Theory Comput Syst
DOI 10.1007/s00224-011-9312-0

A Note on Exact Algorithms for Vertex Ordering Problems on Graphs

Hans L. Bodlaender · Fedor V. Fomin · Arie M.C.A. Koster · Dieter Kratsch · Dimitrios M. Thilikos

Abstract In this note, we give a proof that several vertex ordering problems can be solved in $O^*(2^n)$ time and $O^*(2^n)$ space, or in $O^*(4^n)$ time and polynomial space. The algorithms generalize algorithms for the TRAVELLING SALESMAN PROBLEM by Held and Karp (J. Soc. Ind. Appl. Math. 10:196–210, 1962) and Gurevich and

This research was partially supported by the project *Treewidth and Combinatorial Optimization* with a grant from the Netherlands Organization for Scientific Research NWO and by the Research Council of Norway and by the DFG research group "Algorithms, Structure, Randomness" (Grant number GR 883/9-3, GR 883/9-4). The research of the last author was supported by the Spanish CICYT project TIN-2004-07925 (GRAMMARS). Parts of this paper appeared earlier in the conclusions section of [2].
D.M. Thilikos was supported by the project "Kapodistrias" (ΑΠ 02839/28.07.2008) of the National and Kapodistrian University of Athens (project code: 70/4/8757).

H.L. Bodlaender (✉)
Department of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
e-mail: hansb@cs.uu.nl

F.V. Fomin
Department of Informatics, University of Bergen, 5020 Bergen, Norway
e-mail: fomin@ii.uib.no

A.M.C.A. Koster
Lehrstuhl II für Mathematik, RWTH Aachen University, Wüllnerstr. 5b, 52062 Aachen, Germany
e-mail: koster@math2.rwth-aachen.de

D. Kratsch
LITA, Université de Metz, 507045 Metz Cedex 01, France
e-mail: kratsch@sciences.univ-metz.fr

D.M. Thilikos
Department of Mathematics, National and Kapodistrian University of Athens, Panepistimioupolis, 15784 Athens, Greece
e-mail: sedthilk@math.uoa.gr

Published online: 21 January 2011

Shelah (SIAM J. Comput. 16:486–502, 1987). We survey a number of vertex ordering problems to which the results apply.

Keywords Graphs · Algorithms · Exponential time algorithms · Exact algorithms · Vertex ordering problems

1 Introduction

In this note, we look at exact algorithms with "moderately exponential time" for graph problems. We show that with relatively simple adaptations of the existing algorithms for the TRAVELLING SALESMAN PROBLEM, a large collection of vertex ordering problems can be solved in $O^*(2^n)$ time and $O^*(2^n)$ space or in $O^*(4^n)$ time and polynomial space. (Here, the O^*-notation suppresses factors that are polynomial in n.) The algorithms that use $O^*(2^n)$ time and $O^*(2^n)$ space employ dynamic programming and have the same structure as the classical algorithm for TSP by Held and Karp [13]. The algorithms with $O^*(4^n)$ time and polynomial space are of a recursive nature and follow a technique first used for TSP by Gurevich and Shelah [12].

This paper is organized as follows. In Sect. 2, we give some preliminary definitions and discuss the form of problems we can handle. A general theorem that gives for all problems of this specific form an algorithm of the Held-Karp type is given and proved in Sect. 3. A similar theorem with proof for Gurevich-Shelah type algorithms (i.e., with polynomial space) is given in Sect. 4. Then, in Sect. 5, we discuss a number of well known vertex ordering problems on graphs to which these theorems can be applied. A few final remarks are made in Sect. 6.

2 Preliminaries

Definitions We assume the reader to be familiar with standard notions from graph theory. Throughout this paper, $n = |V|$ denotes the number of vertices of graph $G = (V, E)$. For a graph $G = (V, E)$ and a set of vertices $W \subseteq V$, the subgraph of G induced by W is the graph $G[W] = (W, \{\{v, w\} \in E \mid v, w \in W\})$.

A *vertex ordering* of a graph $G = (V, E)$ is a bijection $\pi : V \to \{1, 2, \ldots, |V|\}$. For a vertex ordering π and $v \in V$, we denote by $\pi_{<,v}$ the set of vertices that appear before v in the ordering: $\pi_{<,v} = \{w \in V \mid \pi(w) < \pi(v)\}$. Likewise, we define $\pi_{\leq,v}$, $\pi_{>,v}$, and $\pi_{\geq,v}$.

Let $\Pi(S)$ be the set of all permutations of a set S. So, $\Pi(V)$ is the set of all vertex orderings of G, and let for disjoint sets S and R, $\Pi(S, R)$ be the set of all permutations of $S \cup R$ which start with a permutation of S and end with a permutation of R.

A graph $G = (V, E)$ is *chordal*, if every cycle in G of length at least four has a chord, i.e., there is an edge connecting two non-consecutive vertices in the cycle. A *triangulation* of a graph $G = (V, E)$ is a graph $H = (V, F)$ that contains G as a subgraph ($F \subseteq E$) and is chordal.

Form of Problems In this paper, we consider problems of the following form. We have a polynomial time computable function f, mapping each 3-tuple, consisting of a graph $G = (V, E)$, a vertex set $S \subseteq V$, and a vertex $v \in V$ to an integer. Then, we consider the problems to compute

$$\min_{\pi \in \Pi(V)} \max_{v \in V} f(G, \pi_{<,v}, v) \tag{1}$$

or

$$\min_{\pi \in \Pi(V)} \sum_{v \in V} f(G, \pi_{<,v}, v). \tag{2}$$

Note that values $f(G, S, v)$ do not depend on the ordering of S. Several examples of problems that can be formulated in the form of (1) or (2) are given in Sect. 5. For a better intuition of what follows, we give one such example here. The CUTWIDTH problem asks for a given graph $G = (V, E)$ to find a vertex ordering π with

$$\max_{v \in V} |\{\{w, x\} \in E \wedge \pi(w) \leq \pi(v) < \pi(x)\}|$$

as small as possible. In other words, we look for a vertex ordering that minimizes the maximum over all vertices v of the number of edges with one endpoint before v or v itself, and one endpoint after v in the ordering. Thus CUTWIDTH is equivalent to (1), when setting

$$f(G, S, v) = |\{\{w, x\} \in E \mid w \in S \cup \{v\} \wedge x \in V - S - \{v\}\}|.$$

3 Exact Algorithms with Exponential Space

In this section, we show that a large collection of vertex ordering problems on graphs can be solved in $O^*(2^n)$ time and $O^*(2^n)$ space. The technique exploited here is dynamic programming in the style of the Held-Karp algorithms for the TRAVELLING SALESMAN PROBLEM [13].

Theorem 1 *Let f be a polynomial time computable function, mapping each 3-tuple, consisting of a graph $G = (V, E)$, a vertex set $S \subseteq V$, and a vertex $v \in V$ to an integer. Then we can compute in $O^*(2^n)$ time and $O^*(2^n)$ space the following values for a given graph $G = (V, E)$:*

$$\min_{\pi \in \Pi(V)} \max_{v \in V} f(G, \pi_{<,v}, v)$$

or

$$\min_{\pi \in \Pi(V)} \sum_{v \in V} f(G, \pi_{<,v}, v).$$

The proof of the theorem follows the arguments of Held and Karp in [13] and an algorithm of this type for TREEWIDTH from [3].

Let f be as in the statement of Theorem 1. We first give the algorithm that uses $O^*(2^n)$ time and space to compute $\min_{\pi\in\Pi(V)} \max_{v\in V} f(G, \pi_{<,v}, v)$. Define

$$A_G(S) = \min_{\pi\in\Pi(S)} \max_{v\in S} f(G, \pi_{<,v}, v).$$

We set $A_G(\emptyset) = -\infty$. Note that $A_G(V)$ is the value to compute.

Lemma 2 *Let $G = (V, E)$ be a graph, and $S \subseteq V$. If $S \neq \emptyset$, then*

$$A_G(S) = \min_{w\in S} \max\{f(G, S - \{w\}, w), A_G(S - \{w\})\}$$

Proof Suppose $A_G(S) = \max_{v\in S} f(G, \pi_{<,v}, v)$ for $\pi \in \Pi(S)$, then let w be the vertex on the last position of π. Now $\pi_{<,w} = S - \{w\}$. Thus, we have

$$\begin{aligned} A_G(S) &= \max_{v\in S} f(G, \pi_{<,v}, v) \\ &= \max\{f(G, \pi_{<,w}, w), \max_{v\in S-\{w\}} f(G, \pi_{<,v}, v)\} \\ &\geq \max\{f(G, S - \{w\}, w), A_G(S - \{w\})\} \end{aligned}$$

This shows that

$$A_G(S) \geq \min_{w\in S} \max\{f(G, S - \{w\}, w), A_G(S - \{w\})\}$$

Suppose $\max\{f(G, S - \{w\}, w), A_G(S - \{w\})\}$ is minimal for $w \in S$, and

$$A_G(S - \{w\}) = \max_{v\in S-\{w\}} f(G, \pi'_{<,v}, v)$$

for a permutation $\pi' \in \Pi(S - \{w\})$. Let π be the permutation in $\Pi(S)$, that starts with π' and ends with w. Now,

$$\begin{aligned} A_G(S) &\leq \max_{v\in S} f(G, \pi_{<,v}, v) \\ &= \max\{f(G, \pi_{<,w}, w), \max_{v\in S-\{w\}} f(G, \pi_{<,v}, v)\} \\ &= \max\{f(G, S - \{w\}, w), \max_{v\in S-\{w\}} f(G, \pi'_{<,v}, v)\} \\ &= \max\{f(G, S - \{w\}, w), A_G(S - \{w\})\} \end{aligned}$$

This shows that

$$A_G(S) \leq \min_{w\in S} \max\{f(G, S - \{w\}, w), A_G(S - \{w\})\}$$

and thus completes the proof of this lemma. □

Lemma 2 directly gives us a method to compute $A_G(V)$ by dynamic programming: we compute all values $A_G(S)$ in order of increasing number of elements in S,

Algorithm 1 Dynamic-Programming-Algorithm(Graph $G = (V, E)$)

```
Set A(∅) = −∞.
for i = 1 to n do
  for all vertex sets S ⊂ V with |S| = i do
    Set A(S) = min_{w∈S} max{f(G, S − {w}, w), A(S − {w})}
  end for
end for
return A(V)
```

using the formulas given by Lemma 2. We then output $A_G(V)$. Each single value can be computed in polynomial time; we need to store and compute 2^n values, thus the running time and the space are $O^*(2^n)$. See Algorithm 1.

The computation of $\min_{\pi\in\Pi(V)} \sum_{v\in V} f(G, \pi_{<,v}, v)$ is similar. We define

$$B_G(S) = \min_{\pi\in\Pi(S)} \sum_{v\in S} f(G, \pi_{<,v}, v)$$

Now, $B_G(\emptyset) = 0$, and, similar to Lemma 2, we have

$$B_G(S) = \min_{w\in S} f(G, S - \{w\}, w) + B_G(S - \{w\})$$

The remaining details are similar to the maximization case and left to the reader.

In a practical implementation, several improvements to the scheme of Algorithm 1 can be made; an algorithmic engineering study for TREEWIDTH has been carried out, see [3].

4 Exact Algorithms with Polynomial Space

In this section, we give a variant of Theorem 1. This variant applies to the same collection of problems. In contrast with Theorem 1, Theorem 3 uses polynomial space but more (i.e. $O^*(4^n)$) time. It employs recursion instead of dynamic programming, and has the same structure as the algorithm for TSP by Gurevich and Shelah [12]. An algorithm of this type for TREEWIDTH appears in [3].

Theorem 3 *Let f be a polynomial time computable function, mapping each 3-tuple, consisting of a graph $G = (V, E)$, a vertex set $S \subseteq V$, and a vertex $v \in V$ to an integer. Then we can compute in $O^*(4^n)$ time and polynomial space the following values for a given graph $G = (V, E)$:*

$$\min_{\pi\in\Pi(V)} \max_{v\in V} f(G, \pi_{<,v}, v)$$

or

$$\min_{\pi\in\Pi(V)} \sum_{v\in V} f(G, \pi_{<,v}, v)$$

Again, we concentrate on the computation of $\min_{\pi\in\Pi(V)}\max_{v\in V} f(G,\pi_{<,v},v)$, and leave the variant where we take instead the sum to the reader.

Define, for a graph $G=(V,E)$, sets of vertices $L, S\subseteq V$, $L\cap S=\emptyset$, $S\neq\emptyset$:

$$C_G(L,S)=\min_{\pi\in\Pi(L,S)}\max_{v\in S} f(G,\pi_{<,v},v)$$

Note that we want to compute the value $C_G(\emptyset,V)$.

Lemma 4 *Let $G=(V,E)$ be a graph, and $S\subseteq V$, $L\subseteq V$, $L\cap S=\emptyset$.*

1. *If $S=\{x\}$, then $C_G(L,S)=f(G,L,x)$.*
2. *Suppose $|S|\geq 2$ and $1\leq k<|S|$. Then*

$$C_G(L,S)=\min_{S'\subseteq S,|S'|=k}\max\{C_G(L,S'),C_G(L\cup S',S-S')\}$$

Proof If $S=\{x\}$, then each $\pi\in\Pi(L,S)$ first has the vertices in L in some ordering and then x. So $\pi_{<,x}=L$, and hence $\max_{v\in S} f(G,\pi_{<,v},v)=f(G,L,x)$. Part (1) now directly follows.

Suppose now that $|S|\geq 2$. Consider $S'\subseteq S$ with $S'\neq\emptyset$. Let $\pi'\in\Pi(L,S')$ fulfill

$$C_G(L,S')=\max_{v\in S'} f(G,\pi'_{<,v},v)$$

and let $\pi''\in\Pi(L\cup S',S-S')$ fulfill

$$C_G(L\cup S',S-S')=\max_{v\in S-S'} f(G,\pi''_{<,v},v)$$

By definition, π' and π'' exist. Define now a vertex ordering $\pi\in\Pi(L,S)$ as follows: first we start with the vertices in $L\cup S'$ in the same order as they appear in π', and then take the vertices in $S-S'$ in the same order as they appear in π''. I.e., we first have the vertices in L, then the vertices in S', and then the vertices in $S-S'$. For $v\in S'$, $L\subseteq\pi_{<,v}=\pi'_{<,v}$, and for $v\in S-S'$, $L\cup S'\subseteq\pi_{<,v}=\pi''_{<,v}$.

Now

$$\begin{aligned}C_G(L,S)&\leq\max_{v\in S} f(G,\pi_{<,v},v)\\&=\max\{\max_{v\in S'} f(G,\pi'_{<,v},v),\ \max_{v\in S-S'} f(G,\pi''_{<,v},v)\}\\&=\max\{C_G(L,S'),C_G(L\cup S',S-S')\}\end{aligned}$$

As this holds for each $S'\subseteq S$ with $S'\neq\emptyset$, we have

$$C_G(L,S)\leq\min_{S'\subseteq S,|S'|=k}\max\{C_G(L,S'),C_G(L\cup S',S-S')\}$$

For the other direction, let $\pi\in\Pi(L,S)$ fulfill

$$C_G(L,S)=\max_{v\in S} f(G,\pi_{<,v},v)$$

Let $S' \subseteq S$ be the set obtained by taking the first k elements in S appearing in π, i.e., $|S'| = k$ and all elements in S' appear before all elements in $S - S'$ in π. We have that $\pi \in \Pi(L \cup S', S - S')$. Let $\pi' \in \Pi(L, S')$ be obtained from π by restricting π to $L \cup S'$. Now

$$\begin{aligned} C_G(L, S) &= \max_{v \in S} f(G, \pi_{<,v}, v) \\ &= \max\{\max_{v \in S'} f(G, \pi_{<,v}, v), \max_{v \in S - S'} f(G, \pi_{<,v}, v)\} \\ &= \max\{\max_{v \in S'} f(G, \pi'_{<,v}, v), \max_{v \in S - S'} f(G, \pi_{<,v}, v)\} \\ &\leq \max\{C_G(L, S'), C_G(L \cup S', S - S')\} \end{aligned}$$

This shows the result. □

Our algorithm uses recursion, each time employing Lemma 4 with $k = \lfloor |S|/2 \rfloor$. The algorithm is given in pseudo-code in Algorithm 2.

Correctness of Algorithm 2 follows directly from Lemma 4. The running time can be estimated as follows. Let $T(k)$ be the number of recursive calls made when Recursive is called with the third argument S with $|S| = k$. Clearly, $T(1) = 1$. If $k > 1$, then for each of the $\binom{k}{\lfloor k/2 \rfloor}$ subsets of S of size $\lfloor k/2 \rfloor$, we have a recursive call with third parameter of size $\lfloor |S|/2 \rfloor$ and a recursive call with third parameter of size $\lceil |S|/2 \rceil$; and thus we use per subset S' two calls at this level of the recursion, and $T(\lfloor k/2 \rfloor) + T(\lceil k/2 \rceil)$ calls deeper in the recursion tree. So

$$T(k) \leq \binom{k}{\lfloor k/2 \rfloor} (T(\lfloor k/2 \rfloor) + T(\lceil k/2 \rceil) + 2)$$

It follows that $T(k) < 4^k$. As the time per recursive call is bounded by a polynomial in n, the total time is bounded by $O^*(4^n)$. In most cases, the dynamic programming algorithm from Sect. 3 is more practical than the recursive algorithm, as already the enumeration over all subsets of size $n/2$ is very time consuming, except for very small values of n, but for such values, the space requirements for the $O^*(2^n)$ algorithm can be expected to be small enough for modern computers.

Algorithm 2 Recursive(Graph G, vertex set L, vertex set S)

```
if |S|=1 then
  Suppose S = {v}.
  return f(G, L, v)
end if
Set opt = ∞.
for all vertex sets S' ⊆ S, |S'| = ⌊|S|/2⌋ do
  Compute v1 = Recursive(G, L, S');
  Compute v2 = Recursive(G, L ∪ S', S − S');
  Set opt = min{opt, max{v1, v2}};
end for
return opt
```

5 Linear Ordering Problems

There are several problems to which Theorems 1 and 3 can be applied. Several of these will be discussed below. Diaz et al. [6] wrote an excellent survey on vertex ordering and related problems, and their algorithmic aspects. For a survey on the relations between treewidth, pathwidth, and other parameters, we refer the reader to [1].

5.1 Treewidth

Treewidth is a well studied graph parameter. While treewidth is usually defined in terms of tree decompositions, it also has a characterization as a vertex ordering problem (see e.g., [1, 4, 5]). Using this characterization, in [3] explicit proofs of algorithms as in Theorems 1 and 3 are given for TREEWIDTH. For a formulation of treewidth in the form of (1), we refer the reader to [3].

Several improvements on these algorithms were made: using balanced separators and potential maximal cliques, a polynomial space algorithm using $O(2.9512^n)$ time was given in [3]. This was improved further with a clever method to list and count the number of potential maximal cliques to $O(2.6151^n)$ time by Fomin and Villanger [7]. Several papers give improved algorithms for TREEWIDTH, if we allow exponential space. An algorithm with $O(1.9601^n)$ time was given in 2004 by Fomin et al. [9]. This was improved further in [7, 8, 10, 17]; the current best running time is given by a recent paper by Fomin and Villanger [8], who solve TREEWIDTH in $O(1.7347^n)$ time.

5.2 Minimum Fill-In

A problem, related to treewidth, is the MINIMUM FILL-IN problem. Exact algorithms with exponential space for MINIMUM FILL-IN were obtained by Fomin et al. [9], and later improved [7, 8, 10, 17]; the currently fastest algorithm uses $O(1.7347^n)$ time and space [8]. These algorithms use the same techniques as for TREEWIDTH. The MINIMUM FILL-IN problem has important applications in Gaussian elimination.

The *minimum fill-in* of a graph $G = (V, E)$ is the minimum over all triangulations $H = (V, E_H)$ of G of $|E_H - E|$, i.e., the minimum number of edges that, when added to G, make G chordal.

For a graph $G = (V, E)$, a vertex ordering of its vertices $\pi \in \Pi(V)$, and a vertex $v \in V$, let

$$R_\pi(v) = |\{w \in V \mid \pi(w) > \pi(v) \wedge \text{ there is a path from } v \text{ to } w \text{ in } G[\pi_{\leq,v} \cup \{w\}]\}|$$

The following proposition can be shown in the same way as a similar result for TREEWIDTH in [3].

Proposition 5 *Let $G = (V, E)$ be a graph, and k a non-negative integer. The minimum fill-in of G is at most k if and only if there is a vertex ordering π of G, such that*

$$\sum_{v \in V} R_\pi(v) \leq k + |E|$$

It follows that MINIMUM FILL-IN is equivalent to

$$\min_{\pi\in\Pi(G)} \sum_{v\in V} |\{w \in V - S - \{v\} \mid \text{ there is a path from } v \text{ to } w \text{ in } G[S \cup \{v, w\}]\}|.$$

While for TREEWIDTH there are polynomial space algorithms that are faster than the $O^*(4^n)$ bound implied by Theorem 3, this remains open for MINIMUM FILL-IN.

5.3 Pathwidth

The pathwidth of a graph is usually defined in terms of path decompositions, but it has several equivalent characterizations, see e.g., [1] for an overview. Kinnersley [14] showed that pathwidth can be defined as a vertex ordering problem. We use this characterization to obtain the exact algorithms.

Definition 6 The *vertex separation number* of a vertex ordering π of $G = (V, E)$ is

$$\max_{v\in V} |\{w \in V \mid \exists x \in V : \{w, x\} \in E \wedge \pi(w) < \pi(v) \leq \pi(x)\}|.$$

The *vertex separation number* of a graph G is the minimum vertex separation number over all vertex orderings of G.

Theorem 7 (Kinnersley [14]) *The vertex separation number of a graph equals its pathwidth.*

We thus see that the VERTEX SEPARATION NUMBER is of the form for which we can apply Theorems 1 and 3: set in (1),

$$f(G, S, v) = |\{w \in S \mid \exists x \in V - S : \{w, x\} \in E\}|.$$

Very recently, Suchan and Villanger [16] obtained a faster exact algorithm for PATHWIDTH, i.e., using $O(1.9657^n)$ time and exponential space. It is open if this can be used for a faster algorithm with polynomial space.

5.4 Minimum Interval Graph Completion

Another problem, related to PATHWIDTH, which can be solved with Theorems 1 and 3 is the MINIMUM INTERVAL GRAPH COMPLETION problem. The MINIMUM INTERVAL GRAPH COMPLETION problem is the following: given a graph $G = (V, E)$, what is the minimum size of a set of edges, that, when added to G, yields an interval graph. The problem is known to be equivalent to the SUM CUT problem and the PROFILE problem, see for example [6]. In the SUM CUT problem, we look for a vertex ordering π which minimizes

$$\sum_{v\in V} |\{w \in V \mid \exists x \in V : \{w, x\} \in E \wedge \pi(w) < \pi(v) \leq \pi(x)\}|$$

A similar reformulation as for PATHWIDTH is possible, using the same f but the summation variant (2).

5.5 Cutwidth and Variants

The *cutwidth* of a vertex ordering π of a graph $G = (V, E)$ is

$$\max_{v \in V} |\{\{w, x\} \in E \mid \pi(w) \leq \pi(v) < \pi(x)\}|$$

We discussed in Sect. 2 that this problem fits the form of (1). Variants can be handled in similar ways.

The *modified cutwidth* of a vertex ordering π of a graph $G = (V, E)$ is

$$\max_{v \in V} |\{\{w, x\} \in E \mid \pi(w) < \pi(v) < \pi(x)\}|$$

The cutwidth (modified cutwidth) of a graph is the minimum cutwidth (modified cutwidth) of a vertex ordering of it. The parameters have variants for directed acyclic graphs. The cutwidth (modified cutwidth) of a directed acyclic graph $G = (V, A)$ is the minimum cutwidth (modified cutwidth) of a topological ordering of G; the latter are defined similar to the undirected counterparts.

Fitting MINIMUM CUTWIDTH for directed acyclic graphs into the form of Theorems 1 and 3 can be done as follows: when there is an arc $(v, w) \in A$ with $w \in S$, we set $f(G, S, v)$ to a very high value (e.g., $|E| + 1$), and otherwise we set $f(G, S, v)$ to the number of arcs with head in $S \cup \{v\}$ and tail in $V - S - \{v\}$. MODIFIED CUTWIDTH can be dealt with in a similar way.

5.6 Optimal Linear Arrangement

The OPTIMAL LINEAR ARRANGEMENT problem, of which the decision variant was proved NP-complete in [11], asks, given a graph $G = (V, E)$, for the minimum over all vertex orderings π of $\sum_{\{v,w\} \in E} |\pi(v) - \pi(w)|$. The following simple lemma shows that we can write the problem again in the form where we can apply Theorems 1 and 3.

Lemma 8 *For each graph $G = (V, E)$, and for each vertex ordering π of G,*

$$\sum_{\{x,y\} \in E} |\pi(x) - \pi(y)| = \sum_{v \in V} |\{\{x, y\} \in E \mid \pi(x) \leq \pi(v) < \pi(y)\}|$$

The directed variant, where we look for topological orderings π of a directed acyclic graph $G = (V, A)$ with $\sum_{(v,w) \in A} (f(w) - f(v))$ can be handled in a similar way; see also the discussion in Sect. 5.5.

5.7 Directed Feedback Arc Set

The DIRECTED FEEDBACK ARC SET is the following: given a directed graph $G = (V, A)$, find a set of arcs $F \subseteq A$ with $|F|$ as small as possible, such that $(V, A - F)$ is acyclic, i.e., each cycle in G contains at least one arc in F. It is a variant of the well known FEEDBACK VERTEX SET and DIRECTED FEEDBACK

VERTEX SET problems (which look for a set of vertices that break all cycles). (The problem to find in an undirected graph a minimum size set of edges that breaks all cycles is trivial; its weighted variant is a reformulation of the polynomial time solvable MINIMUM SPANNING TREE problem. The (DIRECTED) FEEDBACK VERTEX SET problems are trivially solvable in $O^*(2^n)$ time with linear space, and thus we have to focus only to DIRECTED FEEDBACK ARC SET.) One can also look at a weighted variant: each arc has a weight, and we look for a set of arcs that break all cycles of minimum total weight.

The following lemma shows that we can formulate (WEIGHTED) DIRECTED FEEDBACK ARC SET in a form such that Theorems 1 and 3 can be applied. Recall that a graph is acyclic, if and only if it has a topological ordering.

Lemma 9 *Let $G = (V, A)$ be a directed graph, and let $w : A \to \mathbf{N}$ be a function that assigns each arc a non-negative integer weight. Let $K \in \mathbf{N}$ be an integer. There exists a set of arcs $F \subseteq A$ with $(V, A - F)$ acyclic and $\sum_{a \in F} w(a) \leq K$, if and only if there is a vertex ordering π of G, such that $\sum_{(x,y)\in A,\ \pi(x)>\pi(y)} w((x, y)) \leq K$.*

Thus, (WEIGHTED) DIRECTED FEEDBACK ARC SET is equivalent to determining the following value, which is of the form of (2):

$$\min_{\pi \in \Pi(V)} \sum_{v \in V} \sum_{(v,x)\in A, x \in \pi_{<,v}} w((x, y)).$$

5.8 Summary

The following theorem summarizes the discussion in the paragraphs above.

Theorem 10 *Each of the following problems*: TREEWIDTH, MINIMUM FILL-IN, PATHWIDTH, SUM CUT, MINIMUM INTERVAL GRAPH COMPLETION, CUTWIDTH, DIRECTED CUTWIDTH, MODIFIED CUTWIDTH, DIRECTED MODIFIED CUTWIDTH, OPTIMAL LINEAR ARRANGEMENT, DIRECTED OPTIMAL LINEAR ARRANGEMENT *and* DIRECTED FEEDBACK ARC SET

1. *can be solved in $O^*(2^n)$ time and $O^*(2^n)$ space.*
2. *can be solved in $O^*(4^n)$ time and polynomial space.*

In each case, the $O^*(2^n)$ algorithm resembles the classic Held-Karp algorithm for TSP [13], and the $O^*(4^n)$ its variant by Gurevich and Shelah [12]. Note that for TREEWIDTH, MINIMUM FILL-IN and PATHWIDTH faster algorithms with exponential space are known [8, 16], and for TREEWIDTH a faster algorithm with polynomial space is known [9, 10].

6 Concluding Remarks

This note discusses simple exponential time algorithms for a collection of vertex ordering problems. Recently, Koivisto and Parviainen [15] have exploited the ideas

further, and showed that a tradeoff between time and space can be made, i.e., they give a range of algorithms, running in $O(c^n)$ time and $O(s^n)$ space, for various values of c and s.

Computational experiments in [3] showed that the $O^*(2^n)$ time algorithm for TREEWIDTH is practical for small graphs, especially when one applies a few optimizations to the algorithm. For practical instances, the observed use of space appears to be significantly better than the predicted 2^n, but still exponential. It would be very interesting to carry out a similar algorithm engineering study for other problems than TREEWIDTH, e.g., for some of those that were listed in Sect. 5. The versions of the algorithms that use polynomial space, at the cost of more time, are likely to be very time consuming and hence in their current form not practical: note that these already start by making a recursive call for each partition of the vertices in two sets of equal (if $|V|$ is even) size. It seems more promising to perform algorithm engineering studies using the techniques of Koivisto and Parviainen [15] for space that still is exponential, but better than $O^*(2^n)$; such an approach may give practical algorithms for larger values of n compared to the $O^*(2^n)$ time and space algorithms.

Acknowledgements We thank the anonymous referees for their comments.

References

1. Bodlaender, H.L.: A partial k-arboretum of graphs with bounded treewidth. Theor. Comput. Sci. **209**, 1–45 (1998)
2. Bodlaender, H.L., Fomin, F.V., Koster, A.M.C.A., Kratsch, D., Thilikos, D.M.: On exact algorithms for treewidth. Technical Report UU-CS-2006-032, Department of Information and Computing Sciences, Utrecht University, Utrecht, the Netherlands (2006)
3. Bodlaender, H.L., Fomin, F.V., Koster, A.M.C.A., Kratsch, D., Thilikos, D.M.: On exact algorithms for treewidth. In: Azar, Y., Erlebach, T. (eds.) Proceedings of the 14th Annual European Symposium on Algorithms, ESA 2006. Lecture Notes in Computer Science, vol. 4168, pp. 672–683. Springer, Berlin (2006)
4. Clautiaux, F., Moukrim, A., Négre, S., Carlier, J.: Heuristic and meta-heuristic methods for computing graph treewidth. RAIRO Oper. Res. **38**, 13–26 (2004)
5. Dendris, N.D., Kirousis, L.M., Thilikos, D.M.: Fugitive-search games on graphs and related parameters. Theor. Comput. Sci. **172**, 233–254 (1997)
6. Díaz, J., Petit, J., Serna, M.: A survey of graph layout problems. ACM Comput. Surv. **34**, 313–356 (2002)
7. Fomin, F.V., Villanger, Y.: Treewidth computation and extremal combinatorics. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukuewics, I. (eds.) Proceedings of the 35th International Colloquium on Automata, Languages and Programming, ICALP 2008, Part I. Lecture Notes in Computer Science, vol. 5125, pp. 210–221. Springer, Berlin (2008)
8. Fomin, F.V., Villanger, Y.: Finding induced subgraphs via minimal triangulations. In: Marion, J.-Y., Schwentick, T. (eds.) Proceedings 27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010. Dagstuhl Seminar Proceedings, vol. 5, pp. 383–394. Schloss Dagstuhl, Germany (2010). Leibniz-Zentrum für Informatik
9. Fomin, F.V., Kratsch, D., Todinca, I.: Exact (exponential) algorithms for treewidth and minimum fill-in. In: Díaz, J., Karhumäki, J., Lepistö, A., Sanella, D. (eds.) Proceedings of the 31st International Colloquium on Automata, Languages and Programming, ICALP 2004. Lecture Notes in Computer Science, vol. 3142, pp. 568–580. Springer, Berlin (2004)

10. Fomin, F.V., Kratsch, D., Todinca, I., Villanger, Y.: Exact algorithms for treewidth and minimum fill-in. SIAM J. Comput. **38**, 1058–1079 (2008)
11. Garey, M.R., Johnson, D.S., Stockmeyer, L.: Some simplified NP-complete graph problems. Theor. Comput. Sci. **1**, 237–267 (1976)
12. Gurevich, Y., Shelah, S.: Expected computation time for Hamiltonian path problem. SIAM J. Comput. **16**, 486–502 (1987)
13. Held, M., Karp, R.: A dynamic programming approach to sequencing problems. J. Soc. Ind. Appl. Math. **10**, 196–210 (1962)
14. Kinnersley, N.G.: The vertex separation number of a graph equals its path width. Inf. Process. Lett. **42**, 345–350 (1992)
15. Koivisto, M., Parviainen, P.: A space-time tradeoff for permutation problems. In: Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, pp. 484–492 (2010)
16. Suchan, K., Villanger, Y.: Computing pathwidth faster than 2^n. In: Chen, J., Fomin, F.V. (eds.) Proceedings of the 4th International Workshop on Parameterized and Exact Computation, IWPEC 2009. Lecture Notes in Computer Science, vol. 5917, pp. 324–335. Springer, Berlin (2009)
17. Villanger, Y.: Improved exponential-time algorithms for treewidth and minimum fill-in. In: Correa, J.R., Hevia, A., Kiwi, M.A. (eds.) Proceedings of the 7th Latin American Symposium on Theoretical Informatics, LATIN 2006. Lecture Notes in Computer Science, vol. 3887, pp. 800–811. Springer, Berlin (2006)