Computing the pathwidth of directed graphs with small vertex cover

Yasuaki Kobayashi[1,a)]

Abstract: We give an algorithm that computes the pathwidth of a given directed graph D in $3^{\tau(D)}n^{O(1)}$ time where n is the number of vertices of D and $\tau(D)$ is the number of vertices of a minimum vertex cover of the underlying graph of D. This result extends that of [Chapelle *et al.*, 2013] for undirected graphs to directed graphs. Moreover, our algorithm is based on a standard dynamic programming with a simple tree-pruning trick, which is extremely simple and easy to implement.

1. Introduction

Chapelle *et al*. [7] proved the following theorem.

Theorem 1 ([7])**.** *The pathwidth of an undirected graph G can be computed in $3^{\tau(G)}n^{O(1)}$ time where n is the number of vertices of G and $\tau(G)$ is the size of the minimum vertex cover of G.*

The notion of pathwidth is also defined on directed graphs. In this paper, we extend Theorem 1 to directed graphs.

Theorem 2. *The pathwidth of a directed graph D can be computed in $3^{\tau(D)}n^{O(1)}$ time where n is the number of vertices of D and $\tau(D)$ is the size of the minimum vertex cover of the underlying graph of D.*

It is known that the pathwidth of an undirected graph G equals the pathwidth of a directed graph G' where G' is obtained from G by replacing each edge $\{u, v\}$ of G by a pair of anti-parallel arcs (u, v) and (v, u) (see [1], for example).

Computing pathwidth is NP-hard [10] for undirected graphs and for directed graphs, and is also considered in parameterized complexity. A problem is said to be *fixed parameter tractable* (*FPT*) if there is an algorithm for the problem with running time $f(k)n^{O(1)}$ where n is the size of instance and k is a parameter (see [8], for more information). The decision version of the pathwidth problem, deciding whether the pathwidth of a given undirected graph is at most a parameter k, is FPT [3], [6]. However, the fixed parameter tractability of the directed case remains unclear. It is only recently that XP algorithms for the decision problem on directed graphs are found by [13], [15].

The algorithm of [6] is, in fact, an FPT algorithm for pathwidth parameterized by treewidth. The running time is, however, somewhat large even when the treewidth of an input graph is small. It is natural to ask whether there is a faster FPT algorithm for pathwidth using more restricted parameters such as vertex cover number. Chapelle *et al*. [7] positively answered this question by proving Theorem 1.

Our result is along this line. We give an FPT algorithm that computes the pathwidth of a given directed graph parameterized by vertex cover number. Our algorithm is a standard dynamic programming for vertex ordering problems [5] with a simple tree-pruning trick (some variants of this trick can be found in the recent work for exact algorithms for pathwidth [12], [14], [15]). Although our analysis is basically based on the same idea with [7], our algorithm is much simpler than theirs. In particular, we use a vertex cover in the analysis but not in the algorithm, while the algorithm of [7] uses a vertex cover explicitly. Moreover, our algorithm works on undirected and directed graphs. It is not clear if the algorithm in [7] can be adapted to directed graphs. On the other hand, [7] not only proved Theorem 1 but also gave an FPT algorithm for treewidth parameterized by vertex cover number.

2. Preliminaries

Let $D = (V, A)$ be a directed graph with $n = |V|$. For a vertex $x \in V$, we denote by $N^-(x)$ the set of in-neighbors of x and, for $X \subseteq V$, by $N^-(X)$ the set of in-neighbors of X, i.e. $N^-(X) = \bigcup_{x \in X} N^-(x) \setminus X$.

Let σ be a sequence of vertices in V. We assume all the sequences of vertices in this paper have no repetition, that is, the vertices in σ are distinct from each other. The length of σ is denoted by $|\sigma|$. We will write the sequence as a list of vertices $\sigma = (v_1, v_2, \ldots, v_{|\sigma|})$. For $0 \leq i \leq |\sigma|$, the *prefix* of σ of length i, denoted by σ_i, is the sequence consisting of the first i vertices in σ appearing in the same order as in σ. The set of vertices in σ is denoted by $V(\sigma)$. A *permutation* of D is a sequence consisting of all the vertices in V. For an integer k, we say σ is *k-feasible* for D if $|N^-(V(\sigma'))| \leq k$ for all prefix σ' of σ and is *strongly k-feasible* for D if there is a k-feasible permutation τ of D such that σ is a prefix of τ. We extend these notations to vertex sets: a subset U of V is (strongly) k-feasible if there is a (strongly) k-feasible sequence σ with $V(\sigma) = U$. We may drop the reference to D when the reference is clear. The *vertex separation number* of D is the minimum integer k such that V is k-feasible.

[1] Gakushuin University, Toshima-ku, Tokyo, Japan 171-8588
[a)] yasuaki.kobayashi@gakushuin.ac.jp

For every directed graph D, the vertex separation number of D equals its pathwidth [16] (the undirected version of this fact can be found in [11]). Because of this fact, we work on the vertex separation number.

Our algorithm is based on the following standard dynamic programming algorithm. Fix an integer k. It is straightforward to see that a proper subset $U \subset V$ is strongly k-feasible if and only if there exists a vertex $u \in V \setminus U$ such that $U \cup \{u\}$ is also strongly k-feasible. Given this relation, the vertex separation number of D can be computed in $2^n n^{O(1)}$ time.

In order to reduce the running time, we run this dynamic programming only on some subsets of V. To this end, we need the following notions. An *expansion* U^* of $U \subseteq V$ is recursively defined as:

(1) $U^* = U$ if there is no vertex $u \in V \setminus U$ with $N^-(u) \subseteq U \cup N^-(U)$;

(2) $U^* = (U \cup \{u \in V \setminus U : N^-(u) \subseteq U \cup N^-(U)\})^*$ otherwise.

We say U is *relevant* when $U^* = U$. It is easy to see that, for each $U \subseteq V$, there is a unique expansion U^* of U. The key to our dynamic programming is the following lemma.

Lemma 1. *Let U be a k-feasible subset of V. Then the expansion U^* of U is also k-feasible. Moreover, if U is strongly k-feasible, so is U^*.*

Proof. By the definition of expansion, the first statement is clear as $N^-(X) \subseteq N^-(U)$ for every X with $U \subseteq X \subseteq U^*$. Thus, in the following, we prove the second statement.

When $U^* = U$, the lemma is obvious. Next, assume there is a vertex $v \in V \setminus U$ such that $N^-(v) \subseteq U \cup N^-(U)$. Since U is strongly k-feasible, there is a k-feasible permutation $\pi = (v_1, v_2, \ldots, v_n)$ of D such that $U = V(\pi_{|U|})$. Let j be such that $v_j = v$. Since $v \in V \setminus U$, we have $j > |U|$. Let

$$\tau = (v_1, v_2, \ldots, v_{|U|}, v_j, v_{|U|+1}, \ldots, v_{j-1}, v_{j+1}, \ldots, v_n).$$

In words, τ is obtained from π by moving v to the position immediately after $v_{|U|}$. Since $\pi_i = \tau_i$ for $1 \leq i \leq |U|$ and $N^-(V(\pi_i) \cup \{v\}) \subseteq N^-(V(\pi_i))$ for $|U| < i \leq n$, τ is k-feasible and hence $U \cup \{v\}$ is strongly k-feasible. A straightforward induction proves the lemma. □

This lemma is a special case of Lemma 5 in [15]. Similar special cases appeared in [12], [14].

3. Proof of Theorem 2

Given a directed graph $D = (V, A)$ with n vertices and an integer k, our algorithm decides whether V is k-feasible or not in $3^{\tau(D)} n^{O(1)}$ time where $\tau(D)$ is the size of a minimum vertex cover of the underlying graph of D. The algorithm uses a straightforward dynamic programming over relevant sets, which is described as follows. Let U be a strongly k-feasible relevant subset of V. Then there exists $v \in V \setminus U$ such that $U \cup \{v\}$ is strongly k-feasible. By Lemma 1, the expansion of $U \cup \{v\}$ is also strongly k-feasible and is relevant. This discussion immediately gives us a dynamic programming over k-feasible relevant subsets of V. The correctness of this dynamic programming is straightforward.

In what follows, fix a minimum vertex cover C of the underlying graph of D. The next lemma is crucial for our running time analysis.

Lemma 2. *There is an injective mapping from the relevant subsets of V to the collection of ordered tripartitions of C.*

Proof. Let (L, M, R) be an arbitrary ordered tripartition of C. We show that there is at most one relevant set U such that $L = U \cap C$ and $M = N^-(U) \cap C$. Suppose U is such a set. Let $v \in V \setminus C$. Since C is a vertex cover of the underlying graph of G, we have $N^-(v) \subseteq C$. Since U is relevant, we have $v \in U$ if and only if $N^-(v) \subseteq L \cap M$. Thus, U satisfying above conditions is unique, when one exists. □

It is easy to see that the running time of our dynamic programming is in $|\mathcal{R}| \cdot n^{O(1)}$ where $\mathcal{R}$ is the set of all relevant subsets of V. By Lemma 2, $|\mathcal{R}|$ is bounded by $3! \cdot 3^{|C|}$ and hence Theorem 2 follows.

Acknowledgments

The author thanks Hisao Tamaki for his suggestions for improving the presentation of the paper.

References

[1] Barát, J.: Directed path-width and monotonicity in digraph searching, *Graphs and Combinatorics* 22(2), 161–172 (2006).

[2] Bodlaender, H. L.: A tourist guide through treewidth, *Acta Cybernetica* 11, 1–23 (1993).

[3] Bodlaender, H. L.: A linear-time algorithm for finding tree-decompositions of small treewidth *SIAM Journal on Computing* 25(6), 1305–1317 (1996).

[4] Bodlaender, H. L., Fomin, F. V., Koster, A. M. C. A, Kratsch, D., Thilikos, D. M.: On exact algorithms for treewidth, *ACM Transactions on Algorithms* 9(1), 1–12 (2012).

[5] Bodlaender, H. L., Fomin, F. V., Koster, A. M. C. A, Kratsch, D., Thilikos, D. M.: A note on exact algorithms for vertex ordering problems on graphs, *Theory of Computing Systems* 50(2), 420–432 (2012).

[6] Bodlaender, H. L, Kloks, T.: Efficient and constructive algorithms for the pathwidth and treewidth of graphs, *Journal of Algorithms* 21(2), 358–402 (1996).

[7] Chapelle, M., Liedloff, M., Todinca, I., Villanger, Y.: Treewidth and pathwidth parameterized by the vertex cover number, In *Proceedings of WADS 2013*, LNCS, vol. 8037, pp. 232–243 (2013).

[8] Downey, R. G., Fellows, M. R.: *Parameterized Complexity*, Springer (1998).

[9] Fellows, M. R., Lokshtanov, D., Misra, N., Rosamond, F. A., Saurabh, S.: Graph layout problems parameterized by vertex cover, In *Proceedings of ISAAC 2008*, LNCS, vol. 5369, pp. 294–305 (2008). Addison-Wesley, 2nd edition (1973).

[10] Kashiwabara, T., Fujisawa, T.: NP-completeness of the problem of finding a minimum-clique-number interval graph containing a given graph as a subgraph, In *Proceedings of ISCAS 1979*, pp. 657–660 (1979).

[11] Kinnersley, N. G.: The vertex separation number of a graph equals its path-width, *Information Processing Letters* 42(6), 345–350 (1992).

[12] Kitsunai, K., Kobayashi, Y., Komuro, K., Tamaki, H., Tano, T.: Computing directed pathwidth in $O(1.89^n)$ time, In *Proceedings of IPEC 2012*, LNCS, vol. 7535, pp. 182–193 (2012).

[13] Nagamochi, H.: Linear layouts in submodular systems In *Proceedings of ISAAC 2012*, LNCS, vol. 7676, pp. 475–484 (2012).

[14] Suchan, K., Villanger, Y.: Computing pathwidth faster than 2^n, In *Proceedings of IWPEC 2009*, LNCS, vol. 5917, pp. 324–335 (2009).

[15] Tamaki, H.: A polynomial time algorithm for bounded directed pathwidth, In *Proceedings of WG 2011*, LNCS, vol. 6986, pp. 331–342 (2011).

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