Parameterized Algorithms for Book Embedding Problems

Sujoy Bhore[1], Robert Ganian[1], Fabrizio Montecchiani[2], and Martin Nöllenburg[1(✉)]

[1] Algorithms and Complexity Group, TU Wien, Vienna, Austria
{sujoy,rganian,noellenburg}@ac.tuwien.ac.at
[2] Engineering Department, University of Perugia, Perugia, Italy
fabrizio.montecchiani@unipg.it

Abstract. A *k-page book embedding* of a graph G draws the vertices of G on a line and the edges on k half-planes (called *pages*) bounded by this line, such that no two edges on the same page cross. We study the problem of determining whether G admits a k-page book embedding both when the linear order of the vertices is fixed, called Fixed-Order Book Thickness, or not fixed, called Book Thickness. Both problems are known to be NP-complete in general. We show that Fixed-Order Book Thickness and Book Thickness are fixed-parameter tractable parameterized by the vertex cover number of the graph and that Fixed-Order Book Thickness is fixed-parameter tractable parameterized by the pathwidth of the vertex order.

1 Introduction

A *k-page book embedding* of a graph G is a drawing that maps the vertices of G to distinct points on a line, called *spine*, and each edge to a simple curve drawn inside one of k half-planes bounded by the spine, called *pages*, such that no two edges on the same page cross [21,26]; see Fig. 1 for an illustration. This kind of layout can be alternatively defined in combinatorial terms as follows. A k-page book embedding of G is a linear order $\prec$ of its vertices and a coloring of its edges which guarantee that no two edges uv, wx of the same color have their vertices ordered as $u \prec w \prec v \prec x$. The minimum k such that G admits a k-page book embedding is the *book thickness* of G, denoted by $\mathrm{bt}(G)$, also known as the *stack number* of G. Book embeddings have been extensively studied in the literature, among others due to their applications in bioinformatics, VLSI, and parallel computing (see, e.g., [8,20] and refer also to [12] for a survey). A famous result by Yannakakis [30] states that every planar graph has book thickness at

Research of Fabrizio Montecchiani supported in part by MIUR under Grant 20174LF3T8 AHeAD: efficient Algorithms for HArnessing networked Data. Robert Ganian acknowledges support by the FWF (Project P 31336, "NFPC") and is also affiliated with FI MUNI, Brno, Czech Republic. Research of Sujoy Bhore and Martin Nöllenburg is supported by the Austrian Science Fund (FWF) grant P 31119.

D. Archambault and C. D. Tóth (Eds.): GD 2019, LNCS 11904, pp. 365–378, 2019.
https://doi.org/10.1007/978-3-030-35802-0_28

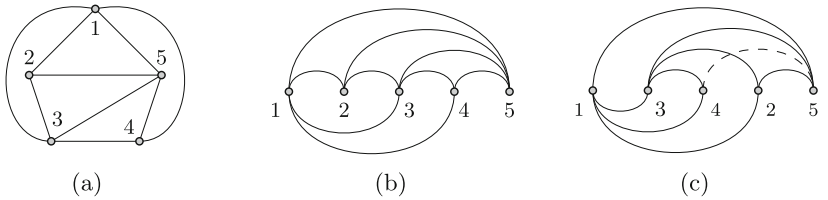


Fig. 1. (a) A planar graph G with book thickness two. (b) A 2-page book embedding of G. (c) A linear order of G such that its fixed-order book thickness is three (and the corresponding 3-page book embedding).

most four. Several other bounds are known for special graph families, for instance planar graphs with vertex degree at most four have book thickness two [3], while graphs of treewidth $w > 2$ have book thickness $w + 1$ [13,18].

Given a graph G and a positive integer k, the problem of determining whether $\mathrm{bt}(G) \leq k$, called Book Thickness, is known to be NP-complete. Namely, Bernhart and Kainen [4] proved that $\mathrm{bt}(G) \leq 2$ if and only if G is subhamiltonian, i.e., G is a subgraph of a planar Hamiltonian graph. Since deciding whether a graph is subhamiltonian is an NP-complete problem, Book Thickness is also NP-complete in general [8]. Book Thickness has been studied also when the linear order $\prec$ of the vertices is fixed, indeed, this is one of the original formulations of the problem, which arises in the context of sorting with parallel stacks [8]. We call this problem Fixed-Order Book Thickness and we denote by fo-bt$(G, \prec)$ the *fixed-order book thickness* of a graph G. Obviously, we have fo-bt$(G, \prec) \geq \mathrm{bt}(G)$, see Fig. 1. Deciding whether fo-bt$(G, \prec) \leq 2$ corresponds to testing the bipartiteness of a suitable conflict graph, and thus it can be solved in linear time. On the other hand, deciding if fo-bt$(G, \prec) \leq 4$ is equivalent to finding a 4-coloring of a circle graph and hence is an NP-complete problem [29].

Our Results. In this paper we study the parameterized complexity of Book Thickness and Fixed-Order Book Thickness. For both problems, when the answer is positive, we naturally also expect to be able to compute a corresponding k-page book embedding as a witness. While both problems are NP-complete already for small fixed values of k on general graphs, it is natural to ask which structural properties of the input (formalized in terms of structural parameters) allow us to solve these problems efficiently. Indeed, already Dujmovic and Wood [14] asked whether Book Thickness can be solved in polynomial time when the input graph has bounded treewidth [28]—a question which has turned out to be surprisingly resilient to existing algorithmic techniques and remains open to this day. Bannister and Eppstein [2] made partial progress towards answering Dujmovic and Wood's question by showing that Book Thickness is fixed-parameter tractable parameterized by the treewidth of G when $k = 2$.

We provide the first fixed-parameter algorithms for Fixed-Order Book Thickness and also the first such algorithm for Book Thickness that can be used when $k > 2$. In particular, we provide fixed-parameter algorithms for:

1. Fixed-Order Book Thickness parameterized by the vertex cover number of the graph;
2. Fixed-Order Book Thickness parameterized by the pathwidth of the graph and the vertex order; and
3. Book Thickness parameterized by the vertex cover number of the graph.

Results 1 and 2 are obtained by combining dynamic programming techniques with insights about the structure of an optimal book embedding. Result 3 then applies a kernelization technique to obtain an equivalent instance of bounded size (which can then be solved, e.g., by brute force). All three of our algorithms can also output a corresponding k-page book embedding as a witness (if it exists).

The remainder of this paper is organized as follows. Section 2 contains preliminaries and basic definitions. Results 1 and 2 on Fixed-Order Book Thickness are presented in Sect. 3, while Result 3 on Book Thickness is described in Sect. 4. Conclusions and open problems are found in Sect. 5. Some proofs are omitted due to space constraints; they are included in the full version [5].

2 Preliminaries

We use standard terminology from graph theory [10]. For $r \in \mathbb{N}$, we write $[r]$ as shorthand for the set $\{1, \ldots, r\}$. Parameterized complexity [9,11] focuses on the study of problem complexity not only with respect to the input size n but also a parameter $k \in \mathbb{N}$. The most desirable complexity class in this setting is FPT (*fixed-parameter tractable*), which contains all problems that can be solved by an algorithm running in time $f(k) \cdot n^{\mathcal{O}(1)}$, where f is a computable function. Algorithms running in this time are called *fixed-parameter algorithms*.

A k-page book embedding of a graph $G = (V, E)$ will be denoted by a pair $\langle \prec, \sigma \rangle$, where $\prec$ is a linear order of V, and $\sigma \colon E \to [k]$ is a function that maps each edge of E to one of k pages $[k] = \{1, 2, \ldots, k\}$. In a k-page book embedding $\langle \prec, \sigma \rangle$ it is required that for no pair of edges $uv, wx \in E$ with $\sigma(uv) = \sigma(wx)$ the vertices are ordered as $u \prec w \prec v \prec x$, i.e., each page is crossing-free.

We consider two graph parameters for our algorithms. A *vertex cover* C of a graph $G = (V, E)$ is a subset $C \subseteq V$ such that each edge in E has at least one end-vertex in C. The *vertex cover number* of G, denoted by $\tau(G)$, is the size of a minimum vertex cover of G. The second parameter is *pathwidth*, a classical graph parameter [27] which admits several equivalent definitions. The definition that will be most useful here is the one tied to linear orders [22]; see also [23,24] for recent works using this formulation. Given an n-vertex graph $G = (V, E)$ with a linear order $\prec$ of V such that $v_1 \prec v_2 \prec \cdots \prec v_n$, the *pathwidth* of $(G, \prec)$ is the minimum number κ such that for each vertex v_i ($i \in [n]$), there are at most κ vertices left of v_i that are adjacent to v_i or a vertex right of v_i. Formally, for each v_i we call the set $P_i = \{v_j \mid j < i, \exists q \geq i \text{ such that } v_j v_q \in E\}$ the *guard set* for v_i, and the pathwidth of $(G, \prec)$ is simply $\max_{i \in [n]} |P_i|$. The elements of the guard sets are called the *guards* (for v_i). We remark that the pathwidth of G is equal to the minimum pathwidth over all linear orders $\prec$.

3 Algorithms for Fixed-Order Book Thickness

Recall that in Fixed-Order Book Thickness the input consists of a graph $G = (V, E)$, a linear order $\prec$ of V, and a positive integer k. We assume that $V = \{v_1, v_2, \ldots, v_n\}$ is indexed such that $i < j \Leftrightarrow v_i \prec v_j$. The task is to decide if there is a page assignment $\sigma\colon E \to [k]$ such that $\langle \prec, \sigma \rangle$ is a k-page book embedding of G, i.e., whether fo-bt$(G, \prec) \leq k$. If the answer is 'YES' we shall return a corresponding k-page book embedding as a witness. In fact, our algorithms will return a book embedding with the minimum number of pages.

3.1 Parameterization by the Vertex Cover Number

As our first result, we will show that Fixed-Order Book Thickness is fixed-parameter tractable when parameterized by the *vertex cover number*. We note that the vertex cover number is a graph parameter which, while restricting the structure of the graph in a fairly strong way, has been used to obtain fixed-parameter algorithms for numerous difficult problems [1,15,16].

Let C be a minimum vertex cover of size $\tau = \tau(G)$; we remark that such a vertex cover C can be computed in time $\mathcal{O}(2^\tau + \tau \cdot n)$ [7]. Moreover, let $U = V \setminus C$. Our first observation shows that the problem becomes trivial if $\tau \leq k$.

Observation 1. *Every n-vertex graph G with a vertex cover C of size k admits a k-page book embedding with any vertex order $\prec$. Moreover, if G and C are given as input, such a book embedding can be computed in $\mathcal{O}(n + k \cdot n)$ time.*

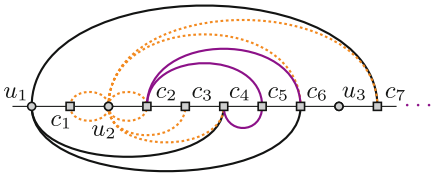
Proof. Let $C = \{c_1, \ldots, c_k\}$ be a vertex cover of size k and let σ be a page assignment on k pages defined as follows. For each $i \in [k]$ all edges uc_i with $u \in U \cup \{c_1, \ldots, c_{i-1}\}$ are assigned to page i. Now, consider the edges assigned to any page $i \in [k]$. By construction, they are all incident to vertex c_i, and thus no two of them cross each other. Therefore, the pair $\langle \prec, \sigma \rangle$ is a k-page book embedding of G and can be computed in $\mathcal{O}(n + k \cdot n)$ time. □

We note that the bound given in Observation 1 is tight, since it is known that complete bipartite graphs with bipartitions of size k and $h > k(k-1)$ have book thickness k [4] and vertex cover number k.

We now proceed to a description of our algorithm. For ease of presentation, we will add to G an additional vertex of degree 0, add it to U, and place it at the end of $\prec$ (observe that this does not change the solution to the instance).

If $\tau \leq k$ then we are done by Observation 1. Otherwise, let S be the set of all possible non-crossing page assignments of the edges whose both endpoints lie in C, and note that $|S| < \tau^{\tau^2}$ and S can be constructed in time $\mathcal{O}(\tau^{\tau^2})$ (recall that $k < \tau$ by assumption). As its first step, the algorithm branches over each choice of $s \in S$, where no pair of edges assigned to the same page crosses.

For each such non-crossing assignment s, the algorithm performs a dynamic programming procedure that runs on the vertices of the input graph in sequential (left-to-right) order. We will define a record set that the algorithm is going to compute for each individual vertex in left-to-right order. Let $c_1 \prec \ldots \prec c_\tau$ be



$$M_2(2,\alpha,s) = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Fig. 2. A partial 2-page book embedding of a graph G with a vertex cover C of size 7. The visibilities of vertices in C (squares) from u_2 are marked by dashed edges (left). Corresponding visibility matrix $M_2(2,\alpha,s)$ (right).

the ordering of vertices of C, and let $u_1 \prec \ldots \prec u_{n-\tau}$ be the ordering of vertices of U.

In order to formalize our records, we need the notion of visibility. Let $i \in [n-\tau]$ and let $E_i = \{u_j c \in E \mid j < i,\, c \in C\}$ be the set of all edges with one endpoint outside of C that lies to the left of u_i. We call $\alpha\colon E_i \to [k]$ a *valid partial page assignment* if $\alpha \cup s$ maps edges to pages in a non-crossing fashion. Now, consider a valid partial page assignment $\alpha\colon E_i \to [k]$. We say that a vertex $c \in C$ is (α, s)-*visible* to u_t (for $t \in [n-\tau]$) on page p if it is possible to draw an edge from u_t to c on page p without crossing any other edge mapped to page p by $\alpha \cup s$. Figure 2 shows the visibilities of a vertex in two pages.

Based on this notion of visibility, for an index $a \in [n-\tau]$ we can define a $k \times \tau$ *visibility matrix* $M_i(a,\alpha,s)$, where an entry (p,b) of $M_i(a,\alpha,s)$ is 1 if c_b is (α,s)-visible to u_a on page p and 0 otherwise (see Fig. 2). Intuitively, this visibility matrix captures information about the reachability via crossing-free edges (i.e., *visibility*) to the vertices in C from u_a on individual pages given a particular assignment α of edges in E_i. Note that for a given tuple (i,a,α,s), it is straightforward to compute $M_i(a,\alpha,s)$ in polynomial time.

Observe that while the number of possible choices of valid partial page assignments $\alpha\colon E_i \to [k]$ (for some $i \in [n-\tau]$) is not bounded by a function of τ, for each $i, a \in [n-\tau]$ the number of possible visibility matrices is upper-bounded by 2^{τ^2}. On a high level, the core idea in the algorithm is to dynamically process the vertices in U in a left-to-right fashion and compute, for each such vertex, a bounded-size "snapshot" of its visibility matrices—whereas for each such snapshot we will store only one (arbitrarily chosen) valid partial page assignment. We will later (in Lemma 1) show that all valid partial page assignments leading to the same visibility matrices are "interchangeable".

With this basic intuition, we can proceed to formally defining our records. Let $X = \{x \in [n-\tau] \mid \exists c \in C : u_x \text{ is the immediate successor of } c \text{ in } \prec\}$ be the set of indices of vertices in U which occur immediately after a cover vertex; we will denote the integers in X as $x_1, \ldots, x_z$ (in ascending order), and we note that $z \leq \tau$. For a vertex $u_i \in U$, we define our record set as follows: $\mathcal{R}_i(s) = \{\big(M_i(i,\alpha,s), M_i(x_1,\alpha,s), M_i(x_2,\alpha,s), \ldots, M_i(x_z,\alpha,s)\big) \mid \exists \text{ valid partial page assignment } \alpha\colon E_i \to [k]\}$. Note that each entry in $\mathcal{R}_i(s)$ captures one possible set (a "snapshot") of at most $\tau+1$ visibility matrices: the visi-

bility matrix for u_i itself, and the visibility matrices for the z non-cover vertices which follow immediately after the vertices in C. The intuition behind these latter visibility matrices is that they allow us to update our visibility matrix when our left-to-right dynamic programming algorithm reaches a vertex in C (in particular, as we will see later, for $i \in X$ it is not possible to update the visibility matrix $M_i(i, \alpha, s)$ only based on $M_{i-1}(i-1, \alpha, s)$). Along with $\mathcal{R}_i(s)$, we also store a mapping Λ_i^s from $\mathcal{R}_i(s)$ to valid partial page assignments of E_i which maps $(M_0, \ldots, M_z) \in \mathcal{R}_i(s)$ to some α such that $(M_0, \ldots, M_z) = (M_i(i, \alpha, s), M_i(x_1, \alpha, s), M_i(x_2, \alpha, s), \ldots, M_i(x_z, \alpha, s))$.

Let us make some observations about our records $\mathcal{R}_i(s)$. First, $|\mathcal{R}_i(s)| \leq 2^{\tau^3+\tau^2}$. Second, if $\mathcal{R}_{n-\tau}(s) \neq \emptyset$ for some s, since $u_{n-\tau}$ is a dummy vertex of degree 0, then there is a valid partial page assignment $\alpha\colon E_{n-\tau} \to [k]$ such that $s \cup \alpha$ is a non-crossing page assignment of *all* edges in G. Hence we can output a k-page book embedding by invoking $\Lambda_{n-\tau}^s$ on any entry in $\mathcal{R}_{n-\tau}(s)$. Third (see the full version [5] for the proof):

Observation 2. *If for all $s \in S$ it holds that $\mathcal{R}_{n-\tau}(s) = \emptyset$, then $(G, \prec, k)$ is a NO-instance of* Fixed-Order Book Thickness.

The above implies that in order to solve our instance, it suffices to compute $\mathcal{R}_{n-\tau}(s)$ for each $s \in S$. As mentioned earlier, we do this dynamically, with the first step consisting of the computation of $\mathcal{R}_1(s)$. Since $E_1 = \emptyset$, the visibility matrices $M_1(1, \emptyset, s), M_1(x_1, \emptyset, s), \ldots, M_1(x_z, \emptyset, s)$ required to populate $\mathcal{R}_1(s)$ depend only on s and are easy to compute in polynomial time.

Finally, we proceed to the dynamic step. Assume we have computed $\mathcal{R}_{i-1}(s)$. We branch over each possible page assignment β of the (at most τ) edges incident to u_{i-1}, and each tuple $\rho \in \mathcal{R}_{i-1}(s)$. For each such β and $\gamma = \Lambda_{i-1}^s(\rho)$, we check whether $\beta \cup \gamma$ is a valid partial page assignment (i.e., whether $\beta \cup \gamma \cup s$ is non-crossing); if this is not the case, we discard this pair of (β, ρ). Otherwise we compute the visibility matrices $M_i(i, \beta \cup \gamma, s), M_i(x_1, \beta \cup \gamma, s), \ldots, M_i(x_z, \beta \cup \gamma, s)$, add the corresponding tuple into $\mathcal{R}_i(s)$, and set Λ_i^s to map this tuple to $\beta \cup \gamma$. We remark that here the use of $\Lambda_{i-1}^s(\rho)$ allows us not to distinguish between $i \in X$ and $i \notin X$—in both cases, the partial page assignment γ will correctly capture the visibility matrix for u_i.

Lemma 1. *The above procedure correctly computes $\mathcal{R}_i(s)$ from $\mathcal{R}_{i-1}(s)$.*

Proof. Consider an entry $(M_0, \ldots, M_z)$ computed by the above procedure from some $\beta \cup \gamma$. Since we explicitly checked that $\beta \cup \gamma$ is a valid partial page assignment, this implies that $(M_0, \ldots, M_z) \in \mathcal{R}_i(s)$, as desired.

For the opposite direction, consider a tuple $(M_0, \ldots, M_z) \in \mathcal{R}_i(s)$. By definition, there exists some valid partial page assignment α of E_i such that $M_0 = M_i(i, \alpha, s)$, $M_1 = M_i(x_1, \alpha, s)$, $\ldots$, $M_z = M_i(x_z, \alpha, s)$. Now let β be the restriction of α to the edges incident to u_{i-1}, and let γ' be the restriction of α to all other edges (i.e., all those not incident to u_{i-1}). Since $\gamma' \cup s$ is non-crossing and in particular γ' is a valid partial page assignment for E_{i-1}, $\mathcal{R}_{i-1}(s)$ must contain an entry $\omega = (M_{i-1}(i-1, \gamma', s), \ldots, (M_{i-1}(x_z, \gamma', s))$—let $\gamma = \Lambda_{i-1}^s(\omega)$.

To conclude the proof, it suffices to show that (1) $\beta \cup \gamma$ is a valid partial page assignment, and (2) $(M_i(i, \beta \cup \gamma', s), \ldots, M_i(x_z, \beta \cup \gamma', s))$, which is the original tuple corresponding to the hypothetical α, is *equal* to $(M_i(i, \beta \cup \gamma, s), \ldots, M_i(x_z, \beta \cup \gamma, s))$, which is the entry our algorithm computes from β and γ. Point (1) follows from the fact that $M_{i-1}(i-1, \gamma', s) = M_{i-1}(i-1, \gamma, s)$ in conjunction with the fact that u_{i-1} is adjacent only to vertices in C. Point (2) then follows by the same argument, but applied to each visibility matrix in the respective tuples: for each $x \in X$ we have $M_{i-1}(x, \gamma', s) = M_{i-1}(x, \gamma, s)$—meaning that the visibilities of u_x were identical before considering the edges incident to u_{i-1}—and so assigning these edges to pages as prescribed by β leads to an identical outcome in terms of visibility. □

This proves the correctness of our algorithm. The runtime is upper-bounded by the product of $|S| < \tau^{\tau^2}$ (the initial branching factor), n (the number of times we compute a new record set $\mathcal{R}_i(s)$), and $2^{\tau^3+\tau^2} \cdot \tau^\tau$ (to consider all combinations of γ and β so to compute a new record set from the previous one). A minimum-page book embedding can be computed by trying all possible choices for $k \in [\tau]$. We summarize Result 1 below.

Theorem 1. *There is an algorithm which takes as input an n-vertex graph G with a vertex order $\prec$, runs in time $2^{\mathcal{O}(\tau^3)} \cdot n$ where τ is the vertex cover number of G, and computes a page assignment σ such that $(\prec, \sigma)$ is a (fo-bt$(G, \prec)$)-page book embedding of G.*

3.2 Parameterization by the Pathwidth of the Vertex Ordering

As our second result, we show that Fixed-Order Book Thickness is fixed-parameter tractable parameterized by the pathwidth of $(G, \prec)$. We note that while the pathwidth of G is always upper-bounded by the vertex cover number, this does not hold when we consider a fixed ordering $\prec$, and hence this result is incomparable to Theorem 1. For instance, if G is a path, it has arbitrarily large vertex cover number while $(G, \prec)$ may have a pathwidth of 1, while on the other hand if G is a star, it has a vertex cover number of 1 while $(G, \prec)$ may have arbitrarily large pathwidth. To begin, we can show that the pathwidth of $(G, \prec)$ provides an upper bound on the number of pages required for an embedding (see the full version [5] for the proof).

Lemma 2. *Every n-vertex graph $G = (V, E)$ with a linear order $\prec$ of V such that $(G, \prec)$ has pathwidth k admits a k-page book embedding $\langle \prec, \sigma \rangle$, which can be computed in $\mathcal{O}(n + k \cdot n)$ time.*

We note that the bound given in Lemma 2 is also tight for the same reason as for Observation 1: complete bipartite graphs with bipartitions of size k and $h > k(k-1)$ have book thickness k [4], but admit an ordering $\prec$ with pathwidth k.

We now proceed to a description of our algorithm. Our input consists of the graph G, the vertex ordering $\prec$, and an integer k that upper-bounds the desired number of pages in a book embedding. Let κ be our parameter, i.e., the

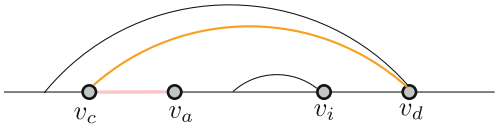


Fig. 3. An assignment of the edges of S_i to a page p, where the edge v_cv_d is the (α, i, p)-important edge of v_a. Any vertex w with $v_c \prec w \prec v_a$ is visible to v_a, and any vertex $w' \prec v_c$ is not visible to v_a.

pathwidth of $(G, \prec)$; observe that due to Lemma 2, we may assume that $k \leq \kappa$. The algorithm performs a dynamic programming procedure on the vertices $v_1, v_2, \ldots, v_n$ of the input graph G in right-to-left order along $\prec$. For technical reasons, we initially add a vertex v_0 of degree 0 to G and place it to the left of v_1 in $\prec$; note that this does not increase the pathwidth of G.

We now adapt the concept of *visibility* introduced in Sect. 3.1 for use in this algorithm. First, let us expand our notion of guard set (see Sect. 2) as follows: for a vertex v_i, let $P^*_{v_i} = \{g^i_1, \ldots, g^i_m\}$ where for each $j \in [m-1]$, g^i_j is the j-th guard of v_i in reverse order of $\prec$ (i.e., g^i_1 is the guard that is nearest to v_i in $\prec$), and $g^i_m = v_0$. For a vertex v_i, let $E_i = \{v_av_b \mid v_av_b \in E, b > i\}$ be the set of all edges with at least one endpoint to the right of v_i and let $S_i = \{g^i_jv_b \mid g^i_j \in P^*_{v_i}, g^i_jv_b \in E_i\}$ be the restriction of E_i to edges between a vertex to the right of v_i and a guard in $P^*_{v_i}$. An assignment $\alpha\colon E_i \to [k]$ is called a *valid partial page assignment* if α maps the edges in E_i to pages in a non-crossing manner. Given a valid partial page assignment $\alpha\colon E_i \to [k]$ and a vertex v_a with $a \leq i$, we say a vertex v_x $(x < a)$ is *α-visible* to v_a on a page p if it is possible to draw the edge v_av_x in page p without crossing any other edge mapped to p by α.

Before we proceed to describing our algorithm, we will show that the visibilities of vertices w.r.t. valid partial page assignments exhibit a certain regularity property. Given $a \leq i \leq n$, $p \in [k]$, and a valid partial page assignment α of E_i, let the (α, i, p)-*important* edge of v_a be the edge $v_cv_d \in S_i$ with the following properties: (1) $\alpha(v_cv_d) = p$, (2) $c < a$, and (3) $|a - c|$ is minimum among all such edges in S_i. If multiple edges with these properties exist, we choose the edge with minimum $|d - c|$. Intuitively, the (α, i, p)-important edge of v_a is simply the shortest edge of S_i which encloses v_a on page p; note that it may happen that v_a has no (α, i, p)-important edge. Observe that, if the edge exists, its left endpoint is $v_c \in P^*_{v_i}$, and we call v_c the (α, i, p)-*important* guard of v_a. The next observation easily follows from the definition of (α, i, p)-important edge, see also Fig. 3.

Observation 3. *If v_a has no (α, i, p)-important edge, then every vertex v_x with $x < a$ is α-visible to v_a. If the (α, i, p)-important guard of v_a is v_c, then v_x $(x < a)$ is α-visible to v_a if and only if $x \geq c$.*

Observation 3 not only provides us with a way of handling vertex visibilities in the pathwidth setting, but also allows us to store all the information we

require about vertex visibilities in a more concise way than via the matrices used in Sect. 3.1. For an index $i \in [n]$, a vertex v_a where $a \leq i$ and a valid partial page assignment α, we define the *visibility vector* $U_i(v_a, \alpha)$ as follows: the p-th component of $U_i(v_a, \alpha)$ is the (α, i, p)-important guard of v_a, and $\diamond$ if v_a has no (α, i, p)-important guard. Observe that since the number of pages is upper-bounded by κ by assumption and the cardinality of $P^*_{v_i}$ is at most $\kappa + 1$, there are at most $(\kappa + 2)^\kappa$ possible distinct visibility vectors for any fixed i.

Observe that thanks to Observation 3 the visibility vector $U_i(v_i, \alpha)$ provides us with complete information about the visibility of vertices v_b $(b < i)$ from v_i—notably, v_b is not α-visible to v_i on page p if and only if v_b lies to the left of the (α, i, p)-important guard $U_i(v_i, \alpha)[p]$ (and, in particular, if $U_i(v_i, \alpha)[p] = \diamond$ then every such v_b is α-visible to v_i on page p). On a high level, the algorithm will traverse vertices in right-to-left order along $\prec$ and store the set of all possible visibility vectors at each vertex. To this end, it will use the following observation to update its visibility vectors.

Observation 4. *Let α be a valid partial page assignment of E_i and p be a page. If $v_{i-1} \notin P^*_{v_i}$, then a vertex v_b $(b < i - 1)$ is α-visible to v_{i-1} on page p if and only if v_b is α-visible to v_i on page p.*

Proof. By definition v_{i-1} and v_i are consecutive in $\prec$. Let v_b (for $b < i - 1$) be a vertex that is α-visible to v_{i-1} on page p. If v_b is not α-visible to v_i on p, then there must be a vertex w between v_{i-1} and v_i that is incident to an edge in E_i separating v_{i-1} and v_i on page p. But this contradicts that v_{i-1} and v_i are consecutive in $\prec$. The other direction follows by the same argument. ◻

There is, however, a caveat: Observation 4 does not (and in fact cannot) allow us to compute the new visibility vector if $v_{i-1} \in P^*_i$. To circumvent this issue, our algorithm will not only store the visibility vector $U_i(v_i, \alpha)$ but also the visibility vectors for each guard of v_i. We now prove that we can compute the visibility vector for any vertex from the visibility vectors of the guards—this is important when updating our records, since we will need to obtain the visibility records for new guards that are introduced at some step of the algorithm.

Lemma 3. *Let $v_a \prec v_i$, α be a valid partial page assignment of E_i, $p \in [k]$ be a page, and assume $v_a \notin P^*_i$. Let $v_b \in P^*_i \cup \{v_i\}$ be such that $b > a$ and $|b - a|$ is minimized, i.e., v_b is the first guard to the right of v_a. Then $U_i(v_a, \alpha) = U_i(v_b, \alpha)$.*

Proof. Let v_x for $x < a$ be any vertex that is α-visible to v_a in page p and assume v_x is not α-visible to v_b. Then there must be an edge $wz \in E_i$ separating v_a from v_b in page p, i.e., $v_a \prec w \prec v_b$. But in that case w is a guard in P^*_i closer to v_a contradicting the choice of v_b. Conversely, let v_x for $x < a$ be a vertex that is not α-visible to v_a in page p. Then there must be an edge $wz \in E_i$ separating v_x from v_a on page p. Then edge wz also separates v_x from v_b and v_x is not α-visible to v_b. Therefore, the visibility vectors $U_i(v_a, \alpha)$ and $U_i(v_b, \alpha)$ corresponding to the vertices v_a and v_b, respectively, are equal. ◻

We can now formally define our record set as $Q_i = \{(U_i(v_i, \alpha), U_i(g_1^i, \alpha), \ldots, U_i(g_{m-1}^i, \alpha)) \mid \exists$ valid partial page assignment $\alpha\colon E_i \to [k]\}$, where each individual element (record) in Q_i can be seen as a queue starting with the visibility vector for v_i and then storing the visibility vectors for individual guards (note that there is no reason to store an "empty" visibility vector for g_m^i). To facilitate the construction of a solution, we will also store a function Λ_i from Q_i to valid partial page assignments of E_i which maps each tuple $\omega \in Q_i$ to some α such that $\omega = (U_i(v_i, \alpha), U_i(g_1^i, \alpha), \ldots, U_i(g_{m-1}^i, \alpha))$.

Let us make some observations about our records Q_i. First of all, since there are at most $(\kappa+2)^\kappa$ many visibility vectors, $|Q_i| \leq (\kappa+2)^{\kappa^2}$. Second, if $|Q_0| > 0$ then, since $E_0 = E$, the mapping $\Lambda_0(\omega)$ will produce a valid page assignment of E for any $\omega \in Q_0$. On the other hand, if G admits a k-page book embedding α with order $\prec$, then α witnesses the fact that Q_0 cannot be empty. Hence, the algorithm can return one, once it correctly computes Q_0 and Λ_0.

The computation is carried out dynamically and starts by setting $Q_n = \{\omega\}$, where $\omega = (\diamond)$, and $\Lambda_n(\omega) = \emptyset$. For the inductive step, assume that we have correctly computed Q_i and Λ_i, and the aim is to compute Q_{i-1} and Λ_{i-1}. For each $\omega = (\omega_1, \ldots, \omega_m) \in Q_i$, we compute an intermediate record ω' which represents the visibility vector of v_{i-1} w.r.t. $\alpha = \Lambda_i(\omega)$ as follows:

- if $v_{i-1} \in P_i^*$, then $\omega' = (\omega_2, \ldots, \omega_m)$, and
- if $v_{i-1} \notin P_i^*$, then $\omega' = (\omega_1, \ldots, \omega_m)$ (Recall Observation 4).

We now need to update our intermediate record ω' to take into account the new guards. In particular, we expand ω' by adding, for each new guard $g_j^{i-1} \in P_{i-1}^* \setminus P_i^*$, an intermediate visibility vector $U_{i-1}(g_j^{i-1}, \alpha)$ at the appropriate position in ω' (i.e., mirroring the ordering of guards in P_{i-1}^*). Recalling Lemma 3, we compute this new intermediate visibility vector $U_{i-1}(g_j^{i-1}, \alpha)$ by copying the visibility vector that immediately succeeds it in ω'.

Next, let $F_{i-1} = E_{i-1} \setminus E_i$ be the at most κ new edges that we need to account for, and let us branch over all assignments $\beta\colon F_{i-1} \to [k]$. For each such β, we check whether $\alpha \cup \beta$ is a valid partial page assignment of E_{i-1}, i.e., whether the new edges in F_{i-1} do not cross with each other or other edges in E_i when following the chosen assignment β and the assignment α obtained from Λ_i. As expected, we discard any β such that $\alpha \cup \beta$ is not valid.

Our final task is now to update the intermediate visibility vectors $U_{i-1}(*, \alpha)$ (with $*$ being a placeholder) to $U_{i-1}(*, \alpha \cup \beta)$. This can be done in a straightforward way by, e.g., looping over each edge $e \in F_{i-1}$, obtaining the page $p = \beta(e)$ that e is mapped to, reading $U_{i-1}(*, \alpha)[p]$ and replacing that value by the guard g incident to e if g occurs to the right of $U_{i-1}(*, \alpha)[p]$ and to the left of $*$. Finally, we enter the resulting record ω' into Q_{i-1}.

Lemma 4. *The above procedure correctly computes Q_{i-1} from Q_i.*

Proof. Consider an entry ω' computed by the above procedure from some $\alpha \cup \beta$ and ω. Since we explicitly checked that $\alpha \cup \beta$ is a valid partial page assignment for E_{i-1}, there must exist a record $(U_{i-1}(v_{i-1}, \alpha \cup \beta), U_{i-1}(g_1^{i-1}, \alpha \cup$

$\beta), \ldots, U_{i-1}(g_{m-1}^{i-1})) \in Q_{i-1}$, and by recalling Observation 3, Lemma 3 and Observation 4 it can be straightforwardly verified that this record is equal to ω'.

For the opposite direction, consider a tuple $\omega_0 \in Q_{i-1}$ that arises from the valid partial page assignment γ of E_{i-1}, and let β, α be the restrictions of γ to F_{i-1} and E_i, respectively. Since α is a valid partial page assignment of E_i, there must exist a tuple $\omega \in Q_i$ that arises from α. Let $\alpha' = \Lambda_i(\omega)$. To conclude the proof, it suffices to note that during the branching stage the algorithm will compute a record from a combination of α' (due to ω being in Q_i) and β, and the record computed in this way will be precisely ω_0. □

This proves the correctness of the algorithm. The runtime is upper bounded by $\mathcal{O}(n \cdot (\kappa+2)^{\kappa^2} \cdot \kappa^{\kappa})$ (the product of the number of times we compute a new record, the number of records and the branching factor for β). A minimum-page book embedding can be obtained by trying all possible choices for $k \in [\kappa]$). We summarize Result 2 below.

Theorem 2. *There is an algorithm which takes as input an n-vertex graph $G = (V, E)$ with a vertex ordering $\prec$ and computes a page assignment σ of E such that $(\prec, \sigma)$ is a (fo-bt$(G, \prec)$)-page book embedding of G. The algorithm runs in $n \cdot \kappa^{\mathcal{O}(\kappa^2)}$ time where κ is the pathwidth of $(G, \prec)$.*

4 Algorithms for Book Thickness

We now turn our attention to the general definition of book thickness (without a fixed vertex order). We show that, given a graph G, in polynomial time we can construct an equivalent instance G^* whose size is upper-bounded by a function of $\tau(G)$. Such an algorithm is called a *kernelization* and directly implies the fixed-parameter tractability of the problem with this parameterization [9,11].

Theorem 3. *There is an algorithm which takes as input an n-vertex graph $G = (V, E)$ and a positive integer k, runs in time $\mathcal{O}(\tau^{\tau^{\mathcal{O}(\tau)}} + 2^{\tau} \cdot n)$ where $\tau = \tau(G)$ is the vertex cover number of G, and decides whether $bt(G) \leq k$. If the answer is positive, it can also output a k-page book embedding of G.*

Proof. If $k > \tau$, by Observation 1 we can immediately conclude that G admits a k-page book embedding. Hence we shall assume that $k \leq \tau$. We will also compute a vertex cover C of size τ in time $\mathcal{O}(2^{\tau} \cdot n)$ using well-known results [7].

For any subset $U \subseteq C$ we say that a vertex of $V \setminus C$ is of *type* U if its set of neighbors is equal to U. This defines an equivalence relation on $V \setminus C$ and partitions $V \setminus C$ into at most $\sum_{i=0}^{\tau} \binom{\tau}{i} = 2^{\tau}$ distinct types. In what follows, we denote by V_U the set of vertices of type U. We claim the following.

Claim. Let $v \in V_U$ such that $|V_U| \geq 2 \cdot k^{\tau} + 2$. Then G admits a k-page book embedding if and only if $G' = G \setminus \{v\}$ does. Moreover, a k-page book embedding of G' can be extended to such an embedding for G in linear time.

Proof. (of the Claim). One direction is trivial, since removing a vertex from a book embedding preserves the property of being a book embedding of the resulting graph. So let $\langle \prec, \sigma \rangle$ be a k-page book embedding of G'. We prove that a k-page book embedding of G can be easily constructed by inserting v right next to a suitable vertex u in V_U and by assigning the edges of v to the same pages as the corresponding edges of u. We say that two vertices $u_1, u_2 \in V_U$ are *page equivalent*, if for each vertex $w \in U$, the edges u_1w and u_2w are both assigned to the same page according to σ. Each vertex in V_U has degree exactly $|U|$, hence this relation partitions the vertices of V_U into at most $k^{|U|} \leq k^\tau$ sets. Since $|V_U| \setminus \{v\} \geq 2 \cdot k^\tau + 1$, at least three vertices of this set, which we denote by u_1, u_2, and u_3, are page equivalent. Consider now the graph induced by the edges of these three vertices that are assigned to a particular page. By the above argument, such a graph is a $K_{h,3}$, for some $h > 0$. However, since already $K_{2,3}$ does not admit a 1-page book embedding, we have $h \leq 1$, that is, each u_i has at most one edge on each page. Then we can extend $\prec$ by introducing v right next to u_1 and assign each edge vw to the same page as u_1w. Since each such edge vw runs arbitrarily close to the corresponding crossing-free edge u_1w, this results in a k-page book embedding of G and concludes the proof of the claim. □

We now construct a kernel G^* from G of size $\mathcal{O}(k^\tau)$ as follows. We first classify each vertex of G based on its type. We then remove an arbitrary subset of vertices from each set V_U with $|V_U| > 2 \cdot k^\tau + 1$ until $|V_U| = 2 \cdot k^\tau + 1$. Thus, constructing G^* can be done in $\mathcal{O}(2^\tau + \tau \cdot n)$ time, where 2^τ is the number of types and $\tau \cdot n$ is the maximum number of edges of G. From our claim above we can conclude that G^* admits a k-page book embedding if and only if G does. Determining the book thickness of G^* can be done by guessing all possible linear orders and page assignments in $O(k^\tau! \cdot k^{k^\tau}) = O(\tau^{\tau^{O(\tau)}})$ time. A k-page book embedding of G^* (if any) can be extended to one of G by iteratively applying the constructive procedure from the proof of the above claim, in $O(\tau \cdot n)$ time. □

The next corollary easily follows from Theorem 3, by applying a binary search on the number of pages $k \leq \tau$ and by observing that a vertex cover of minimum size τ can be computed in $2^{\mathcal{O}(\tau)} + \tau \cdot n$ time [7].

Corollary 1. *Let G be a graph with n vertices and vertex cover number τ. A book embedding of G with minimum number of pages can be computed in $\mathcal{O}(\tau^{\tau^{\mathcal{O}(\tau)}} + \tau \log \tau \cdot n)$ time.*

5 Conclusions and Open Problems

We investigated the parameterized complexity of Book Thickness and Fixed-Order Book Thickness. We proved that both problems can be parameterized by the vertex cover number of the graph, and that the second problem can be parameterized by the pathwidth of the fixed linear order. The algorithm for Book Thickness is the first fixed-parameter algorithm that works for general values of k, while, to the best of our knowledge, no such algorithms were known for Fixed-Order Book Thickness.

We believe that our techniques can be extended to the setting in which we allow edges on the same page to cross, with a given budget of at most c crossings over all pages. This problem has been studied by Bannister and Eppstein [2] with the number of pages k restricted to be either 1 or 2. It would also be interesting to investigate the setting where an upper bound on the maximum number of crossings *per edge* is given as part of the input, which is studied in [6].

The main question that remains open is whether Book Thickness (and Fixed-Order Book Thickness) can be solved in polynomial time (and even fixed-parameter time) for graphs of bounded treewidth, which was asked by Dujmović and Wood [14]. As an intermediate step towards solving this problem, we ask whether the two problems can be solved efficiently when parameterized by the treedepth [25] of the graph. Treedepth restricts the graph structure in a stronger way than treewidth, and has been used to obtain algorithms for several problems which have proven resistant to parameterization by treewidth [17,19].

References

1. Bannister, M.J., Cabello, S., Eppstein, D.: Parameterized complexity of 1-planarity. J. Graph Algorithms Appl. **22**(1), 23–49 (2018). https://doi.org/10.7155/jgaa.00457
2. Bannister, M.J., Eppstein, D.: Crossing minimization for 1-page and 2-page drawings of graphs with bounded treewidth. J. Graph Algorithms Appl. **22**(4), 577–606 (2018). https://doi.org/10.7155/jgaa.00479
3. Bekos, M.A., Gronemann, M., Raftopoulou, C.N.: Two-page book embeddings of 4-planar graphs. Algorithmica **75**(1), 158–185 (2016). https://doi.org/10.1007/s00453-015-0016-8
4. Bernhart, F., Kainen, P.C.: The book thickness of a graph. J. Comb. Theory Ser. B **27**(3), 320–331 (1979). https://doi.org/10.1016/0095-8956(79)90021-2
5. Bhore, S., Ganian, R., Montecchiani, F., Nöllenburg, M.: Parameterized algorithms for book embedding problems. CoRR abs/1908.08911 (2019). http://arxiv.org/abs/1908.08911
6. Binucci, C., Di Giacomo, E., Hossain, M.I., Liotta, G.: 1-page and 2-page drawings with bounded number of crossings per edge. Eur. J. Comb. **68**, 24–37 (2018). https://doi.org/10.1016/j.ejc.2017.07.009
7. Chen, J., Kanj, I.A., Xia, G.: Improved upper bounds for vertex cover. Theor. Comput. Sci. **411**(40–42), 3736–3756 (2010). https://doi.org/10.1016/j.tcs.2010.06.026
8. Chung, F., Leighton, F., Rosenberg, A.: Embedding graphs in books: a layout problem with applications to VLSI design. SIAM J. Algebraic Discret. Methods **8**(1), 33–58 (1987). https://doi.org/10.1137/0608002
9. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21275-3
10. Diestel, R.: Graph Theory. Graduate Texts in Mathematics, vol. 173. Springer, Heidelberg (2012)
11. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. TCS. Springer, London (2013). https://doi.org/10.1007/978-1-4471-5559-1

12. Dujmović, V., Wood, D.R.: On linear layouts of graphs. Discrete Math. Theor. Comput. Sci. **6**(2), 339–358 (2004)
13. Dujmovic, V., Wood, D.R.: Graph treewidth and geometric thickness parameters. Discrete Comput. Geom. **37**(4), 641–670 (2007). https://doi.org/10.1007/s00454-007-1318-7
14. Dujmović, V., Wood, D.R.: On the book thickness of k-trees. Discrete Math. Theor. Comput. Sci. **13**(3), 39–44 (2011)
15. Fellows, M.R., Lokshtanov, D., Misra, N., Rosamond, F.A., Saurabh, S.: Graph layout problems parameterized by vertex cover. In: Algorithms and Computation (ISAAC 2008), pp. 294–305 (2008). https://doi.org/10.1007/978-3-540-92182-0_28
16. Ganian, R.: Improving vertex cover as a graph parameter. Discrete Math. Theor. Comput. Sci. **17**(2), 77–100 (2015)
17. Ganian, R., Ordyniak, S.: The complexity landscape of decompositional parameters for ILP. Artif. Intell. **257**, 61–71 (2018). https://doi.org/10.1016/j.artint.2017.12.006
18. Ganley, J.L., Heath, L.S.: The pagenumber of k-trees is $O(k)$. Discrete Appl. Math. **109**(3), 215–221 (2001). https://doi.org/10.1016/S0166-218X(00)00178-5
19. Gutin, G.Z., Jones, M., Wahlström, M.: The mixed Chinese postman problem parameterized by pathwidth and treedepth. SIAM J. Discrete Math. **30**(4), 2177–2205 (2016). https://doi.org/10.1137/15M1034337
20. Haslinger, C., Stadler, P.F.: RNA structures with pseudo-knots: graph-theoretical, combinatorial, and statistical properties. Bull. Math. Biol. **61**(3), 437–467 (1999). https://doi.org/10.1006/bulm.1998.0085
21. Kainen, P.C.: Some recent results in topological graph theory. In: Bari, R.A., Harary, F. (eds.) Graphs and Combinatorics, pp. 76–108. Springer, Berlin (1974). https://doi.org/10.1007/BFb0066436
22. Kinnersley, N.G.: The vertex separation number of a graph equals its path-width. Inf. Process. Lett. **42**(6), 345–350 (1992). https://doi.org/10.1016/0020-0190(92)90234-M
23. Lodha, N., Ordyniak, S., Szeider, S.: SAT-encodings for special treewidth and pathwidth. In: Gaspers, S., Walsh, T. (eds.) SAT 2017. LNCS, vol. 10491, pp. 429–445. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66263-3_27
24. Mallach, S.: Linear ordering based MIP formulations for the vertex separation or pathwidth problem. In: Brankovic, L., Ryan, J., Smyth, W.F. (eds.) IWOCA 2017. LNCS, vol. 10765, pp. 327–340. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78825-8_27
25. Nešetřil, J., Ossona de Mendez, P.: Sparsity. AC, vol. 28. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27875-4
26. Ollmann, L.T.: On the book thicknesses of various graphs. In: 4th Southeastern Conference on Combinatorics, Graph Theory and Computing, vol. 8, p. 459 (1973)
27. Robertson, N., Seymour, P.D.: Graph minors. I. Excluding a forest. J. Comb. Theory Ser. B **35**(1), 39–61 (1983). https://doi.org/10.1016/0095-8956(83)90079-5
28. Robertson, N., Seymour, P.D.: Graph minors. II. Algorithmic aspects of tree-width. J. Algorithms **7**(3), 309–322 (1986). https://doi.org/10.1016/0196-6774(86)90023-4
29. Unger, W.: The complexity of colouring circle graphs. In: Finkel, A., Jantzen, M. (eds.) STACS 1992. LNCS, vol. 577, pp. 389–400. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-55210-3_199. (extended abstract)
30. Yannakakis, M.: Embedding planar graphs in four pages. J. Comput. Syst. Sci. **38**(1), 36–67 (1989). https://doi.org/10.1016/0022-0000(89)90032-9