Balanced Vertex-Orderings of Graphs

Therese Biedl[†] Timothy Chan[†] Yashar Ganjali[‡]
MohammadTaghi Hajiaghayi[§] David R. Wood[¶]

May 24, 2002

Abstract

We consider the problem of determining a balanced ordering of the vertices of a graph; that is, the neighbors of each vertex v are as evenly distributed to the left and right of v as possible. This problem, which has applications in graph drawing for example, is shown to be NP-hard, and remains NP-hard for bipartite simple graphs with maximum degree six. We then describe and analyze a number of methods for determining a balanced vertex-ordering, obtaining optimal orderings for directed acyclic graphs and graphs with maximum degree three. Finally we consider the problem of determining a balanced vertex-ordering of a bipartite graph with a fixed ordering of one bipartition. When only the imbalances of the fixed vertices count, this problem is shown to be NP-hard. On the other hand, we describe an optimal linear time algorithm when the final imbalances of all vertices count. We obtain a linear time algorithm to compute an optimal vertex-ordering of a bipartite graph with one bipartition of constant size.

keywords: graph algorithm, graph drawing, vertex-ordering, balanced.

1 Introduction

A number of algorithms for graph drawing use a 'balanced' ordering of the vertices of the graph as a starting point [18, 19, 25, 30, 32, 33]. Here balanced means that the neighbors of each vertex v are as evenly distributed to the left and right of v as possible. We consider the problem of determining such a vertex-ordering.

Throughout this paper $G = (V, E)$ is a connected graph without loops which may be directed or undirected. We assume G is simple unless explicitly called a

[†]Department of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada. Research supported by NSERC. E-mail: {biedl,tmchan}@uwaterloo.ca

[‡]Department of Electrical Engineering, Stanford University, Stanford, CA 94305, U.S.A. Completed while a graduate student in the Department of Computer Science at the University of Waterloo. E-mail: yganjali@Stanford.edu

[§]Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A. Completed while a graduate student in the Department of Computer Science at the University of Waterloo. E-mail: hajiagha@theory.lcs.mit.edu

[¶]School of Computer Science, Carleton University, Ottawa, ON K1S 5B6, Canada. Research supported by NSERC. Partially completed while at Monash University, McGill University and The University of Sydney (supported by the ARC). E-mail: davidw@scs.carleton.ca

multigraph. The number of vertices of G is denoted by $n = |V|$ and the number of edges of G is denoted by $m = |E|$. vw refers to the undirected edge $\{v, w\} \in E$ if G is undirected, and to the directed edge $(v, w) \in E$ if G is directed. We denote by $E(v)$ the set of (outgoing) edges $\{vw \in E\}$ incident to a vertex v. The *degree* of v is $\deg(v) = |E(v)|$.

A *vertex-ordering* π of G is a total ordering on V or equivalently a numbering $(v_1, v_2, \ldots, v_n)$ of V. Each edge $v_iv_j \in E(v_i)$ with $i < j$ is a *successor edge* of v_i, and v_j is a *successor* of v_i. Similarly each edge $v_iv_j \in E(v)$ with $j < i$ is a *predecessor edge* of v_i, and v_j is a *predecessor* of v_i. The number of predecessor and successor edges of a vertex v_i is denoted by $\text{pred}_\pi(v_i)$ and $\text{succ}_\pi(v_i)$, respectively. That is, $\text{pred}_\pi(v_i) = |\{v_iv_j \in E(v) : j < i\}|$ and $\text{succ}_\pi(v_i) = |\{v_iv_j \in E(v) : i < j\}|$. We omit the subscript π if the ordering in question is clear. Note that for directed graphs, we only count the number of outgoing edges incident to a vertex v_i in $\text{pred}(v_i)$ and $\text{succ}(v_i)$. In a given vertex-ordering, a vertex v is called a

$$(\min\{\text{pred}(v), \text{succ}(v)\}, \max\{\text{pred}(v), \text{succ}(v)\})\text{-vertex},$$

and the *imbalance* of v is defined to be $\phi(v) = |\text{succ}(v) - \text{pred}(v)|$. We say v is *balanced* if $\phi(v)$ is minimum, taken over all partitions of the edges incident to v into predecessor and successor edges. A vertex has even imbalance if and only if it has even degree; hence the imbalance of a vertex with odd degree is at least one. In a vertex-ordering of a simple graph, a vertex v is balanced if and only if $\phi(v) \leq 1$.

The *total imbalance* of a vertex-ordering is the sum of the imbalance of each vertex. We say a vertex-ordering is *perfectly balanced* if every vertex is balanced. Thus a vertex-ordering of a simple graph is perfectly balanced if and only if the total imbalance is equal to the number of odd degree vertices. We are interested in the following problem.

BALANCED VERTEX-ORDERING
Instance : A (directed) graph $G = (V, E)$, integer $K \geq 0$.
Question : Does G have a vertex-ordering with total imbalance $\sum_{v \in V} \phi(v) \leq K$?

For a given graph, a vertex-ordering with minimum total imbalance is said to be *optimal*. Note that $\phi(v) = 2\max\{\text{succ}(v), \text{pred}(v)\} - \deg(v)$. Hence the problem of finding an optimal vertex-ordering is equivalent to finding a vertex-ordering which minimizes

$$\sum_{v \in V} \max\{\text{pred}(v), \text{succ}(v)\} \quad . \tag{1}$$

However, for approximation-purposes, the balanced vertex-ordering problem and minimizing (1) are not equivalent. Since $\frac{1}{2}\deg(v) \leq \max\{\text{succ}(v), \text{pred}(v)\} \leq \deg(v)$, an arbitrary vertex-ordering will be a 2-approximation for the problem of minimizing (1).

There is another equivalent formulation of the balanced vertex-ordering problem, which shall prove useful to consider. In a particular vertex-ordering, let $\phi'(v) = 2\lfloor\frac{1}{2}|\text{succ}(v) - \text{pred}(v)|\rfloor$. Here, $\phi'(v)$ may be zero for both even and odd

degree vertices v. Since

$$\sum_v \phi(v) \;=\; |\,\{v : \deg(v) \text{ is odd}\}\,| + \sum_v \phi'(v) \;\;,$$

a vertex-ordering is optimal if and only if it minimizes $\sum_v \phi'(v)$.

In a vertex-ordering of an undirected graph $G = (V, E)$, the total imbalance is equal to the total imbalance of the same vertex-ordering of the symmetric directed graph $(V, \{(v, w), (w, v) : vw \in E\})$. Hence the balanced ordering problem for directed graphs is a generalization of the same problem for undirected graphs.

In related work, Wood [30, 33] takes a local minimum approach to the balanced vertex-ordering problem. The algorithms here apply simple rules to move vertices within an existing ordering to reduce the total imbalance. Certain structural properties of the produced vertex-orderings are obtained, which are used in an algorithm for graph drawing.

In this paper we present the following results. In Section 2 we show, using a reduction from NAE-3SAT, that the balanced vertex-ordering problem is NP-complete. In particular, we prove that determining whether a given graph has a perfectly balanced vertex-ordering is NP-complete, and remains NP-complete for bipartite graphs with maximum degree six.

Section 3 explores the relationship between balanced vertex-orderings and the connectivity of undirected graphs. We describe an algorithm for determining a vertex-ordering with the minimum number of highly unbalanced vertices; that is, vertices v with $\text{pred}(v) = 0$ or $\text{succ}(v) = 0$. The same algorithm determines optimal vertex-orderings of undirected graphs with maximum degree three.

Section 4 describes and analyses a heuristic approach for determining a balanced vertex-ordering of an arbitrary graph. This algorithm has been successfully used in [3, 31] to establish improved bounds for the area of orthogonal graph drawings. We analyze the performance of this algorithm, establishing a worst-case upper bound on the total imbalance which is tight in the case of the complete graph. Furthermore, the method determines perfectly balanced vertex-orderings of directed acyclic graphs.

In Section 5 we consider the problem of determining a balanced vertex-ordering of a bipartite graph where a fixed vertex-ordering of one bipartition is given. The problem where only the imbalance of the fixed vertices in the ordering counts, is shown to be NP-complete. On the other hand, we present linear time algorithms for the problems where only the final imbalance of the unsettled vertices counts, and where the final imbalance of all vertices count. A corollary of this final result is that the balanced ordering problem is solvable in linear time if the number of vertices in one bipartition is constant.

2 Complexity

In this section we show that the balanced vertex-ordering problem is NP-complete. Our reduction is from the Not-All-Equal-3SAT problem (NAE-3SAT for short). Here we are given a set U of boolean variables and a collection C of clauses over U such that each clause $c \in C$ has $2 \leq |c| \leq 3$. The problem is to determine whether

there is a truth assignment for U such that each clause in C has at least one true literal and at least one false literal. In a given instance of NAE-3SAT, the number of times a variable x appears is called the *order* of x, and is denoted by d_x. NAE-3SAT is NP-complete [27], and it is well-known (see for example [21]) that NAE-3SAT remains NP-complete if all literals are positive and/or every variable x has $d_x \leq 3$.

Theorem 1. *Determining if a given graph has a perfectly balanced vertex-ordering is NP-complete, and remains NP-complete for bipartite undirected graphs with maximum degree six.*

Proof. Let I be an instance of NAE-3SAT such that all literals are positive and every variable x has $d_x \leq 3$. We now convert I to an instance of the balanced vertex-ordering problem. Construct a graph G as follows. For each variable $x \in U$ add the gadget shown in Fig. 1 to G. In particular, add the vertices $x_0, x_1, \ldots, x_{2d_x}$ to G. We call x_0 the *variable vertex* associated with the variable x. Now add edges $x_j x_{j+1}$, $1 \leq j \leq 2d_x - 1$, to G, along with the edges $x_0 x_{2j-1}$, $1 \leq j \leq d_x$. In addition, add a *clause vertex* c_0 to G for each clause $c \in C$, and insert an edge $x_0 c_0$ for each variable x appearing in c.

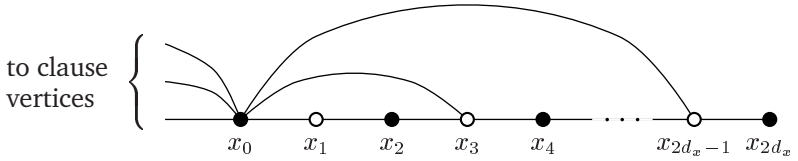


Figure 1: The gadget associated with a variable x.

We claim that the instance of NAE-3SAT is satisfiable if and only if G has a perfectly balanced vertex-ordering. To prove the only-if direction construct a vertex-ordering of G with all the clause vertices in the middle of the vertex-ordering in arbitrary order, and for each variable x, put $x_0, x_1, \ldots, x_{2d_x}$ to the left (respectively, right) of the clause vertices if x is true (false). For the true variables x, order the vertices $x_{2d_x}, x_{2d_x-1}, \ldots, x_0$ from left to right, and for the false variables x, order the vertices $x_0, x_1, \ldots, x_{2d_x}$ from left to right. For each variable $x \in U$, the vertex x_0 has d_x predecessor edges and d_x successor edges (going to clause vertices and to $\{x_{2j-1}, 1 \leq j \leq d_x\}$). Thus x_0 is balanced. The vertices x_j, $1 \leq j \leq 2d_x$, are either $(1,1)$, $(1,2)$ or $(0,1)$, and are thus balanced. Since every clause $c \in C$ contains at least one true literal and at least one false literal, the vertex c_0 has at least one successor and at least one predecessor. Since $\deg(c_0) \leq 3$, c_0 is balanced. Hence every vertex is balanced, and thus the vertex-ordering is perfectly balanced.

For the if direction, assume we have a perfectly balanced vertex-ordering, and consider the vertex x_0 for some variable x.

Case 1. x_1 is to the right of x_0: As x_1 has degree two, x_2 must be to the right of x_1. Similarly, as x_2 has degree two, x_3 must be to the right of x_2. As x_3 has degree three, and already has two predecessors x_0 and x_2, its third neighbor x_4 must be to the right of x_3. By induction, all of $x_1, x_2, \ldots, x_{2d_x}$ must be to the right of x_0. Thus x_0 is to the left of its neighbors $x_1, x_3, \ldots, x_{2d_x-1}$. Since x_0 is balanced, it must be to the right of its remaining d_x neighbors, which are the clause vertices of the clauses containing x. Set the variable x to false.

Case 2. x_1 is to the left of x_0: Then symmetrically, x_0 is to the left of its d_x adjacent clause vertices. Set x to true.

A clause vertex c_0 has degree two or three. Hence c_0 has at least one predecessor and at least one successor, and thus c contains at least one false variable and at least one true variable; that is, c is satisfied.

We have shown that the given instance of NAE-3SAT is satisfied if and only if the graph G has a perfectly balanced vertex-ordering. G is simple and bipartite (with the vertices partitioned into the sets $\{c_0 : c \in C\} \cup \{x_{2j-1} : x \in U, 1 \leq j \leq d_x\}$ and $\{x_{2j} : x \in U, 0 \leq j \leq d_x\}$). Observe that the maximum degree of G is twice the maximum order which is at most three. Thus the maximum degree of G is at most six. It is trivial to check if a given vertex-ordering is perfectly balanced. Since NAE-3SAT is NP-complete [27], and the construction of G is polynomial, testing if a graph has a perfectly balanced vertex-ordering is NP-complete for simple bipartite graphs with maximum degree six. □

For an intended application in 3-D orthogonal graph drawing [32] it is important to consider balanced vertex-orderings of graphs with minimum degree five and maximum degree six. We now show that we still have NP-completeness in this case, at least for multigraphs.

Lemma 1. *Determining if a bipartite undirected multigraph with minimum degree five and maximum degree six has a perfectly balanced vertex-ordering is NP-complete.*

Proof. Let I be an instance of NAE-3SAT containing only positive literals. For each clause c of I, if $c = x \vee y \vee z$ then set $c = x \vee x \vee y \vee y \vee z \vee z$, and if $c = x \vee y$ then set $c = x \vee x \vee x \vee y \vee y \vee y$. Thus each clause now has exactly six literals. This does not affect whether there is a solution to I.

For each variable x with $d_x \geq 4$, introduce two new variables y and z, called *replacement* and *special* variables, respectively. Replace two occurrences of x by y, and add new *special* clauses $x \vee z$ and $y \vee z$. Thus d_x decreases by one, and in any not-all-equal truth assignment $x = y$; that is, this operation does not affect whether I is satisfiable. Repeat the above step until each variable has order two or three. Since this operation can be applied at most $3m$ times, where m is the number of clauses, the size of the instance is still polynomial. All clauses now contain two or six variables. Now construct a graph G similar to that in Theorem 1, but using the gadget shown in Fig. 2.

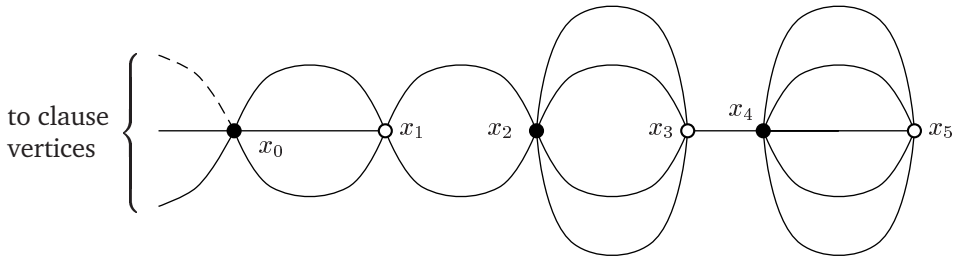


Figure 2: The gadget associated with a variable x.

Since each clause has two or six literals, each clause vertex has degree two or six in G. If a clause vertex has degree two in G; that is, it corresponds to a special clause, then simply replace it by an edge between its two neighbors. This does not affect whether the graph has a perfectly balanced ordering. Thus all clause vertices now have degree six. A variable vertex x_0 has degree five if $d_x = 2$, and degree six if $d_x = 3$. A vertex x_i, $1 \leq i \leq 5$, has degree five or six. Thus the graph has minimum degree five and maximum degree six. Furthermore the graph is bipartite with the following 2-coloring. For each original variable or replacement variable, color the gadget as shown in Fig. 2. For each special variable, color the gadget in the opposite way to Fig. 2. Special variables were only in special clauses, and since the corresponding special clause vertices have been replaced by an edge, the only neighbors of a special variable vertex are original or replacement variable vertices (and of course the vertices within the gadget). Thus the graph is bipartite.

We now show that the same argument in Theorem 1 holds for this graph. A clause vertex c_0 is perfectly balanced if and only if c_0 is a (2,4)-vertex or a (3,3)-vertex if and only if c contains at least one true literal and at least one false literal. A variable vertex is perfectly balanced if and only if it is a (2,3)-vertex or a (3,3)-vertex, and thus must appear completely to the right or left of the vertices corresponding to the clauses containing it. Clearly, any arrangement of the vertices within a gadget other that shown in Fig. 2 will increase the imbalance (except for the reverse order). It follows that by the same argument in Theorem 1, this graph has a perfectly balanced ordering if and only if the instance of NAE-3SAT is satisfiable. □

A strategy for producing 3-D orthogonal point-drawings of maximum degree six graphs which is employed by Eades *et al.* [15] and Wood [32], is to position the vertices along the main diagonal of a cube. For graphs with minimum degree five, minimizing the number of bends in such a drawing is equivalent to finding an optimal ordering of the vertices along the diagonal; see [32]. As a consequence of Lemma 1 we therefore have the following result.

Theorem 2. *Let G be bipartite undirected multigraph with maximum degree six. It is NP-hard to find a 3-D orthogonal point-drawing of G with a diagonal vertex layout, and with the minimum number of bends.* □

3 Connectivity and Maximum Degree

We now examine relationships between balanced vertex-orderings and the vertex-connectivity of a graph.

3.1 st-Orderings

A vertex-ordering $(v_1, v_2, \ldots, v_n)$ of an undirected graph $G = (V, E)$ is an st-ordering if $v_1 = s$, $v_n = t$, and for every other vertex v_i, $1 < i < n$, with $\deg(v_i) \geq 2$, we have $\text{pred}(v_i) \geq 1$ and $\text{succ}(v_i) \geq 1$. Lempel *et al.* [22] show that for any biconnected graph $G = (V, E)$ and for any $s, t \in V$, there exists an st-ordering of G. Cheriyan and Reif [6] extended this result to directed graphs.

Even and Tarjan [17] develop a linear time algorithm to compute an st-ordering of an undirected biconnected graph (also see [16, 24, 29]). Under the guise of *bipolar orientations*, st-orderings have also been studied in [7, 10, 26]. In related work, Papakostas and Tollis [25] describe an algorithm for producing so-called bst-orderings of graphs with maximum degree four; these are st-orderings with a lower bound on the number of perfectly balanced vertices of degree four.

We now give an example of a 2-connected graph for which every st-ordering is not optimal. Consider the graph G_k ($k \geq 3$) obtained by replacing each vertex of a k-cycle by the graph H shown in Fig. 3, and connecting the black vertices in adjacent copies by an edge. An st-ordering of G_k starts with one copy of H, then the next copy of H, and so on. Consider the induced vertex-ordering of a copy of H which is not the first or last copy in the ordering. The first and last vertices in the ordering of H must be the black vertices, as otherwise there would be a vertex v with $\text{succ}(v) = 0$ or $\text{pred}(v) = 0$. Any permutation of the white vertices of H between the two black vertices gives an st-ordering. Thus, in an st-ordering of G_k, not counting the vertices in the first and last copies of H, the imbalance of each white vertex is one, and the imbalance of each black vertex is eight. Thus the total imbalance is at least $24(k-2)$.

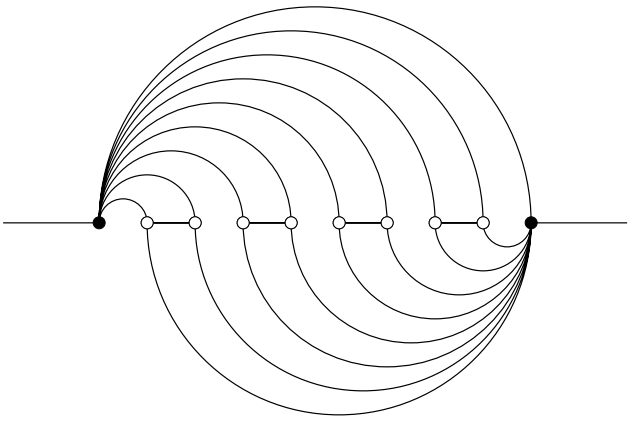


Figure 3: An st-ordering of H.

As shown in Fig. 4, the vertex-ordering of H with the black vertices in the middle gives a vertex-ordering of G_k which is not an st-ordering. Half the white vertices have an imbalance of three, and the other half have an imbalance of one. Black vertices are balanced, except for the first and last black vertices which have an imbalance of two. Hence the total imbalance is $16k + 4$. This example says that st-orderings are not always a means for finding optimal orderings. Nevertheless they immediately give the following upper bound on the total imbalance.

Lemma 2. *The total imbalance in an st-ordering of an n-vertex m-edge graph $G = (V, E)$ is at most $2m - 2n + 4$ if G is undirected and $m - 2n + 4$ if G is directed.* □

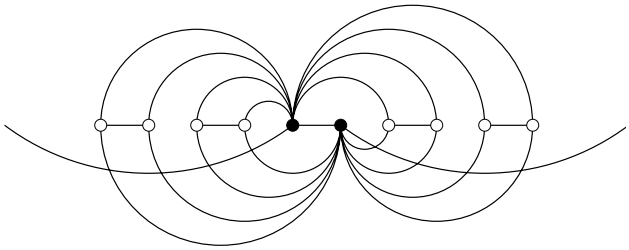


Figure 4: A more balanced vertex-ordering of H.

3.2 Combining st-Orderings

The following algorithm determines a vertex-ordering of a graph based on st-orderings of its biconnected components (*blocks*). On one hand, in Section 2 we proved that given an optimal vertex-ordering of each biconnected component, it is NP-hard to find an optimal vertex-ordering of the graph. However, this algorithm and variations of it have proved useful in many graph drawing algorithms [2, 23, 28] as it gives bounds on the number of highly unbalanced vertices (see Lemma 3 below). Moreover, we employ this method to obtain optimal vertex-orderings of graphs with maximum degree three.

It is well-known that the blocks of a graph can be stored in the form of a tree; this is the so-called *block-cut-tree*, which we denote by $\mathcal{BC}(G)$ for a graph G. A block containing exactly one cut-vertex is called an *end-block*.

COMBINE st-ORDERINGS
Input: undirected graph $G = (V, E)$
Output: vertex-ordering of G

Let B_1 be an end-block of G.
Complete a depth-first traversal of $\mathcal{BC}(G)$ starting at B_1, and
    let $B_1, B_2, \ldots, B_r$ be the depth-first numbering of the blocks of G.
Let t_1 be a cut-vertex of B_1, and let s_1 be a vertex of B_1 distinct from t_1.
Initialize the *current ordering* to be an s_1t_1-ordering of B_1.
for $i = 2, 3, \ldots, r$ **do**
    Let s_i be the (unique) cut-vertex of B_i with some block B_j with $j < i$.
    if B_i is an end-block of G **then**
        Let t_i be a vertex of B_i distinct from s_i.
    else
        Let t_i be a cut-vertex of B_i with some block B_j with $j > i$.
    end-if
    Let $(v_1^i, v_2^i, \ldots, v_{n_i}^i)$ be an s_it_i-ordering of B_i (with $v_1^i = s_i$ and $v_{n_i}^i = t_i$).
    Append $(v_2^i, v_3^i, \ldots, v_{n_i}^i)$ to the current ordering.
end-for

Lemma 3. *Let G be an undirected graph with k end-blocks, and assume $k \geq 2$; that is, G has at least one cutvertex. Then* COMBINE st-ORDERINGS *algorithm determines*

a vertex ordering in linear time, with one vertex v having $\text{pred}(v) = 0$*, and* $k - 1$ *vertices* v *having* $\text{succ}(v) = 0$*.*

Proof. By the definition of st-ordering, a vertex $v \in V$ which is not s_i or t_i for some i, has $\text{pred}(v) > 0$ and $\text{succ}(v) > 0$. We now count the number of vertices with zero successors. A vertex s_i has $\text{succ}(s_i) > 0$. A vertex t_i for which B_i is not an end-block has $\text{succ}(t_i) > 0$. The vertex t_1, for which B_1 is an end-block, has $\text{succ}(t_i) > 0$. The remaining vertices t_i with B_i an end-block have $\text{succ}(t_i) = 0$. Hence the number of vertices v having $\text{succ}(v) = 0$ is $k - 1$. We now count the number of vertices with zero predecessors. A vertex t_i has $\text{pred}(t_i) > 0$. For each $i \geq 2$, s_i is chosen to be the cut-vertex with some block B_j ($j < i$) — such a block must exist because of the depth-first numbering of the blocks. Hence s_i has predecessors in B_j, and therefore the only vertex with zero predecessors is s_1. Since the block-cut-tree and the st-orderings can be determined in linear time, and since the block-cut-tree has linear size, the algorithm runs in linear time. □

The next result easily follows from Lemma 3.

Lemma 4. *Given a non-biconnected n-vertex m-edge undirected graph with k end-blocks, the* COMBINE st-ORDERINGS *algorithm determines in linear time a vertex ordering with total imbalance at most* $2m - 2n + 2k$*.* □

We now show that the COMBINE st-ORDERINGS algorithm determines a vertex-ordering with the minimum number of vertices with zero predecessors or zero successors. Consider an end-block B. Then either the first vertex of B in the ordering has no predecessors, or the last vertex of B in the ordering has no successors, for in an end-block B only one vertex has neighbors outside of B. The next result follows.

Lemma 5. *Every vertex-ordering of an undirected graph with k end-blocks has at least k vertices v having* $\text{pred}(v) = 0$ *or* $\text{succ}(v) = 0$*.*

Note that for a triangulated planar graph G, vertex-orderings can be determined which are more balanced than st-orderings. de Fraysseix *et al.* [11] show that G has a *canonical* vertex-ordering $(v_1, v_2, \ldots, v_n)$ with $\text{pred}(v_i) \geq 2$ for every vertex v_i, $3 \leq i \leq n$, and with $\text{succ}(v_i) \geq 1$ for every vertex v_i, $1 \leq i \leq n - 1$. Kant [18] generalizes canonical orderings to the case of 3-connected planar graphs, and it is easy to extend canonical orderings to 3-connected non-planar graphs (Kant, private communication, 1992; see also [9]). Kant and He [19] show that if G is 4-connected then G has a vertex-ordering with every vertex v_i, $3 \leq i \leq n - 2$, having $\text{succ}(v_i) \geq 2$ and $\text{pred}(v_i) \geq 2$. The next result follows.

Lemma 6. *An n-vertex m-edge 4-connected triangulated planar undirected graph has a vertex-ordering with total imbalance at most* $2m - 4n + 12$*.* □

3.3 Graphs with Maximum Degree Three

We now apply the results from the previous section to obtain optimal vertex-orderings of graphs with maximum degree three.

Lemma 7. *An st-ordering of biconnected undirected graph G with maximum degree at most 3 is optimal.*

Proof. Supppose G has n vertices. Clearly the result holds if $n = 2$. Assume from now on that $n \geq 3$. In this case, all vertices have degree at least two by biconnectivity and at most three by assumption. Let n_3 be the number of degree three vertices in G. In an st-ordering,

$$\sum_v \phi(v) = \begin{cases} 2 + 2 + n_3 = n_3 + 4 & \text{, if } \deg(s) = \deg(t) = 2 \\ 3 + 3 + (n_3 - 2) = n_3 + 4 & \text{, if } \deg(s) = \deg(t) = 3 \\ 2 + 3 + (n_3 - 1) = n_3 + 4 & \text{, if } \{\deg(s), \deg(t)\} = \{2, 3\} \end{cases} .$$

By considering the degrees of the first and last vertex, and since every degree three vertex v has $\phi(v) \geq 1$, it is easily seen that any vertex-ordering of G has total imbalance at least $n_3 + 4$. □

Theorem 3. *Given an undirected graph $G = (V, E)$ with maximum degree at most three, the* COMBINE st-ORDERINGS *algorithm determines in linear time an optimal vertex-ordering of G.*

Proof. As noted in Section 1, finding an optimal vertex-ordering is equivalent to minimizing $\sum_v \phi'(v)$, where $\phi'(v) = 2\lfloor \frac{1}{2} |\text{succ}(v) - \text{pred}(v)| \rfloor$. For graphs with maximum degree three, $\phi'(v) = 2$ if v is a $(0, 2)$- or $(0, 3)$-vertex, and $\phi'(v) = 0$ otherwise. Hence minimizing $\sum_v \phi'(v)$ is equivalent to minimizing the number of $(0, 2)$- and $(0, 3)$-vertices. Every vertex with degree one must have zero predecessors or zero successors, thus minimizing the number of $(0, 2)$- and $(0, 3)$-vertices is equivalent to minimizing the number of vertices with zero predecessors or zero successors. By Lemma 3 and Lemma 5, the COMBINE st-ORDERINGS algorithm determines in linear time, a vertex-ordering with the minimum possible number of vertices with zero predecessors or zero successors. Therefore the COMBINE st-ORDERINGS algorithm determines an optimal vertex-ordering for graphs with maximum degree three. □

Observe that in the reduction in Theorem 1, the variable vertices are cut-vertices, and that each biconnected component has maximum degree three. By Theorem 3, an optimal ordering of a graph with maximum degree three can be determined in linear time. Hence, we have the following result.

Corollary 1. *Finding the optimal vertex-ordering of a graph is NP-hard, even if given an optimal vertex-ordering of each biconnected component.* □

4 Median Placement Heuristic

We now describe a heuristic for the balanced vertex-ordering problem which provides a tight upper bound for the total imbalance of the vertex-orderings produced. The algorithm inserts each vertex, in turn, mid-way between its already inserted neighbors. At any stage of the algorithm we refer to the ordering under construction as the *current ordering*. Similar methods were introduced by Biedl and Kaufmann [3] and Biedl, Madden, and Tollis [4].

MEDIAN PLACEMENT
Input: vertex-ordering $I = (u_1, u_2, \ldots, u_n)$ of a (directed) graph G
(called the *insertion ordering*)
Output: vertex-ordering of G

for $i = 1, 2, \ldots, n$ **do**
Let $w_1, w_2, \ldots, w_k$ be the predecessors of u_i in the insertion ordering,
ordered by their position in the current ordering.
if $k = 0$ **then** Insert u_i arbitrarily into the current ordering.
else if k is even **then** Insert u_i arbitrarily between $w_{k/2}$ and $w_{k/2+1}$.
else (k is odd) Insert u_i immediately before or after $w_{(k+1)/2}$
to minimize the imbalance of $w_{(k+1)/2}$.
(In this case $w_{(k+1)/2}$ is called the *median neighbor* of u_i.)
end-for

Using the median-finding algorithm of Blum *et al.* [5], and the algorithm of Dietz and Sleator [12] to maintain the vertex-ordering and orderings of the adjacency lists of G, the algorithm can be implemented in linear time.

For a given insertion ordering I of a (directed) graph $G = (V, E)$, let X be the set of vertices $u \in V$ for which $\text{pred}_I(u)$ is odd.

Lemma 8. *The algorithm* MEDIAN PLACEMENT *determines in linear time a vertex-ordering of a (directed) graph $G = (V, E)$ with total imbalance*

$$\sum_{v \in V} \phi(v) \;\leq\; |X| + \sum_{u \in V} \text{succ}_I(u) \;\; .$$

Proof. When a vertex u is inserted into the current ordering, the predecessors of u in I are precisely the neighbors of u which have already been inserted into I. Thus immediately after u is inserted, $\phi(u) = 0$ if $\text{pred}_I(u)$ is even and $\phi(u) = 1$ if $\text{pred}_I(u)$ is odd. Even if all the successors of u (in the insertion ordering) are inserted on the one side of u, in the final ordering, the imbalance $\phi(u) \leq \text{succ}_I(u)$ if $\text{pred}_I(u)$ is even, and $\phi(u) \leq \text{succ}_I(u) + 1$ if $\text{pred}_I(u)$ is odd. Thus the total imbalance is at most $|X| + \sum_u \text{succ}_I(u)$. □

4.1 Undirected Graphs

Theorem 4. *The algorithm* MEDIAN PLACEMENT *determines in linear time a vertex-ordering of an n-vertex m-edge undirected graph with total imbalance*

$$\sum_{v} \phi(v) \;\leq\; m + \min\{|X|, n - |X|\} \;\leq\; m + \left\lfloor \frac{n}{2} \right\rfloor \;\; .$$

Proof. That $\sum_v \phi(v) \leq m + |X|$ follows immediately from Lemma 8 since $\sum_u \text{succ}_I(u) = m$ for undirected graphs. For each vertex $v \in V$, let $X(v)$ be the set of vertices $u \in X$ such that v is the median neighbor of u when u is inserted into the current ordering. Thus elements of $X(v)$ are successors of v in I, and $\sum_v |X(v)| = |X|$.

Since vertices in $X(v)$ are inserted to balance v, $\phi(v) \leq \text{succ}_I(v) - |X(v)|$ if $\text{pred}_I(v) + |X(v)|$ is even, and $\phi(v) \leq 1 + \text{succ}_I(v) - |X(v)|$ if $\text{pred}_I(v) + |X(v)|$ is odd. Thus

$$\sum_v \phi(v) \;\leq\; n + \sum_v \text{succ}_I(v) - \sum_v |X(v)| \;=\; n + m - |X| \;. \qquad \square$$

A simple calculation shows that any vertex-ordering of the complete graph K_n has total imbalance $\lfloor \frac{n^2}{2} \rfloor = m + \lfloor \frac{n}{2} \rfloor$. Thus Theorem 4 provides an upper bound on the total imbalance which is tight in this case. Comparing the bound on the total imbalance established by the MEDIAN PLACEMENT algorithm (Theorem 4) versus the analogous bound for the imbalance of st-orderings of biconnected graphs (Lemma 2), the MEDIAN PLACEMENT algorithm is better for simple graphs with average degree at least five. One the other hand, for simple 4-connected triangulated planar graphs (which have average degree just under six), the bound in Lemma 6 is better than that in Theorem 4.

We now prove that the vertex-orderings produced by the MEDIAN PLACEMENT algorithm are in some sense locally optimal.

Lemma 9. *For undirected graphs, assuming the existing vertex-ordering is fixed, each iteration of the* MEDIAN PLACEMENT *algorithm inserts the vertex u to minimize the increase in the total imbalance.*

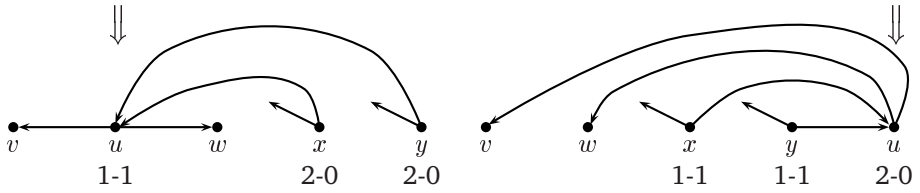
Proof. When inserting a vertex u, only the imbalance of u and its neighbors may change. Thus we need only consider the positions in the current ordering between the neighbors of u as potential places for the insertion of u. If k is even the position in the current ordering between $w_{k/2}$ and $w_{k/2+1}$ is called the *median position*. If k is odd there are two *median positions*: immediately before and after $w_{(k+1)/2}$. Assume that there exists a position to insert u in the current vertex-ordering, which is not a median position, but minimizes the total imbalance of the current ordering.

Suppose k is even. If moving u to the median position involves moving u past t neighbors of u, then doing so decreases $\phi(u)$ by $2t$, while the imbalance of each of these t neighbors increases by at most 2. Thus moving u to the median position does not increase the total imbalance.

Suppose k is odd. If moving u to the closer median position involves moving u past t neighbors of u, then doing so decreases $\phi(u)$ by $2t$, while the imbalance of each of these t neighbors increases by at most 2. Thus moving u to the closer median position does not increase the total imbalance. The imbalance of u is the same in either median position, and only the imbalance of $w_{(k+1)/2}$ differs with u in the different median positions. Thus by inserting u in a median position which minimize $\phi(w_{(k+1)/2})$, we minimize the total imbalance. $\square$

4.2 Directed Graphs

We now analyze the MEDIAN PLACEMENT algorithm in the general case of directed graphs. Firstly observe that Lemma 9 does not hold for directed graphs as the example in Fig. 5 shows. Using the MEDIAN PLACEMENT algorithm the total imbalance becomes four, whereas there exists a position, illustrated in Fig. 5(b), to insert u with total imbalance two.



(a) median placement insertion (b) minimum imbalance insertion

Figure 5: Inserting vertex u into a vertex-ordering of a directed graph.

Lemma 8 suggests that a good insertion ordering for the MEDIAN PLACEMENT algorithm applied to a directed graph, is one with small $\sum_u \text{succ}(u)$. For any vertex-ordering of a directed graph, $\sum_u \text{succ}(u)$ or $\sum_u \text{pred}(u)$ is at most $\frac{m}{2}$. Thus using an arbitrary vertex-ordering or its reverse as the insertion ordering in the MEDIAN PLACEMENT algorithm we obtain a vertex-ordering with total imbalance at most $\frac{m}{2} + n$. For acyclic graphs, a reverse topological ordering has $\text{succ}(u) = 0$ for all vertices u. Since such an ordering can be determined in linear time (see [8] for example) we have the following result (which was implicitly used by Biedl and Kaufmann [3, Theorem 4] to establish upper bounds on the area of orthogonal graph drawings.)

Theorem 5. *A perfectly balanced vertex-ordering of a directed acyclic graph can be determined in linear time* (*with total imbalance* $|X|$). □

For a directed graph $G = (V, E)$ which is not necessarily acyclic, a good insertion ordering can be obtained by first removing edges to make G acyclic. A *feedback arc set* of G is a set of edges $F \subseteq E$ such that $G \setminus F$ is acyclic. Since the successor edges in a vertex-ordering form a feedback arc set, and a reverse topological ordering of the graph obtained by removing a feedback arc set F has $\sum_u \text{succ}(u) = |F|$, finding a vertex-ordering with minimum $\sum_u \text{succ}(u)$ is equivalent to finding a minimum feedback arc set, which is NP-hard [20].

Berger and Shor [1] establish an asymptotically tight bound for the size of a feedback arc set. They show that, for directed graphs of maximum degree Δ and without 2-cycles, the minimum of $\sum_u \text{succ}(u)$ (taken over all vertex-orderings) is $\frac{m}{2} - \Theta(m/\sqrt{\Delta})$, and a vertex-ordering with $\sum_u \text{succ}(u) = \frac{m}{2} - \Theta(m/\sqrt{\Delta})$ can be determined in $O(mn)$ time. Using this as the insertion ordering in algorithm MEDIAN PLACEMENT, by Lemma 8 with $|X| \leq n$, we obtain the following result.

Theorem 6. *Every n-vertex m-edge directed graph without 2-cycles has a vertex-ordering, which can be computed in $O(mn)$ time, with total imbalance*

$$\sum_v \phi(v) \;\leq\; n + \frac{m}{2} - \Theta\left(\frac{m}{\sqrt{\Delta}}\right) \quad . \qquad \square$$

Only for small values of Δ is the constant in the $\Theta(m/\sqrt{\Delta})$ term evaluated; thus for graph drawing purposes only the $n + m/2$ term can be used. This bound can be improved by using a result of Eades *et al.* [14]. They give a linear time greedy heuristic for finding a feedback arc set, and prove an exact bound on $\sum_u \text{succ}(u)$,

which in a number of instances, provides a better result than that in [1]. In particular, they show that every directed graph without 2-cycles has a vertex-ordering with $\sum_u \text{succ}(u) \leq \frac{m}{2} - \frac{n}{6}$. For directed graphs with 2-cycles simply delete both edges in each 2-cycle, apply the above result, and insert the 2-cycles back into the graph. This adds one successor to one vertex, and increases the number of edges by two. Thus the same bound $\sum_u \text{succ}(u) \leq \frac{m}{2} - \frac{n}{6}$ holds. Using this ordering as the insertion ordering in algorithm MEDIAN PLACEMENT, by Lemma 8 with $|X| \leq n$, we obtain the following result.

Theorem 7. *Every n-vertex m-edge directed graph has a vertex ordering, which can be computed in linear time, with total imbalance at most $\frac{m}{2} + \frac{5n}{6}$.* □

Applying Theorem 7 with the algorithm of Biedl and Kaufmann [3] for orthogonal graph drawing with bounded aspect ratios, yields an improved bound of $(\frac{3}{4}m + \frac{5}{12}n) \times (\frac{3}{4}m + \frac{5}{12}n)$ for the area, compared with area $(\frac{3}{4}m + \frac{1}{2}n) \times (\frac{3}{4}m + \frac{1}{2}n)$ as stated in [3, Theorem 5].

5 Partially Fixed Orderings of Bipartite Graphs

We have seen that the MEDIAN PLACEMENT heuristic finds an optimal ordering for an acyclic directed graph, but in general, does not necessarily find an optimal ordering. We now turn to another special case where this algorithm finds an optimal ordering.

Consider the following variant of the balanced ordering problem: Given a bipartite graph $G = (A, B; E)$ and a fixed ordering of the vertices of A, how difficult is it to insert the vertices of B into this ordering so that the resulting ordering has minimum total imbalance? There are actually three variants of the problem. We can consider the total imbalance, or only the imbalance of the vertices in B, or only the imbalance of vertices in A. We now show that the first two of these problems are solvable with the MEDIAN PLACEMENT heuristic, whereas (surprisingly so) the third problem is NP-complete.

5.1 Total Imbalance and Imbalance in B

If only the final imbalance of vertices in B counts, then the MEDIAN PLACEMENT heuristic determines a perfectly balanced vertex-ordering, since a vertex $v \in B$ is placed in the middle of its neighbors, and no neighbor of v is inserted into the current ordering after v is inserted. We now prove that a variant of the MEDIAN PLACEMENT heuristic determines an optimal vertex-ordering if we count the imbalance of all vertices.

Theorem 8. *Given a bipartite graph $G = (A, B; E)$ and a fixed vertex-ordering of A, there is a linear time algorithm which determines an optimal vertex-ordering of G.*

Proof. It follows from the same technique used in the proof of Lemma 9 that there is an optimal vertex-ordering in which each vertex in B is placed in (one of) its median position(s). Thus we need only consider such vertex-orderings. A vertex in

B with even degree has one median position, and a vertex in B with with odd degree has two median positions (either side of its median neighbor). Which of these two positions a vertex in B with odd degree is placed only affects the imbalance of the median neighbor. Recall that for each vertex $v \in A$, $X(v)$ is the set of vertices $u \in B$ with odd degree such that v is the median neighbor of u.

Thus an optimal vertex-ordering can be determined as follows. Starting with the given ordering of A, apply the MEDIAN PLACEMENT heuristic using an arbitrary insertion ordering for B. For each vertex $v \in A$, partition $X(v)$ into sets $L(v)$ and $R(v)$ such that by placing the vertices in $L(v)$ immediately to the left of v, and placing the vertices in $R(v)$ immediately to the right of v, the imbalance of v is minimized. To do so, we also count the neighbors of v not in $X(v)$ in the imbalance of v; for each such neighbor we know whether it will be placed to the left or to the right of v. In the resulting ordering, each vertex in B is in (one of) its median position(s), and subject to this constraint, each vertex in A has minimum imbalance. Thus the ordering is optimal. The partitioning step and thus the entire algorithm can be computed in linear time. □

Consider the following algorithm to compute a vertex-ordering of a bipartite graph $G = (A, B; E)$. For every vertex-ordering of A, apply the algorithm described in Theorem 8 with this ordering of A fixed. By Theorem 8 this algorithm will compute an optimal vertex-ordering of G. We therefore have the following result.

Corollary 2. *There is a linear time algorithm to compute an optimal vertex-ordering of a bipartite graph $G = (A, B; E)$ if $|A| \in O(1)$.* □

From the standpoint of parameterized complexity (see [13]) this result is of some interest. While the balanced ordering problem is NP-complete for bipartite graphs, if the number of vertices in one bipartition is constant, the problem becomes fixed parameter tractable.

5.2 Imbalance in A

Theorem 9. *Given a bipartite graph $G = (A, B; E)$, it is NP-complete to determine whether a fixed vertex-ordering of A can be extended to a vertex-ordering of G in which all vertices in A are balanced.*

Proof. Let I be an instance of NAE-3SAT such that all literals are positive. Construct a graph G with one vertex c_j for each clause c_j, and four vertices x_i, x_i', l_i and r_i for each variable x_i. Connect each vertex x_i to each clause vertex c_j for which c_j contains the variable x_i. Also connect each of x_i and x_i' to both l_i and r_i. The resulting graph is bipartite, with all the x_i and x_i' vertices in one color class, and all remaining vertices in a second color class, whose vertex-ordering is fixed to

$$(l_1, l_2, \ldots, l_n,\ c_1, c_2, \ldots, c_m,\ r_1, r_2, \ldots, r_n) \ .$$

Suppose there is a vertex-ordering of G in which all fixed vertices are balanced. In particular, this means that for each i, one of x_i and x_i' is to the left of l_i and the other one is to the right. (No other vertices are connected to l_i.) Also, one of x_i and x_i' is to the left of r_i and the other one is to the right. (No other vertices are

connected to r_i.) Thus one of x_i and x'_i is to the left of l_i, and the other one is to the right of r_i. Let x_i be true if x_i is to the left of l_i, and false if x_i is to the right of r_i. Since the clause vertices are balanced, it is easy to see that this gives a solution to NAE-3SAT.

If I is satisfiable, construct a vertex-ordering with x_i to the left of the fixed part and x'_i to the right if x_i is true, and with x_i to the right of the fixed part and x'_i to the left if x_i is false. Every vertex l_i or r_i is a (1,1)-vertex, and every clause vertex is a (1,2)-vertex. Thus the vertex-ordering is perfectly balanced, and therefore the problem is NP-complete. □

While the the above problem is NP-complete in general, it becomes solvable if the maximum degree of the vertices in B is two (regardless of the degrees of vertices in A). In fact, we prove the following stronger result.

Lemma 10. *Given a bipartite graph $G = (A, B; E)$ such that every vertex in B has degree at most two, there is a polynomial time algorithm to extend a fixed vertex-ordering of A into a vertex-ordering of G such that every vertex in A is balanced.*

Proof. We proceed by induction on the number of edges. The claim clearly holds if G has no edges. Assume G has an edge. If G contains a cycle $C = (v_1, u_1 ..., v_k, u_k)$, then without loss of generality assume v_1 is the leftmost vertex in the ordering of A, and $v_i \in A$ and $u_i \in B$ for $1 \leq i \leq k$. Find a balanced ordering of $G - C$ by induction. Insert u_1 to the left of v_1 in the ordering, and for each vertex v_i, $2 \leq i \leq k$, if u_{i-1} is to the left of v_i, put u_i to the right of v_i and vice versa. Since v_1 is the leftmost vertex, the last vertex u_k can be placed to the right of v_1 regardless of what side of v_k it has to be placed. We have added one predecessor and one successor to every vertex in A, so the ordering again is balanced. If G contains no cycle, then it is a forest. Let P be a path of G whose endpoints are leaves, and insert the vertices in $P \cap B$ into the ordering in a similar manner to that for cycles. If a vertex in A has degree two in P then it will remain balanced. If a vertex in A has degree one in P then it is a leaf of G, has no more incident edges in the remaining part of G, has odd degree in the original G, and will have an imbalance of one in the vertex-ordering. Now, remove P from G, and repeat the above step until G is empty. At this point, all even degree vertices in A are balanced, and all odd degree vertices in A have an imbalance of one. □

6 Conclusion and Open Problems

In this paper we have considered the problem of determining a balanced ordering of the vertices of a graph. This problem is shown to be NP-hard by a reduction from NAE-3SAT, and remains NP-hard for bipartite simple graphs with maximum degree six. We then describe and analyze a number of methods for determining a balanced vertex ordering, obtaining optimal orderings for directed acyclic graphs and graphs with maximum degree three. It would be interesting to know if there are polynomial time algorithms for determining optimal vertex orderings of graphs with maximum degree four or five. Another direction for future research is to investigate balanced vertex-orderings of planar graphs. Note that, it has recently

been shown by Kratochvíl and Tuza [21] that Planar NAE-3SAT is solvable in polynomial time.

Acknowledgments

Thanks to Rudolf Fleischer, Erik Demaine, and Ulrik Brandes for helpful observations and discussions.

References

[1] B. BERGER AND P. W. SHOR, Tight bounds for the maximum acyclic subgraph problem. *J. Algorithms*, **25**:1–18, 1997.

[2] T. C. BIEDL AND G. KANT, A better heuristic for orthogonal graph drawings. *Comput. Geom.*, **9(3)**:159–180, 1998.

[3] T. C. BIEDL AND M. KAUFMANN, Area-efficient static and incremental graph drawings. In R. BURKHARD AND G. WOEGINGER, eds., *Proc. 5th Annual European Symp. on Algorithms (ESA'97)*, vol. 1284 of *Lecture Notes in Comput. Sci.*, pp. 37–52, Springer, 1997.

[4] T. C. BIEDL, B. MADDEN, AND I. G. TOLLIS, The three-phase method: A unified approach to orthogonal graph drawing. *International J. Comp. Geometry Appl.*, **10(6)**:553–580, 2000.

[5] M. BLUM, R. W. FLOYD, V. PRATT, R. L. RIVEST, AND R. E. TARJAN, Time bounds for selection. *J. Comput. System Sci.*, **7**:448–461, 1973.

[6] J. CHERIYAN AND J. H. REIF, Directed s-t numberings, rubber bands, and testing k-vertex connectivity. *Combinatorica*, **14(4)**:435–451, 1994.

[7] M. CHROBAK AND D. EPPSTEIN, Planar orientations with low out-degree and compaction of adjacency matrices. *Theoret. Comput. Sci.*, **86(2)**:243–266, 1991.

[8] T. H. CORMEN, C. E. LEISERSON, AND R. L. RIVEST, *Introduction to Algorithms*. McGraw-Hill, 1990.

[9] H. DE FRAYSSEIX AND P. O. DE MENDEZ, Regular orientations, arboricity, and augmentation. In R. TAMASSIA AND I. G. TOLLIS, eds., *Proc. DIMACS International Workshop on Graph Drawing (GD'94)*, vol. 894 of *Lecture Notes in Comput. Sci.*, pp. 111–118, Springer, 1995.

[10] H. DE FRAYSSEIX, P. O. DE MENDEZ, AND P. ROSENSTIEHL, Bipolar orientations revisited. *Discrete Appl. Math.*, **56(2-3)**:157–179, 1995.

[11] H. DE FRAYSSEIX, J. PACH, AND R. POLLACK, How to draw a planar graph on a grid. *Combinatorica*, **10(1)**:41–51, 1990.

[12] P. F. DIETZ AND D. D. SLEATOR, Two algorithms for maintaining order in a list. In *Proc. 19th Annual ACM Symp. on Theory of Comput. (STOC'87)*, pp. 365–372, ACM, 1987.

[13] R. G. DOWNEY AND M. R. FELLOWS, *Parameterized complexity*. Springer, 1999.

[14] P. EADES, X. LIN, AND W. F. SMYTH, A fast and effective heuristic for the feedback arc set problem. *Inform. Process. Lett.*, **47(6)**:319–323, 1993.

[15] P. EADES, A. SYMVONIS, AND S. WHITESIDES, Three dimensional orthogonal graph drawing algorithms. *Discrete Applied Math.*, **103**:55–87, 2000.

[16] J. EBERT, st-ordering the vertices of biconnected graphs. *Computing*, **30(1)**:19–33, 1983.

[17] S. EVEN AND R. E. TARJAN, Computing an st-numbering. *Theoret. Comput. Sci.*, **2(3)**:339–344, 1976.

[18] G. KANT, Drawing planar graphs using the canonical ordering. *Algorithmica*, **16(1)**:4–32, 1996.

[19] G. KANT AND X. HE, Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems. *Theoret. Comput. Sci.*, **172(1-2)**:175–193, 1997.

[20] R. M. KARP, Reducibility among combinatorial problems. In *Complexity of Computer Communications*, pp. 85–103, Plenum Press, 1972.

[21] J. KRATOCHVÍL AND Z. TUZA, On the complexity of bicoloring clique hypergraphs of graphs. In *Proc. 11th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'00)*, pp. 40–41, ACM, 2000.

[22] A. LEMPEL, S. EVEN, AND I. CEDERBAUM, An algorithm for planarity testing of graphs. In *Proc. International Symp. on Theory of Graphs*, pp. 215–232, Gordon and Breach, 1967.

[23] Y. LIU, A. MORGANA, AND B. SIMEONE, A linear algorithm for 2-bend embeddings of planar graphs in the two-dimensional grid. *Discrete Appl. Math.*, **81(1-3)**:69–91, 1998.

[24] Y. MAON, B. SCHIEBER, AND U. VISHKIN, Parallel ear decomposition search (EDS) and st-numbering in graphs. *Theoret. Comput. Sci.*, **47(3)**:277–298, 1986.

[25] A. PAPAKOSTAS AND I. G. TOLLIS, Algorithms for area-efficient orthogonal drawings. *Comput. Geom.*, **9**:83–110, 1998.

[26] P. ROSENSTIEHL AND R. E. TARJAN, Rectilinear planar layouts and bipolar orientations of planar graphs. *Discrete Comput. Geom.*, **1(4)**:343–353, 1986.

[27] T. J. SCHAEFER, The complexity of satisfiability problems. In *Proc. 10th Annual ACM Symp. on Theory of Comput. (STOC'78)*, pp. 216–226, ACM, 1978.

[28] R. TAMASSIA AND I. G. TOLLIS, Planar grid embedding in linear time. *IEEE Trans. Circuits Systems*, **36(9)**:1230–1234, 1989.

[29] R. E. TARJAN, Two streamlined depth-first search algorithms. *Fund. Inform.*, **9(1)**:85–94, 1986.

[30] D. R. WOOD, An algorithm for three-dimensional orthogonal graph drawing. In S. WHITESIDES, ed., *Proc. 6th International Symp. on Graph Drawing (GD'98)*, vol. 1547 of *Lecture Notes in Comput. Sci.*, pp. 332–346, Springer, 1998.

[31] D. R. WOOD, Multi-dimensional orthogonal graph drawing with small boxes. In J. KRATOCHVIL, ed., *Proc. 7th International Symp. on Graph Drawing (GD'99)*, vol. 1731 of *Lecture Notes in Comput. Sci.*, pp. 311–222, Springer, 1999.

[32] D. R. WOOD, Minimising the number of bends and volume in three-dimensional orthogonal graph drawings with a diagonal vertex layout, submitted. See Technical Report CS-AAG-2001-03, Basser Department of Computer Science, The University of Sydney, 2001.

[33] D. R. WOOD, Optimal three-dimensional orthogonal graph drawing in the general position model. *Theoret. Comput. Sci.*, to appear.