Imbalance is Fixed Parameter Tractable*

Daniel Lokshtanov[1], Neeldhara Misra[2], Saket Saurabh[2]

[1] University of California San Diego, San Diego, USA.
daniello@ii.uib.no
[2] Institute of Mathematical Sciences, Chennai, India.
{neeldhara|saket}@imsc.res.in

Abstract. In the Imbalance Minimization problem we are given a graph $G = (V, E)$ and an integer b and asked whether there is an ordering $v_1 \dots v_n$ of V such that the sum of the imbalance of all the vertices is at most b. The imbalance of a vertex v_i is the absolute value of the difference between the number of neighbors to the left and right of v_i. The problem is also known as the Balanced Vertex Ordering problem and it finds many applications in graph drawing. We show that this problem is fixed parameter tractable and provide an algorithm that runs in time $2^{O(b \log b)} \cdot n^{O(1)}$. This resolves an open problem of Kára et al. [COCOON 2005].

1 Introduction

Graph layout problems are combinatorial optimization problems where the objective is to find a permutation of the vertex set that optimizes some function of interest. In this paper we focus on the problem of determining the *imbalance* of a given graph. Given a permutation $v_1 \dots v_n$ on the vertex set we define the left and right neighborhood of v_i to be $N_L(v_i) = N(v_i) \cap \{v_1 \dots v_{i-1}\}$ and $N_R(v_i) = N(v_i) \cap \{v_{i+1} \dots v_n\}$ respectively. The *imbalance* of v_i is $||N_L(v_i)| - |N_R(v_i)||$ and the imbalance of the graph G given the ordering $v_1 \dots v_n$ is simply the sum of the imbalances of the individual vertices. The imbalance of G is the minimum imbalance of G over all orderings of V, and an ordering yielding this imbalance is called an *optimal* ordering.

The Imbalance Minimization problem was first defined by Biedl et al. [2]. Here, we are given a graph $G = (V, E)$ on n vertices and m edges and an integer b and asked whether the imbalance of G is at most b. The problem finds a variety of applications in graph drawing [8, 9, 17, 18, 15]. The computational aspects of graph imbalance were studied by Biedl et al. [2], who show that computing the imbalance of G is NP-hard. Later Kára et al. [10, 11] proved that in fact the problem remains NP-hard for planar graphs with maximum degree six and for 5-regular graphs. In the same paper Kára et al. [10, 11] showed that for every fixed value of b the problem can be solved in time $O(n^b(n + m))$. They ask whether it is possible to give an algorithm for Imbalance Minimization such that the exponent of n is independent of b. More specifically, they ask for the existence of a *fixed parameter tractable* algorithm which is an algorithm with running time $f(b) \cdot n^{O(1)}$ for some function f depending only on b. Parameterized Complexity provides a framework for the study of such algorithms. We refer to [5, 6, 14] for an introduction to parameterized algorithms. Subsequently the problem was studied by Gaspers et al. [7] who prove that Imbalance Minimization in fact is equivalent to Graph Cleaning. Gaspers et al. [7] use this equivalence to obtain an algorithm to check whether the input graph G has imbalance at most b running in time $O(n^{\lfloor b/2 \rfloor}(n + m))$.

In this paper we provide an algorithm running in time $b^{O(b)} \cdot n^{O(1)}$, thereby improving the previous best algorithm by Gaspers et al. [7] running in time $O(n^{\lfloor b/2 \rfloor}(n+m))$ and resolving the question posed by Kára et al. [10, 11]. Whereas the previous algorithms are purely combinatorial, we exploit a connection between the imbalance, cutwidth and treewidth of the input graph and combine this with a dynamic programming approach. On the way we prove that given a graph G

* A preliminary version of this paper appeared in the proceedings of COCOON 2010.

with treewidth at most t and maximum degree Δ an optimal ordering of G can be computed in time $t! \cdot \Delta^{O(t)} \cdot n^{O(1)}$. This shows that the Imbalance Minimization problem is fixed parameter tractable when parameterized by the treewidth and the maximum degree of the input graph. This is in stark contrast to the seemingly similar Cutwidth Minimization problem for which only an $O(n^{t^2\Delta})$ time algorithm is known [16].

2 Preliminaries

We use $G = (V, E)$ to refer to a graph with vertex set V (with $n := |V|$), and $T = (V_T, F)$ to denote trees on the vertex set V_T, with one distinguished vertex $r \in V_T$ called the *root* of the tree. The vertices of the tree T will be called *nodes*. For a tree node $t \in V_T$, use the notation T_t to mean the set of all vertices $t' \in T$ that are descendants of t.

A graph $G' = (V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$. The subgraph G' is called an *induced subgraph* of G if $E' = \{uv \in E \mid u, v \in V'\}$, in this case, G' is also called the subgraph *induced by* V' and denoted with $G[V']$. By $N(u)$ we denote the (open) neighborhood of u, that is the set of all vertices adjacent to u, and by $N[u] = N(u) \cup \{u\}$. By degree of a vertex u we mean $|N(u)|$ and denote it by $d(u)$. We use $\Delta(G)$ to denote $\max_{v \in G}\{d(v)\}$, the maximum degree of the graph G.

We use $[n]$ to refer to the set $\{1, 2, \ldots, n\}$. In the interest of simplicity, we let $V = [n]$ (we will continue to use u, v and so on to denote vertices). Further, if π is a permutation of the vertex set, we define the *left* and *right* neighborhoods of v, respectively, as:

$$N_l(v, \pi) := N(v) \cap \{u \in V \mid \ \pi(u) < \pi(v)\}$$
$$N_r(v, \pi) := N(v) \cap \{u \in V \mid \ \pi(u) > \pi(v)\}$$

Definition 1. (Imbalance of G with respect to π.) *Given a graph $G = (V, E)$ and a permutation $\pi : V \to V$, the imbalance of G with respect to π, $\mathcal{I}(G, \pi)$, is given by*

$$\mathcal{I}(G, \pi) := \sum_{v \in V} ||N_l(v, \pi)| - |N_r(v, \pi)||.$$

Definition 2. (Imbalance of G) *Given a graph $G = (V, E)$ the imbalance of $\mathcal{G}$, $\mathcal{I}(G)$, is given by*

$$\mathcal{I}(G) := min_{\pi}\{\mathcal{I}(G, \pi)\}$$

The *treewidth* of a graph $\mathcal{G}$ is one of the most frequently used structural parameters used to examine the complexity of a problem. The definition is motivated by the desire to identify small separating sets in arbitrary graphs. We now describe the formal definition of the treewidth of $\mathcal{G}$.

Definition 3. (Treewidth) *A tree decomposition of a graph $G = (V, E)$ is a pair $(T, (B_t)_{t \in T})$, where $T = (V_T, F)$ is a tree and $(B_t)_{t \in T}$ is a family of subsets of V such that:*

1. $\cup_{t \in V_T} B_t = V$
2. *For every edge $vw \in E$, there is a $t \in V_T$ such that $v, w \in B_t$.*
3. *For every $v \in V$, the set $B^{-1}(v) := \{t \in V_T | v \in B_t\}$ is nonempty and connected in T.*

The width of the decomposition $(T, (B_t)_{t \in T})$ is the number

$$max\{|B_t| \mid t \in V_T\} - 1.$$

The treewidth $tw(G)$ of G is the minimum of the widths of the tree decompositions of G.

If in the definitions of a tree decomposition and treewidth we restrict T to be a path, then we have the definitions of path decomposition and pathwidth. We use the notation $tw(G)$ and $pw(G)$ to denote the treewidth and the pathwidth of a graph G.

For our results, we enhance the definition of a tree decomposition $(T, (B_t)_{t\in T})$, as follows: T is a tree rooted on some node r where $B_r = \emptyset$, each of its nodes have at most two children and could be one of the following

1. **Introduce node**: a node t that has only one child t' where $B_t \supset B_{t'}$ and $|B_t| = |B_{t'}| + 1$.
2. **Forget node**: a node t that has only one child t' where $B_t \subset B_{t'}$ and $|B_t| = |B_{t'}| - 1$.
3. **Join node**: a node t with two children t_1 and t_2 such that $B_t = B_{t_1} = B_{t_2}$.
4. **Base node**: a node t that is a leaf of T, is different than the root, and $B_t = \emptyset$.

Notice that, according to the above definitions, the root r of T is either a forget or a join node. It is known that any tree decomposition can be transformed to one with the above requirements while maintaining the same width (see e.g. [4, 3]). From now on, when we refer to a tree decomposition $(T, (B_t)_{t\in T})$ we presume the above requirements.

Given a tree decomposition $(T, (B_t)_{t\in T})$ and some node t of V_T, we define as $T_t = (V_t, E_t)$ the subtree of T rooted at t. Clearly, if r is the root of T, it holds that $T_r = T$. We also define $G_t = G[\cup_{s\in V_t} B_s]$.

The problem Tree-Width of deciding whether a graph has treewidth k is NP-complete. It is well known that the natural parameterization of the problem is fixed-parameter tractable.

Tree-Width *Instance:* A graph G and $k \in \mathbb{N}$. *Parameter:* k. *Problem:* Decide if $tw(G) = k$.

For the purposes of our computation, we use an approximation algorithm that is based on a connection between tree decompositions and separators, and use a well known procedure for computing small separators of a graph. We will see that this is enough to derive our main fixed-parameter tractability result.

Proposition 1 ([1]). *There is an algorithm that, given a graph $G = (V, E)$, computes a tree decomposition of G of width at most $3k + 1$ in time*

$$\mathcal{O}(4^k \cdot k^{3.5} \cdot n^2).$$

Here k denotes the treewidth of G.

In what follows, we first show that the problem of imbalance parameterized by the treewidth and the maximum degree of the input graph is fixed-parameter tractable. Combining this with a few structural observations, we obtain the fixed-parameter tractability of the problem when parameterized by the imbalance.

3 Imbalance parameterized by the treewidth and the maximum degree

Given a graph $G = (V, E)$, recall that we are interested in checking whether G has imbalance at most b. Let $(T, (B_t)_{t\in T})$ denote a tree decomposition of the graph G of width k. We consider this problem parameterized by the treewidth and the maximum degree of the graph, that is:

IMBALANCE
Instance: A graph G, a tree-decomposition $(T, (B_t)_{t \in T})$ of width k, and a non-negative integer b.
Parameter: $\Delta(G), k$
Problem: Decide if there exists a permutation $\pi : V \to V$ such that $\mathcal{I}(G, \pi) \leq b$.

A positive answer to the question can be inferred using any permutation π such that $\mathcal{I}(G, \pi)$ is at most b. If there is no such permutation, then we conclude that the answer is in the negative. Therefore, we regard a permutation π for which $\mathcal{I}(G, \pi)$ is at most b as a solution to IMBALANCE. Let $t \in V_t$, G_t be the induced subgraph corresponding to T_t and A_t be $V_t \setminus B_t$. Given a permutation π of the vertices of B_t, we define a *partial* solution for the subgraph G_t to be a permutation on the vertex set V_t which respects the ordering on the vertices of B_t given by π and that minimizes the sum of imbalance of vertices in A_t. When we arrive at the root r of the tree, it suffices to check if (one of) the solution(s) computed at this node is a permutation π such that $\mathcal{I}(G_r, \pi) \leq b$, since $V_r = V$.

We now precisely describe what we compute at every tree node $t \in V_T$. Let X denote the set $\{0, \ldots, \Delta(G)\} \cup \{\infty\}$. Given a set $U \subseteq V$ of size q, let $\mathcal{S}_U$ denote all the bijections from $[q] \to U$ and $\mathcal{G}_U$ denote the family of functions $\{g \mid g : U \to \mathrm{X}\}$. We will use g to remember the number of left neighbors of a vertex in the current tree bag. Now consider

$$\mathcal{F}(U) = \mathcal{G}_U \times \mathcal{S}_U,$$

a family of $(\Delta(G) + 2)^q q!$ functions. Each (g, π) can be thought of as a vector of length q with entries from X paired with a bijection π (which essentially gives a permutation of the vertex set U). At node t, we compute, for every element in $\mathcal{F}(B_t)$, an element of $\mathbb{N}$ – the set of natural numbers, the latter being the partial imbalance given the restrictions encoded in (g, π). We denote this value by $f_t(g, \pi)$.

We now describe how $f_t(g, \pi)$ is computed at the node $t \in V_T$.

Base Nodes. Since the base nodes contain no vertices, there is nothing to compute, and the table is empty.

Introduce Nodes. Let t be the index of the bag in which the vertex u is introduced. First check that the neighborhood conditions imposed by g are respected by the permutation π. Specifically, the permutation π should place $g(u)$ neighbors of u to its left, and its remaining neighbors to the right. If the neighborhood constraint is not respected, we let $f_t(g, \pi) = \infty$. Otherwise, we proceed as follows.

Our goal is to compute $f_t(g, \pi)$ for all $(g, \pi) \in \mathcal{F}(B_t)$. Suppose $\pi : [|B_t| = q] \to B_t$ is a permutation of B_t. Let j denote $\pi^{-1}(u)$. Consider π' obtained from π as follows:

$$\pi'(i) = \begin{cases} \pi(i) & \text{if } i < j, \\ \pi(i+1) & \text{if } j \leq i \leq q-1 \end{cases}$$

Let the index of the child node of t be t'. Now consider $g' : B_{t'} \to \mathrm{X}$ obtained from g as follows:

$$g'(v) = \begin{cases} g(v) & \text{if } v \notin N(u) \text{ or } \pi'^{-1}(v) < j, \\ g(v) - 1 & \text{if otherwise ,} \end{cases}$$

We set $f_t(g, \pi) = f_{t'}(g', \pi')$.

Forget Nodes. Let t be the node where a vertex u is forgotten and let the index of the child node of t be t'. Furthermore let $|B_t| = q$. To obtain $f_t(g, \pi)$ for all $(g, \pi) \in \mathcal{F}(B_t)$ we do the following. Consider $\pi^{(j)} : [q+1] \to B_{t'}$, $j \in [q+1]$, obtained from π as follows:

$$\pi^{(j)}(i) = \begin{cases} \pi(i) & \text{if } i < j, \\ u & \text{if } i = j, \\ \pi(i-1) & \text{if } i > j. \end{cases}$$

Now consider $g^{(s)} : B_{t'} \to \mathrm{X}$, $s \in [d(u)]$, obtained from g as follows:

$$g^{(s)}(v) = \begin{cases} g(v) & \text{if } v \neq u, \\ s & \text{if } v = u, \end{cases}$$

We are now ready to determine $f_t(g, \pi)$. We set:

$$f_t(g, \pi) = \min \Big\{ f_{t'}(g^s, \pi^j) + |d(u) - g^s(u)| \mid j \in [q+1], s \in [d(u)] \Big\}.$$

Join Nodes Let the index of the join node be t, and the indices of its children be t_1 and t_2. To determine $f_t(g, \pi)$, we do the following. Let $g_1 : B_{t_1} \to \mathrm{X}$ and $g_1 : B_{t_2} \to \mathrm{X}$. We let Q denote the collection of all pairs (g_1, g_2) such that $g_1 \in \mathcal{G}_{B_{t_1}}$, $g_2 \in \mathcal{G}_{B_{t_2}}$ and $g_1(v) + g_2(v) = g(v)$, for all $v \in B_t$. Note that $B_t = B_{t_1} = B_{t_2}$. Then set:

$$f_t(g, \pi) = \min \Big\{ f_{t_1}(g_1, \pi) + f_{t_2}(g_2, \pi) \mid (g_1, g_2) \in X \Big\}.$$

This completes the description of how the entries $f_t()$ are computed for $t \in V_T$. The correctness of the algorithm follows from the description and the discussions preceding it. After computing all the tables, we read off the entries corresponding to the imbalance in the table at the root, and check if the any of them is at most b. If it is not so then there is indeed no permutation π such that $\mathcal{I}(G, \pi)$ is bounded by b. If the answer is yes then one can construct in polynomial time a permutation π of V such that $\mathcal{I}(G, \pi) \leq b$ using standard backtracking procedures.

Note that the time taken by the algorithm is proportional to the size of the table stored at each node, time taken to fill the table entries and the total number of nodes in the tree decomposition. The number of nodes in T is known to be linear in the number of vertices of the input. Each table is of size at most $(\Delta(G) + 2)^{k+1}(k+1)!$. Each entry in the table can be filled in $(\Delta(G)^{2k+O(1)})$ time (the join node could take this much amount of time). Thus the time taken by the algorithm is, therefore, $\mathcal{O}(\Delta(G)^{O(k)} k! \cdot n)$. This leads us to the following assertion:

Lemma 1. *Given a graph $G = (V, E)$ along with a tree decomposition $(\mathcal{T}, (B_t)_{t \in T})$ of G with width at most k, we can, in time $\Delta(G)^{O(k)} k! \cdot n^{O(1)}$ check if there exists a permutation π such that $\mathcal{I}(G, \pi) \leq b$.*

4 Some Structural Observations

In this section, we establish that if the imbalance of a graph G is b, then the treewidth of the graph is at most $b/2$. More, in fact, is true: we will show that the imbalance of a graph is at least twice its *pathwidth*.

For our structural result we need an equivalent definition of pathwidth in terms of vertex separators with respect to a linear ordering of the vertices. Let G be a graph and let $\sigma = v_1 v_2 \ldots v_n$ be an ordering of V. For $j \in [n]$ put $V_j = \{v_i : \ i \in [j]\}$ and denote by ∂V_j all vertices

of V_j that have neighbors in $V \setminus V_j$. Furthermore define the set $\mathcal{E}(V_j)$ to be the set of edges with one endpoint in V_j and the other in $V \setminus V_j$. Setting $vs(G, \sigma) = \max_{i \in [n]} |\partial V_i|$, we define the *vertex separation* of G as

$$vs(G) = \min\{vs(G, \sigma) \colon \sigma \text{ is an ordering of } V(G)\}.$$

The following assertion is well-known. It follows directly from the results of Kirousis and Papadimitriou [13] on interval width of a graph, see also [12].

Proposition 2 ([12, 13]). *For any graph G, $vs(G) = pw(G)$.*

We need one more graph parameter to establish our claims in this section. To this end, we have the following definition. Given a graph $G = (V, E)$ and an ordering $\sigma = v_1 \dots v_n$ of the vertices of G, we let $cw(G, \sigma) = \max_{i \in [n]} |\mathcal{E}(V_i)|$. We define the *cutwidth* of the graph G as

$$cw(G) = \min\{cw(G, \sigma) \colon \sigma \text{ is an ordering of } V(G)\}.$$

From the definitions of vertex separation and cutwidth of the graph G together with Proposition 2 we have the following observation.

Observation 1 *For a graph G, $pw(G) \leq cw(G)$.*

The main result of this section is following.

Lemma 2. *Let $G = (V, E)$ be a graph then*

$$tw(G) \leq pw(G) \leq cw(G) \leq \frac{\mathcal{I}(G)}{2}.$$

Furthermore $\Delta(G) \leq \mathcal{I}(G)$ where $\Delta(G)$ is the maximum degree of G and $\mathcal{I}(G)$ is the value of minimum imbalance of G.

Proof. Let $\sigma = v_1 \dots v_n$ be an ordering of V. Define the rank of a vertex v, $r(v)$, to be $|N_r(v, \sigma)| - |N_l(v, \sigma)|$. Then it is easy to observe that $|\mathcal{E}(V_{j+1})| = |\mathcal{E}(V_j)| + r(v_{j+1})$. Expanding this one can show that for any $j \in [n-1]$, $|\mathcal{E}(V_{j+1})| = \sum_{i=1}^{j+1} r(v_i)$.

Let π be the permutation that minimizes the imbalance of G. Now, note that:

$$\mathcal{I}(G) = \sum_{v \in V} ||N_l(v, \pi)| - |N_r(v, \pi)||.$$

Let j be the index at which the quantity $|\mathcal{E}(V_j)|$ is the maximum (with respect to π). If there are multiple indices that witness this maximum, then let j be the smallest among them. We may rewrite the sum above as:

$$\mathcal{I}(G) = \sum_{i=1}^{j} ||N_l(\pi(i))| - |N_r(\pi(i))|| + \sum_{i=j+1}^{n} ||N_l(\pi(i))| - |N_r(\pi(i))||.$$

Now we show that each of the summands above is at least the cutwidth of the graph. Observe that:

$$\sum_{i=1}^{j} ||N_l(\pi(i))| - |N_r(\pi(i))|| \geq \sum_{i=1}^{j} r(v_i) = cw(G, \pi) \geq cw(G).$$

Let π' denote the permutation obtained by reversing π, that is, $\pi'(i) = \pi(n - i + 1)$. Notice that since j is the earliest index at which the quantity $|\mathcal{E}(V_j)|$ is the maximum with respect to

π, $n-j$ is the last index at which $|\mathcal{E}(V_j)|$ is the maximum with respect to π'. Thus, the quantity $\sum_{i=1}^{n-j} r_j = |\mathcal{E}(V_j)|$ is the cutwidth of G with respect to π'. So we have:

$$\sum_{i=j+1}^{n} ||N_l(\pi(i))| - |N_r(\pi(i))|| = \sum_{i=1}^{n-j} ||N_l(\pi'(i))| - |N_r(\pi'(i))|| \geq \sum_{i=1}^{n-j} r(v_i) = cw(G, \pi') \geq cw(G).$$

This implies that $tw(G) \leq pw(G) \leq cw(G) \leq \mathcal{I}(G)/2$.

Now we show the upper bound on the maximum degree of the graph. Note that it suffices to show that the maximum degree of the graph does not exceed $2cw(G)$. Suppose not. Then there exists a vertex v such that $d(v) > 2cw(G)$. Consider the cutwidth minimizing permutation, and note that the vertex v must have more than $cw(G)$ neighbors to either it's right or left (possibly both, but this is the case at least in one direction). The cut just after or before v, depending on whether v has more than $cw(G)$ neighbors to its left or right, is more than $cw(G)$, and this is a contradiction. This completes the proof of the lemma. □

5 Imbalance as a parameter

We now consider the following problem:

IMBALANCE
Instance: A graph G and a non-negative integer $b \in \mathbb{N}$.
Parameter: b.
Problem: Decide if there exists a permutation $\pi : V \to V$ such that $\mathcal{I}(G, \pi) \leq b$.

We use the approximation algorithm provided by Proposition 1 to check if the treewidth of the given graph is at most $1.5b + 1$. If the algorithm outputs no, we conclude, due to Lemma 2 that the imbalance of the graph is more than b. Otherwise, we obtain a tree-decomposition of width at most $1.5b + 1$. Using Lemma 1, we can now check if the imbalance of G is at most b. Now using the bounds obtained in Lemma 2 we conclude that this will require time $b^{\mathcal{O}(b)} n^{O(1)}$, which can be simplified to $2^{\mathcal{O}(b \log b)} n^{O(1)}$.

Theorem 1. *Given a graph $G = (V, E)$ and a non-negative integer b, we can, in time $2^{\mathcal{O}(b \log b)} n^{O(1)}$ check if there exists a permutation π such that $\mathcal{I}(G, \pi)$ is not more than b.*

6 Conclusion

We have described an algorithm that checks, given a graph G and a non-negative integer b, if there exists a permutation of V for which $\mathcal{I}(G, \pi)$ is at most b. Further, whenever the answer is positive, (one of) the corresponding permutation(s) can be computed. This is achieved in $2^{\mathcal{O}(b \log b)} n^{O(1)}$ time. Whether this can be improved to $2^{\mathcal{O}(b)} n^{\mathcal{O}(1)}$, remains open.

Also, since it is well-known that every problem that is FPT admits a kernel (see [14] for definition), there is the (unanswered) question of finding reduction rules for IMBALANCE. In particular, we would like to know if there is a kernel of polynomial size. It is worth noting that the techniques for answering this question in the negative are also well developed. We are not aware of evidence that may favor one conjecture over another, and hence we believe this to be an interesting problem.

References

1. E. Amir, *Efficient approximation for triangulation of minimum treewidth*, in UAI, 2001, pp. 7–15.
2. T. C. Biedl, T. M. Chan, Y. Ganjali, M. T. Hajiaghayi, and D. R. Wood, *Balanced vertex-orderings of graphs*, Discrete Applied Mathematics, 148 (2005), pp. 27–48.
3. H. L. Bodlaender, *A linear-time algorithm for finding tree-decompositions of small treewidth*, SIAM J. Comput., 25 (1996), pp. 1305–1317.
4. H. L. Bodlaender and T. Kloks, *Efficient and constructive algorithms for the pathwidth and treewidth of graphs*, J. Algorithms, 21 (1996), pp. 358–402.
5. R. G. Downey and M. R. Fellows, *Parameterized complexity*, Monographs in Computer Science, Springer-Verlag, New York, 1999.
6. J. Flum and M. Grohe, *Parameterized complexity theory*, Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag, Berlin, 2006.
7. S. Gaspers, M.-E. Messinger, R. J. Nowakowski, and P. Pralat, *Clean the graph before you draw it!*, Inf. Process. Lett., 109 (2009), pp. 463–467.
8. G. Kant, *Drawing planar graphs using the canonical ordering*, Algorithmica, 16 (1996), pp. 4–32.
9. G. Kant and X. He, *Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems*, Theor. Comput. Sci., 172 (1997), pp. 175–193.
10. J. Kára, J. Kratochvíl, and D. R. Wood, *On the complexity of the balanced vertex ordering problem*, in COCOON, vol. 3595 of Lecture Notes in Computer Science, 2005, pp. 849–858.
11. ———, *On the complexity of the balanced vertex ordering problem*, Discrete Mathematics & Theoretical Computer Science, 9 (2007).
12. N. G. Kinnersley, *The vertex separation number of a graph equals its path-width*, Inf. Process. Lett., 42 (1992), pp. 345–350.
13. L. M. Kirousis and C. H. Papadimitriou, *Interval graphs and searching*, Discrete Mathematics, 55 (1985), pp. 181–184.
14. R. Niedermeier, *Invitation to fixed-parameter algorithms*, vol. 31 of Oxford Lecture Series in Mathematics and its Applications, Oxford University Press, Oxford, 2006.
15. A. Papakostas and I. G. Tollis, *Algorithms for area-efficient orthogonal drawings*, Comput. Geom., 9 (1998), pp. 83–110.
16. D. M. Thilikos, M. J. Serna, and H. L. Bodlaender, *Cutwidth ii: Algorithms for partial w-trees of bounded degree*, J. Algorithms, 56 (2005), pp. 25–49.
17. D. R. Wood, *Optimal three-dimensional orthogonal graph drawing in the general position model*, Theor. Comput. Sci., 1-3 (2003), pp. 151–178.
18. ———, *Minimising the number of bends and volume in 3-dimensional orthogonal graph drawings with a diagonal vertex layout*, Algorithmica, 39 (2004), pp. 235–253.