$ and the *pathwidth* of G is the smallest integer k such that there is a path-decomposition of G whose width is k.

The *directed path-decomposition* of a digraph G is defined analogously. A sequence $\{X_i\}$, $1 \leq i \leq t$, of vertex sets is a directed path-decomposition of G if, together with conditions 1 and 3 above, the following condition 2' instead of condition 2 is satisfied:

2'. for each directed edge (u, v) of G, there is a pair i, j of indices such that $i \leq j$, $u \in X_i$, and $v \in X_j$.

The directed pathwidth of G is defined similarly to the pathwidth of an undirected graph. According to Barát [1], the notion of directed pathwidth was introduced by Reed, Thomas, and Seymour around 1995.

For an undirected graph G, let $\hat{G}$ denote the digraph obtained from G by replacing each edge $\{u, v\}$ by a pair of directed edges (u, v) and (v, u). Then, condition 2 for G implies condition 2' for $\hat{G}$ and, conversely, condition 2' for $\hat{G}$ together with condition 3 implies condition 2 for G. Therefore, a directed path-decomposition of $\hat{G}$ is a path-decomposition of G and vice versa. Thus, the problem of computing the pathwidth of an undirected graph is a special case of the problem of computing the directed pathwidth of a digraph. On the other hand, directed pathwidth is of interest since some problems, such as Directed Hamiltonicity, are polynomial time solvable on digraphs with bounded directed pathwidth [10] although not necessarily on those whose underlying graphs have bounded pathwidth. Directed pathwidth is also studied in the context of search games [1], [21].

Computing pathwidth is NP-hard [11] even for bounded degree planar graphs [15], chordal graphs [8], cocomparability graphs [9] and bipartite distance hereditary graphs [13] (although it is polynomial time solvable for permutation graphs [5], cographs [6], and circular-arc graphs [18]). Consequently, computing directed pathwidth is NP-hard even for digraphs whose underlying graphs lie in these classes. On the positive side, pathwidth is fixed parameter tractable [17] with running time linear in n [3]. In contrast, it is open whether directed pathwidth is fixed parameter tractable. Recent work of one of the present authors [20] shows that directed pathwidth admits an XP algorithm, that is, an algorithm with running time $n^{O(k)}$, where k is the directed pathwidth of the given digraph with n vertices.

Without parameterization, both problems can be solved in $O^*(2^n)$ time, where n is the number of vertices and the O^* notation hides polynomial factors, using Bellman-Held-Karp style dynamic programming for vertex ordering problems [4]. Suchan and Villanger [19] improved the running time for pathwidth to $O(1.9657^n)$ and also gave an additive constant approximation of pathwidth in $O(1.89^n)$ time. On the other hand, no algorithm faster than $O^*(2^n)$ time was known for directed pathwidth before the present work.

Our result is as follows.

Theorem 1.1. *The direct pathwidth of a digraph with n vertices can be computed in $O(1.89^n)$ time.*

Our algorithm can be viewed as one based on Bellman-Held-Karp style dynamic programming. For each $U \subseteq V(G)$, let $N^-(U)$ denote the set of in-neighbors of U. Given a positive integer k and a digraph G, we build the collection of "feasible" subsets of $V(G)$, where $U \subseteq V(G)$ is considered feasible if $G[U \cup N^-(U)]$ has a

1 Meiji University, Kawasaki, Japan 214-8571
a) kitsunai@cs.meiji.ac.jp
b) yasu0207@cs.meiji.ac.jp
c) kouki-metal@cs.meiji.ac.jp
d) tamaki@cs.meiji.ac.jp
e) tano-0820@cs.meiji.ac.jp

directed path-decomposition of width $\leq k$ whose last subset X_t contains $N^-(U)$ (for a more precise definition of feasibility see Section 2). To get a non-trivial bound on the number of subsets U, we adopt a strategy essentially due to Suchan and Villanger [19]. Since either U, $N^-(U)$, or $V(G)\setminus(U\cup N^-(U))$ has cardinality at most $n/3$, the hope is that we may be able to obtain a non-trivial upper bound on the number of relevant subsets U using the well-known bound $2^{H(\alpha)n}$ on the number of subsets of $V(G)$ with cardinality at most αn, where $H(x)$ is the binary entropy function, with $\alpha = \frac{1}{3}$. Note that $2^{H(1/3)} < 1.89$. The difficulty, as observed in [19], is that there can be an exponential number of subsets U with $N^-(U) = S$ for a fixed S. The larger constant 1.9657 or the relaxation to additive constant approximation in [19] comes from the need to deal with this problem.

A key to overcoming this difficulty is the following observation whose undirected version is used in [19]. Suppose we are to decide whether the directed pathwidth of G is at most k and S is a vertex set of G with $|S| < k - d$. Let C be the set of strongly connected components of $G[V(G)\setminus S]$ with cardinality greater than d. For each subset $\mathcal{A}$ of C, we are able to define at most one subset U in a "canonical form" such that $N^-(U) = S$, U contains all components in $\mathcal{A}$, but disjoint from all components in $C\setminus\mathcal{A}$. Although there are variants of U that satisfy these conditions, it can be shown that only the canonical one is needed in our dynamic programming computation. Thus, the number of relevant subsets U with $N^-(U) = S$ is bounded by $2^{|C|} \leq 2^{\frac{n}{d+1}}$. See Lemma 2.5 for a more formal treatment.

Our result is established by two algorithms, which we call LARGE-WIDTH and SMALL-WIDTH. Algorithm LARGE-WIDTH deals with the case where $k \geq (\frac{1}{3}+\delta)n$, δ being a small constant, while algorithm SMALL-WIDTH deals with the other case. In algorithm LARGE-WIDTH, the slack of δn allows us to bound the number of subsets U with $N^-(U) = S$ for each S with $|S| \leq n/3$ by $2^{1/\delta}$. In algorithm SMALL-WIDTH, we force a slack of d where d is a large enough constant: we record only those feasible sets U with $|N^-(U)| < k - d$. To process those subsets U with $|N^-(U)| \geq k-d$, we use an algorithm based on the XP algorithm in [20], which runs in $n^{d+(1)}$ time and either decides that the directed pathwidth of G is at most k, decides that U is irrelevant, or produces some proper superset W with $|N^-(W)| < k - d$, which can safely replace U in the search. See Lemma 5.1 for details.

The proofs are omitted in this version and can be found in [14].

The rest of this paper is organized as follows. In Section 2 we define basic concepts and prove some lemmas needed by our algorithms. In Section 3, we state and analyze algorithm LARGE-WIDTH. In Section 4, we review the XP algorithm given in [20] to prepare for the next section. Then, in Section 5, we state and analyze algorithm SMALL-WIDTH. Finally, in Section 6, we combine the two algorithms to prove Theorem 1.1.

2. Preliminaries

Let G be a digraph. We use the standard notation: $V(G)$ is the set of vertices of G, $E(G)$ is the set of edges of G, and $G[U]$, where $U \subseteq V(G)$, is the subgraph of G induced by U. For each vertex $v \in V(G)$, we denote by $N^-_G(v)$ the set of in-neighbor of v, i.e., $N^-_G(v) = \{u \in V(G)\setminus\{v\} | (u, v) \in E(G)\}$. For each subset U of $V(G)$, we denote by $N^-_G(U)$ the set of in-neighbors of U, i.e., $N^-_G(U) = \bigcup_{v\in U} N^-_G(v) \setminus U$. When G is clear from the context, we drop G from this notation. We also use the notation $\tilde{U} = V(G)\setminus(U\cup N^-(U))$ where G is implicit.

We call a sequence σ of vertices of G *non-duplicating* if each vertex of G occurs at most once in σ. We denote by $\Sigma(G)$ the set of all non-duplicating sequences of vertices of G. For each sequence $\sigma \in \Sigma(G)$, we denote by $V(\sigma)$ the set of vertices constituting σ and by $|\sigma| = |V(\sigma)|$ the length of σ.

For each pair of sequences $\sigma, \tau \in \Sigma(G)$ such that $V(\sigma)\cap V(\tau) = \emptyset$, we denote by $\sigma\tau$ the sequence in $\Sigma(G)$ that is σ followed by τ. If $\sigma' = \sigma\tau$ for some τ, then we say that σ is a *prefix* of σ' and that σ' is an *extension* of σ; we say that σ is a *proper* prefix of σ' and that σ' is a *proper* extension of σ if τ is nonempty.

Let G be a digraph and k a positive integer. We say $\sigma \in \Sigma(G)$ is *k-feasible* for G if $|N^-_G(\sigma')| \leq k$ for every prefix σ' of σ. We say that σ is *strongly k-feasible* for G if moreover σ is a prefix of a k-feasible sequence τ with $V(\tau) = V(G)$. We may drop the reference to G and say σ is k-feasible (or strongly k-feasible) when G is clear from the context.

For each $U \subseteq V(G)$, we say that U is k-feasible (strongly k-feasible) if there is a k-feasible (strongly k-feasible) sequence σ with $V(\sigma) = U$.

The directed vertex separation number of digraph G, denoted by $\mathrm{dvsn}(G)$, is the minimum integer k such that $V(G)$ is k-feasible.

It is known that the directed pathwidth of G equals $\mathrm{dvsn}(G)$ for every digraph G [21](see also [12] for the undirected case). Based on this fact, we work on the directed vertex separation number in the remaining of this paper.

The following lemma formulates a straightforward reasoning used twice in the sequel.

Lemma 2.1. *Let $U \subseteq V(G)$, let $X \subseteq V(G)\setminus U$ be such that $N^-(X) \subseteq U \cup N^-(U)$, and let $W = U \cup X$. Suppose that W is k-feasible and U is strongly k-feasible. Then, W is also strongly k-feasible.*

We call $U \subseteq V(G)$ a *full set* (with respect to G), if there is no $v \in N^-(U)$ with $N^-(v) \subseteq U\cup N^-(U)$. For each U, there is a unique superset of U that is a full set, which we denote by $\mathrm{fullset}(U)$. Indeed, $\mathrm{fullset}(U)$ is defined by

$$\mathrm{fullset}(U) = U \cup \{v \in N^-(U) \mid N^-(v) \subseteq U \cup N^-(U)\}.$$

Note that $N^-(\mathrm{fullset}(U)) \subseteq N^-(U)$.

Lemma 2.2. *Let U be an arbitrary subset of $V(G)$. If U is k-feasible then so is* $\mathrm{fullset}(U)$*. Moreover, if U is strongly k-feasible then so is* $\mathrm{fullset}(U)$*.*

Let $U \subseteq V(G)$, $H = G[V(G)\setminus N^-(U)]$. Observe that, for each strongly connected component C of H, either $C \subseteq U$ or $C \subseteq \tilde{U}$, as otherwise $N^-(U)$ would contain a vertex in C.

An undirected counterpart of the following lemma is called the *component push rule* in [19].

Lemma 2.3. *Let U and H be as above and let C be a strongly connected component of H such that $C \subseteq \tilde{U}$, $N^-(C) \subseteq U\cup N^-(U)$, and $|N^-(U)| + |C| \leq k + 1$. If U is k-feasible then $U \cup C$ is*

k-feasible. Moreover, if U is strongly k-feasible then $U \cup C$ is strongly k-feasible.

For each digraph H let $C(H)$ denote the set of all strongly connected components of H. Consider the natural partial ordering $\prec$ on $C(H)$: $C \prec D$ if and only if H contains a directed path from a vertex in C to a vertex in D. For each $U \subseteq V(G)$ with $|N^-(U)| \leq k$, we denote by $\mathrm{push}_k(U)$ the superset of U defined as follows. Let $H = G[V(G) \setminus N^-(U)]$, $s = k - |N^-(U)| + 1$, and define

$$\mathcal{P} = \{C \in C(H) \mid C \subseteq \tilde{U}, |C| \leq s, \text{and there is no } D \in C(H) \text{ with } D \subseteq \tilde{U}, |D| > s, \text{ and } D \prec C\}.$$

Then, we let $\mathrm{push}_k(U) = U \cup \bigcup_{C \in \mathcal{P}} C$.

By a repeated application of Lemma 2.3, we obtain the following lemma

Lemma 2.4. *Let $U \subseteq V(G)$. If U is k-feasible then so is $\mathrm{push}_k(U)$. Moreover, if U is strongly k-feasible then so is $\mathrm{push}_k(U)$.*

Following [19] we use component push rules not only as an algorithmic technique but also as a tool for analysis. The following lemma formalizes this latter aspect.

Lemma 2.5. *Let $S \subseteq V(G)$ with $|S| \leq k$. Then, the number of vertex sets $U \subseteq V(G)$ with $N^-(U) = S$ and $\mathrm{push}_k(U) = U$ is at most $2^{\frac{n}{s+1}}$ where $s = k - |S| + 1$.*

Let $H(x) = -x \log x - (1-x)\log(1-x)$, $0 < x < 1$, denote the binary entropy function. We freely use the following well-known bound on the number of subsets of bounded cardinality.

Proposition 2.1. (see [7], for example) *Let S be a set of n elements and let $0 < \alpha \leq \frac{1}{2}$. Then the number of subsets of S with cardinality at most αn is at most $2^{H(\alpha)n}$.*

3. Algorithm LARGE-WIDTH

Given an integer $k > 0$ and a digraph G with n vertices, Algorithm LARGE-WIDTH decides whether $\mathrm{dvsn}(G) \leq k$ in the following steps.

The algorithm uses function f^* defined as follows. Define $f : 2^{V(G)} \to 2^{V(G)}$ by $f(U) = \mathrm{fullset}(\mathrm{push}_k(U))$. Since $U \subseteq f(U)$, there is some h for each U such that $f^h(U) = f^{h+1}(U)$. We denote this $f^h(U)$ by $f^*(U)$. Note that if $W = f^*(U)$ for some U then $W = \mathrm{fullset}(W) = \mathrm{push}_k(W)$.

(1) Set $\mathcal{U}_1 := \{\{v\} \mid v \in V(G), |N^-(v)| \leq k\}$ and $\mathcal{U}_i := \emptyset$ for $2 \leq i \leq n$.
(2) Repeat the following for $i = 1, 2, \ldots, n-1$.
  (a) For each $U \in \mathcal{U}_i$ and for each $v \in V(G) \setminus U$ with $|N^-(U \cup \{v\})| \leq k$, let $W = f^*(U \cup \{v\})$ and reset $\mathcal{U}_j := \mathcal{U}_j \cup \{W\}$ where $j = |W|$.
(3) If $\mathcal{U}_n$ is not empty then answer "YES"; otherwise answer "NO".

Lemma 3.1. *Algorithm LARGE-WIDTH is correct.*

We analyze the complexity of this algorithm for particular values of k for which this algorithm is intended.

Lemma 3.2. *Let $\delta > 0$ be fixed. For $k > (\frac{1}{3} + \delta)n$, algorithm LARGE-WIDTH runs in $O^*(2^{H(\frac{1}{3})n})$ time.*

4. XP algorithm

We review the XP algorithm for directed pathwidth due to Tamaki [20] which is an essential ingredient in algorithm SMALL-WIDTH.

Theorem 4.1. [20] *Given a positive integer k and a digraph G with n vertices and m edges, it can be decided in $O(mn^{k+1})$ time whether $V(G)$ is k-feasible.*

The algorithm claimed in this theorem is based on the natural search tree consisting of all k-feasible sequences in $\Sigma(G)$. The running time is achieved by pruning this search tree of potentially factorial size into one with $O(n^{k+1})$ search nodes. The following lemma is used to enable this pruning. We say that a proper extension τ of $\sigma \in \Sigma(G)$ is *non-expanding* if $|N^-(\tau)| \leq |N^-(\sigma)|$.

Lemma 4.1. (Commitment Lemma [20]) *Let σ be a strongly k-feasible sequence in $\Sigma(G)$ and let τ be a shortest non-expanding k-feasible extension of σ, that is,*
(1) $|N^-(V(\tau))| \leq |N^-(V(\sigma))|$, and
(2) $|N^-(V(\tau'))| > |N^-(V(\sigma))|$ for every k-feasible proper extension τ' of σ with $|\tau'| < |\tau|$.
Then, τ is strongly k-feasible.

Suppose sequence σ is in the search tree and has a non-expanding k-feasible extension τ. Then the commitment lemma allows σ to "commit to" this descendant τ: we may remove from the search tree all the descendants of σ with length $|\tau|$ but τ. It is shown in [20] that the resulting search tree contains $O(n^{k+1})$ sequences.

To adapt this result for our purposes, we need some details of the pruned search tree. Let σ and τ be two k-feasible sequences of the same length. We say that σ is *preferable to* τ if either $|N^-(V(\sigma))| < |N^-(V(\tau))|$ or $|N^-(V(\sigma))| = |N^-(V(\tau))|$ and $\sigma < \tau$ in the lexicographic ordering on $\Sigma(G)$ based on some fixed total order on $V(G)$. We say σ *suppresses* τ, if σ is preferable to τ and there is some common prefix σ' of σ and τ such that σ is a shortest non-expanding k-feasible extension of σ'.

Let S_i, $1 \leq i \leq n$, denote a set of k-feasible sequence with length i defined inductively as follows. Each member of S_i will represent a node in our search tree at level i.
(1) $S_1 = \{v \mid |N^-(v)| \leq k\}$.
(2) For $1 \leq i < n$, let $T_{i+1} = \{\sigma v \mid \sigma \in S_i, v \in V(G) \setminus V(\sigma), \text{and } |N^-(V(\sigma) \cup \{v\})| \leq k\}$. We let S_{i+1} be the set of elements of T_{i+1} not suppressed by any elements of T_{i+1}.

To analyze the size of each set S_i, [20] assigns a sequence $\mathrm{sgn}(\sigma)$, called the *signature* of σ, to each k-feasible sequence σ as follows.

Call a non-expanding k-feasible extension τ of σ *locally shortest*, if no proper prefix of τ is a non-expanding extension of σ. We define $\mathrm{sgn}(\sigma)$ inductively as follows.
(1) If σ is empty then $\mathrm{sgn}(\sigma)$ is empty.
(2) If σ is nonempty and is a locally shortest non-expanding extension of some prefix of σ, then $\mathrm{sgn}(\sigma) = \mathrm{sgn}(\tau)$, where τ is the shortest prefix of σ with the property that σ is a locally shortest non-expanding k-feasible extension of τ.
(3) Otherwise $\mathrm{sgn}(\sigma) = \mathrm{sgn}(\sigma')v$, where v is the last vertex of σ

and $\sigma = \sigma' v$.

Lemma 4.2. [20] *For each i, $1 \le i \le n$, if σ and τ are two distinct elements of S_i then neither* $\mathrm{sgn}(\sigma)$ *nor* $\mathrm{sgn}(\tau)$ *is the prefix of the other.*

The following properties of the pruned search tree follow from this lemma.

Lemma 4.3. *Suppose $\sigma \in S_{|\sigma|}$ is a non-expanding extension of a singleton sequence v. Then, σ is the only extension of v in $S_{|\sigma|}$.*

Lemma 4.4. *Let $1 \le j \le n$ and let h be the minimum value of $|N^-(V(\sigma))|$ over all sequences σ in $\bigcup_{1\le i\le j} S_i$. Then, we have $|S_i| \le n^{k-h}$ for $1 \le i \le j$.*

5. Algorithm SMALL-WIDTH

Fix $\epsilon > 0$ and fix an integer $d > 1/\epsilon$. The following description of our algorithm depends on d. We assume k, an input to the algorithm, satisfies $k > d$; otherwise the algorithm in Theorem 4.1 runs in $n^{O(1)}$ time.

Our strategy is to record only those sets U with $|N^-(U)| < k-d$ and $U = \mathrm{push}_k(U)$ in our computation. For each S with $|S| < k-d$, by Lemma 2.5, the number of U such that $N^-(U) = S$ and $U = \mathrm{push}_k(U)$ is at most $2^{\epsilon n}$.

To process U with $|N^-(U)| \ge k-d$, we use the XP algorithm in [20]. The following lemma is at the heart of our algorithm.

Lemma 5.1. *There is an algorithm that, given a k-feasible vertex set $U \subseteq V(G)$ with $k-d \le |N^-(U)| \le k$, runs in $n^{d+O(1)}$ time and either*
(1) proves that U is strongly k-feasible,
(2) proves that U is not strongly k-feasible, or
(3) produces some proper superset W of U with $|N^-(W)| < k-d$ such that U is strongly k-feasible if and only if W is strongly k-feasible.

Algorithm SMALL-WIDTH, given G and k, decides if $\mathrm{dvsn}(G) \le k$ in the following steps.
(1) Set $\mathcal{U}_1 := \{\{v\} \mid v \in V(G), |N^-(v)| \le k\}$ and $\mathcal{U}_i := \emptyset$ for $2 \le i \le n$.
(2) Repeat the following for $i = 1, 2, \ldots n-1$.
For each $U \in \mathcal{U}_i$ and for each $v \in V(G) \setminus U$ with $|N^-(U \cup \{v\})| \le k$, let $U' = U \cup \{v\}$ and do the following.
  (a) If $|N^-(U')| < k-d$ then let $W = \mathrm{push}_k(U')$ and reset $\mathcal{U}_j := \mathcal{U}_j \cup \{W\}$, where $j = |W|$.
  (b) If $|N^-(U')| \ge k-d$ then apply the algorithm of Lemma 5.1 to G and U'.
    (i) If U' is found strongly k-feasible, then stop the entire algorithm answering "YES".
    (ii) If U' is found not strongly k-feasible, then do nothing.
    (iii) If a proper superset W of U' with $|N^-(W)| < k-d$ is returned, then reset $\mathcal{U}_j := \mathcal{U}_j \cup \{\mathrm{push}_k(W)\}$, where $j = |\mathrm{push}_k(W)|$.
(3) If $\mathcal{U}_n$ is nonempty then answer "YES"; otherwise answer "NO".

Lemma 5.2. *Algorithm SMALL-WIDTH is correct.*

Lemma 5.3. *For $k \le n/2$, algorithm SMALL-WIDTH runs in $O(2^{(H(k/n)+\epsilon)n})$ time.*

6. Combining the two algorithms

Theorem 1.1 immediately follows from a combination of the two algorithms given in previous sections. Observe $2^{H(1/3)} < 1.89$ and choose positive δ and ϵ so that $2^{H(1/3+\delta)+\epsilon} < 1.89$. If $k > (\frac{1}{3}+\delta)n$, we apply algorithm LARGE-WIDTH; otherwise, we apply algorithm SMALL-WIDTH. From Lemmas 3.2 and 5.3, we see that the running time of the algorithm is $O(1.89^n)$ in both cases.

References

[1] J. Barát: Directed path-width and monotonicity in digraph searching, Graphs and Combinatorics 22(2):161–172, 2006.
[2] H. L. Bodlaender: A Tourist Guide Through Treewidth, *Acta Cybernetica*, Vol. 11, pp. 1–23, 1993.
[3] H. L. Bodlaender: A linear-time algorithm for finding tree-decompositions of small treewidth, *SIAM Journal on Computing*, Vol. 25, pp. 1305–1317, 1996.
[4] H. L. Bodlaender, F. V. Fomin, D. Kratsch and D. Thilikos: A Note on Exact Algorithms for Vertex Ordering Problems on Graphs, *Theory of Computing Systems*, Vol. 50(3), pp. 420–432, 2012.
[5] H. L. Bodlaender, T. Kloks and D. Kratsch: Treewidth and pathwidth of permutation graphs, In *@Proceedings of the 20nd International Colloquium on Automata, Languages and Programming, ICALP1993*, pp. 114–125, 1993.
[6] H. L. Bodlaender and R. H. Möhring: The Pathwidth and Treewidth of Cographs, *SIAM Journal on Discrete Mathematics*, Vol. 6(2), pp. 181–188, 1992.
[7] F. V. Fomin and D. Kratsch: Exact Exponential Algorithms, *Springet-Verlag*, 2010.
[8] J. Gusted: On the pathwidth of chordal graphs, *Discrete Applied Mathematics*, Vol. 45(3), pp. 233–248, 1993.
[9] M. Habib and R. H. Möhring: Treewidth of cocomparability graphs and a new order-theoretic parameter, *Order*, Vol. 11(1), pp. 44-60, 1994.
[10] T. Johnson, N. Robertson, P. D. Seymour and R. Thomas: Directed tree-width, *Journal of Combinatorial Theory, Series B*, Vol. 82(1), pp. 138–154, 2001.
[11] T. Kashiwabara and T. Fujisawa: NP-completeness of the problem of finding a minimum-clique-number interval graph containing a given graph as a subgraphm, In *Proceedings of International Symposium on Circuits and Systems*, pp. 657–660, 1979.
[12] G. N. Kinnersley: The vertex separation number of a graph equals its path-width, *Information Processing Letters*, 42(6), pp. 345–350, 1992.
[13] T. Kloks, H. L. Bodlaender, H. Müller and D. Kratsch: Computing treewidth and minimum fill-in: All you need are the minimal separators, In *Proceedings of the 1st Annual European Symposium on Algorithms*, pp. 260–271, 1993.
[14] K. Kitsunai, Y. Kobayashi, K. Komuro, H. Tamaki and T. Tano: Computing directed pathwidth in $O(1.89^n)$ time, In *Proceedings of the 7th International Symposium on Parameterized and Exact Computation*, to appear, 2012.
[15] B. Monien and I. H. Sudborough: Min cur is NP-complete for edge weighted trees, *Theoretical Computer Science*, Vol. 58(1–3), pp. 209–229,1988.
[16] N. Robertson and P. D. Seymour: Graph minors. I. Excluding a forest, *Journal of Combinatorial Theory, Series B*, Vol. 35(1), pp. 39–61, 1983.
[17] N. Robertson and P. D. Seymour: Graph minors VIII The disjoint paths peoblem, *Journal of Combinatorial Theory, Series B*, Vol. 63, pp. 65–110, 1995.
[18] K. Suchan and I. Todinca: Pathwidth of circular-arc graphs, In *Proceedings of 33rd International Workshop on Graph-Theoretic Concepts in Computer Science*, pp. 258–269, 2007.
[19] K. Suchan and Y. Villanger: Computing Pathwidth Faster Than 2^n, In *Proceedings of the 4th International Workshop on Parameterized and Exact Computation, IWPEC2009*, pp. 324–335, 2009.
[20] H. Tamaki: A Polynomial Time Algorithm for Bounded Directed Pathwidth, In *Proceedings of the 37th International Workshop on Graph-Theoretic Concepts in Computer Science, WG2011*, pp. 331–342, 2011.

[21] B. Yang and Y. Cao: Digraph searching, directed vertex separation and directed pathwidth, *Discrete Applied Mathematics*, Vol. 156(10), pp. 1822–1837, 2008.