Available online at www.sciencedirect.com
SCIENCE DIRECT®
Discrete Applied Mathematics 148 (2005) 27–48
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Balanced vertex-orderings of graphs

Therese Biedl[a,1], Timothy Chan[a,1], Yashar Ganjali[b,2], Mohammad Taghi Hajiaghayi[c,2], David R. Wood[d,3]

[a]*School of Computer Science, University of Waterloo, Waterloo, Canada ON N2L 3G1*
[b]*Department of Electrical Engineering, Stanford University, Stanford, CA 94305, USA*
[c]*Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*
[d]*School of Computer Science, Carleton University, Ottawa, Canada ON K1S 5B6*

Received 6 June 2002; received in revised form 5 November 2004; accepted 1 December 2004

Abstract

In this paper 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 $\mathcal{NP}$-hard, and remains $\mathcal{NP}$-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, trees, and graphs with maximum degree three. For undirected graphs, we obtain a 13/8-approximation algorithm. 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 $\mathcal{NP}$-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

E-mail addresses: biedl@uwaterloo.ca (T. Biedl), tmchan@uwaterloo.ca (T. Chan), yganjali@Stanford.edu (Y. Ganjali), hajiagha@theory.lcs.mit.edu (M.T. Hajiaghayi), davidw@scs.carleton.ca (D.R. Wood).

[1] Research supported by NSERC.

[2] Completed while a graduate student in the School of Computer Science at the University of Waterloo.

[3] Partially completed while at Monash University, McGill University, The University of Sydney (supported by the ARC), and Charles University (supported by COMBSTRU and ITI).

0166-218X/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2004.12.001

1. Introduction

A number of algorithms for graph drawing use a 'balanced' ordering of the vertices of the graph as a starting point [22,23,31,37,38]. Here balanced means that the neighbors of each vertex v are as evenly distributed to the left and right of v as possible. In this paper 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 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 $v_i v_j \in E(v_i)$ with $i < j$ is a *successor edge* of v_i, and v_j is a *successor* of v_i. Similarly each edge $v_i v_j \in E(v_i)$ 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, $\text{pred}_\pi(v_i) = |\{v_i v_j \in E(v_i) : j < i\}|$ and $\text{succ}_\pi(v_i) = |\{v_i v_j \in E(v_i) : i < j\}|$. 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

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

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 $\phi(v) \leqslant 1$.

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. For a given graph, a vertex-ordering with minimum total imbalance is said to be *optimal*. We are interested in the following problem.

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

The balanced vertex-ordering problem can be described in a number of different ways. In a particular vertex-ordering, define

$$\psi(v) = \max\{\text{succ}(v), \text{pred}(v)\}.$$

Then

$$\phi(v) = 2\,\psi(v) - \deg(v). \tag{1}$$

Hence the problem of finding an optimal vertex-ordering is equivalent to finding a vertex-ordering that minimizes

$$\sum_{v \in V} \psi(v). \tag{2}$$

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

There is another equivalent formulation of the balanced vertex-ordering problem, which shall prove useful to consider. In a particular vertex-ordering, let $\phi'(v) = 2\lfloor\frac{1}{2}|\mathrm{succ}(v) - \mathrm{pred}(v)|\rfloor$. 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 [37] 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 $\mathcal{NP}$-complete. In particular, we prove that determining whether a given graph has a perfectly balanced vertex-ordering is $\mathcal{NP}$-complete, and remains $\mathcal{NP}$-complete for bipartite graphs with maximum degree six.

Section 3 considers the balanced vertex-ordering problem on weighted trees. We prove that this problem is (weakly) $\mathcal{NP}$-complete in general. On the other hand, we give a pseudo-polynomial time algorithm for its solution that runs in linear time in the case of unweighted trees.

Section 4 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 $\mathrm{pred}(v) = 0$ or $\mathrm{succ}(v) = 0$. The same algorithm determines optimal vertex-orderings of undirected graphs with maximum degree three.

Section 5 describes and analyses an algorithm for determining a balanced vertex-ordering of an arbitrary graph. This algorithm has been successfully used in [3,36] 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. We prove that this algorithm is a linear-time 13/8-approximation algorithm for the problem of minimizing (2) in undirected graphs.

In Section 6 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 $\mathcal{NP}$-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 $\mathcal{NP}$-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 $2 \leqslant |c| \leqslant 3$. 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 $\mathcal{NP}$-complete [33], and it is well-known (see [26] for example) that NAE-3SAT remains $\mathcal{NP}$-complete if all literals are positive and/or every variable x has $d_x \leqslant 3$.

Theorem 1. *Determining if a given graph has a perfectly balanced vertex-ordering is $\mathcal{NP}$-complete, and remains $\mathcal{NP}$-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 $d_x \leqslant 3$. 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 $x_j x_{j+1}$, $1 \leqslant j \leqslant 2d_x - 1$, to G, along

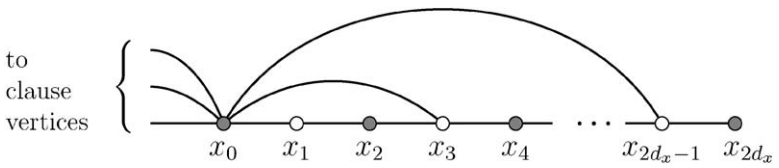

Fig. 1. The gadget associated with a variable x.

with the edges x_0x_{2j-1}, $1 \leqslant j \leqslant d_x$. In addition, add a *clause vertex* c_0 to G for each clause $c \in C$, and insert an edge x_0c_0 for each variable x appearing in c.

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 $\{x_{2j-1}, 1 \leqslant j \leqslant d_x\}$). Thus x_0 is balanced. The vertices x_j, $1 \leqslant j \leqslant 2d_x$, are either $(1, 1)$, $(1, 2)$ or $(0, 1)$-vertices, 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 $\deg(c_0) \leqslant 3$, 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 $\{c_0 : c \in C\} \cup \{x_{2j-1} : x \in U, 1 \leqslant j \leqslant d_x\}$ and $\{x_{2j} : x \in U, 0 \leqslant j \leqslant d_x\}$). 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 $\mathcal{NP}$-complete [33], and the construction of G is polynomial, testing if a graph has a perfectly balanced vertex-ordering is $\mathcal{NP}$-complete for simple bipartite graphs with maximum degree six. □

For an intended application in 3-D orthogonal graph drawing [38] 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 $\mathcal{NP}$-completeness in this case, at least for multigraphs.

Lemma 2. *Determining if a bipartite undirected multigraph with minimum degree five and maximum degree six has a perfectly balanced vertex-ordering is $\mathcal{NP}$-complete.*

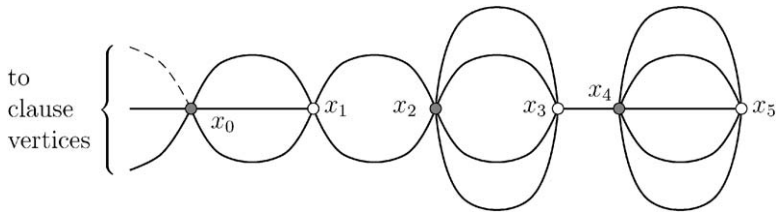

Fig. 2. The gadget associated with a variable x.

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 $d_x \geqslant 4$, 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 receives the same value as 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.

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, $1 \leqslant i \leqslant 5$, 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 a similar argument as 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). By the same argument in Theorem 1, it follows that 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. [17] and Wood [38], 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 [38]. As a consequence of Lemma 2 we therefore have the following result.

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

3. Weighted trees

A natural generalization of the balanced ordering problem is to consider weighted graphs. Given a vertex-ordering $(v_1, v_2, \ldots, v_n)$ of a graph $G=(V, E)$ with positive integer weights $\omega : E \to \mathbb{N}$ on the edges of G, for each vertex $v_i \in V$, we define $\mathrm{pred}(v_i)$ to be the sum of the weights of the predecessor edges of v_i, and $\mathrm{succ}(v_i)$ to be the sum of the weights of the successor edges of v_i. That is,

$$\mathrm{pred}(v_i) = \sum_{\substack{v_i v_j \in E(v_i) \\ j<i}} \omega(v_i v_j) \quad \text{and} \quad \mathrm{succ}(v_i) = \sum_{\substack{v_i v_j \in E(v_i) \\ i<j}} \omega(v_i v_j).$$

Clearly these definitions with all edge-weights equal to one are equivalent to the unweighted case. (One can think of a graph with edge-weights as a multigraph where the multiplicity of an edge equals its weight.) Thus the weighted balanced ordering problem is $\mathcal{NP}$-complete (since the unweighted version is), but in fact, it remains $\mathcal{NP}$-complete even for trees, whereas the unweighted version is solvable on trees, as we now show.

Lemma 4. *It is $\mathcal{NP}$-complete to determine if a given weighted graph has a perfectly balanced vertex-ordering, and remains so for weighted trees.*

Proof. We reduce the partition problem to the weighted balanced ordering problem. Given a set $w_1, w_2, \ldots, w_n$ of positive integers, the partition problem (which is $\mathcal{NP}$-complete [25]) asks whether there is a set $I \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in I} w_i = \sum_{i \notin I} w_i$. Given positive integers $w_1, w_2, \ldots, w_n$, consider the star graph on $n+1$ vertices, which has one vertex connected to all other vertices, and with $w_1, w_2, \ldots, w_n$ being the weights on the edges. Let $W = \sum_i w_i$. For any ordering of the vertices, the total imbalance is at least W, since each leaf must have imbalance w_i. We have a vertex-ordering with total imbalance of W if and only if we can split $w_1, w_2, \ldots, w_n$ into two sets that each sum to exactly $\frac{1}{2}W$; that is, there is a solution to the partition problem. □

Thus the weighted problem is $\mathcal{NP}$-complete, even if the graph is a tree. However, it is only weakly $\mathcal{NP}$-complete, since the partition problem is only weakly $\mathcal{NP}$-complete.

We now describe a pseudo-polynomial time algorithm for determining a perfectly balanced vertex-ordering of a weighted tree.

WEIGHTED TREE ORDERING
Input: tree $G = (V, E)$ with edge-weights $\omega : E \to \mathbb{N}$.
Output: vertex-ordering of G

Let $(v_1, v_2, \ldots, v_n)$ be a pre-order vertex-ordering of G;
(that is, every vertex, except v_1, has exactly one predecessor).
Initialize the current ordering to be (v_1).
for $i = 1, 2, \ldots, n$ **do**
Let v_k be the predecessor of v_i (if $i > 1$).
Partition $E(v_i)$ into L_i and R_i such that:

- $\left|\left(\sum_{v_i v_j \in L_i} \omega(v_i v_j)\right) - \left(\sum_{v_i v_j \in R_i} \omega(v_i v_j)\right)\right|$ is minimized, and
- $v_i v_k \in R_i$ if $v_k v_i \in L_k$,and $v_i v_k \in L_i$ if $v_k v_i \in R_k$.

Insert each successor v_j of v_i into the current ordering

- to the right of v_i if $v_i v_j \in R_i$, and
- to the left of v_i if $v_i v_j \in L_i$.

end-for

Theorem 5. *The* WEIGHTED TREE ORDERING *algorithm determines a perfectly balanced vertex-ordering of the given graph in pseudo-polynomial time.*

Proof. Every vertex v_i, except for v_1 which is inserted into the current ordering at the beginning of the algorithm, is inserted into the current ordering in the kth iteration, where v_k is the (sole) predecessor of v_i. Thus every vertex is inserted into the current ordering exactly once.

In the ith partitioning step we can swap L_i and R_i if $v_i v_k \in L_i \cap L_k$ or $v_i v_k \in R_i \cap R_k$. Hence for all edges $v_i v_j \in E$, we have $v_i v_j \in L_i \cap R_j$ or $v_i v_j \in R_i \cap L_j$. Thus when vertices are inserted into the current ordering, a vertex v_i is to the left of an adjacent vertex v_j if and only if $v_i v_j \in R_i \cap L_j$. Therefore the imbalance

$$\phi(v_i) = \left| \left(\sum_{v_i v_j \in L_i} \omega(v_i v_j) \right) - \left(\sum_{v_i v_j \in R_i} \omega(v_i v_j) \right) \right|,$$

which is chosen to be minimum. Thus each v_i is balanced, and therefore the ordering is perfectly balanced.

Using a dynamic programming algorithm (see [21] for example) the partitioning of $E(v)$ can be completed in $O(W_v \cdot \deg(v))$ time, where W_v is the sum of the weights of the edges

incident to v. Hence the total time is proportional to

$$\sum_{v\in V}\sum_{vw\in E(v)} \deg(v)\cdot\omega(vw) = \sum_{vw\in E}\omega(vw)(\deg(v)+\deg(w)) \leqslant 2\Delta\sum_{vw\in E}\omega(vw) = 2\Delta W,$$

where W is the sum of all edge-weights, and Δ is the maximum degree of G. Clearly, $O(\Delta W)$ is pseudo-polynomial time. Note that for unweighted trees, the partition of $E(v)$ is trivial, and the algorithm runs in linear time. □

4. Connectivity and maximum degree

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

4.1. 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 $\deg(v_i)\geqslant 2$, we have $\mathrm{pred}(v_i)\geqslant 1$ and $\mathrm{succ}(v_i)\geqslant 1$. Lempel et al. [27] 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 [8] extended this result to directed graphs. Even and Tarjan [19,20] develop a linear time algorithm to compute an *st*-ordering of an undirected biconnected graph (also see [7,18,29,35]). Under the guise of *bipolar orientations*, *st*-orderings have also been studied in [9,14,32]. In related work, Papakostas and Tollis [31] 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. In general, *st*-orderings do not have minimum imbalance (in [4] we give an example of a graph for which every *st*-ordering is not optimal), but *st*-orderings immediately give the following upper bound on the total imbalance.

Lemma 6. *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.*

The following algorithm determines a vertex-ordering of a graph based on *st*-orderings of its biconnected components (*blocks*). In Corollary 13 below we prove that given an optimal vertex-ordering of each biconnected component, it is $\mathcal{NP}$-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 [2,28,34] as it gives bounds on the number of highly unbalanced vertices (see Lemma 7 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 $\mathscr{BC}(G)$ 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 7. *Let G be an undirected graph with k end-blocks, and assume $k \geqslant 2$; that is, G has at least one cut-vertex. Then* COMBINE *st*-ORDERINGS *algorithm determines a vertex-ordering in linear time, with one vertex v having* $\mathrm{pred}(v) = 0$, *and $k - 1$ vertices v having* $\mathrm{succ}(v) = 0$.

Proof. By the definition of *st*-ordering, a vertex $v \in V$ that is not s_i or t_i for some i, has $\mathrm{pred}(v) > 0$ and $\mathrm{succ}(v) > 0$. We now count the number of vertices with zero successors. A vertex s_i has $\mathrm{succ}(s_i) > 0$. A vertex t_i for which B_i is not an end-block has $\mathrm{succ}(t_i) > 0$. The vertex t_1, for which B_1 is an end-block, has $\mathrm{succ}(t_i) > 0$. The remaining vertices t_i with B_i an end-block have $\mathrm{succ}(t_i) = 0$. Hence the number of vertices v having $\mathrm{succ}(v) = 0$ is $k - 1$. We now count the number of vertices with zero predecessors. A vertex t_i has $\mathrm{pred}(t_i) > 0$. For each $i \geqslant 2$, 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 7.

Lemma 8. *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 9. *Every vertex-ordering of an undirected graph with k end-blocks has at least k vertices v having* $\mathrm{pred}(v) = 0$ *or* $\mathrm{succ}(v) = 0$.

Note that for a triangulated planar graph G, vertex-orderings can be determined that are more balanced than *st*-orderings. de Fraysseix et al. [15] show that G has a *canonical* vertex-ordering $(v_1, v_2, \ldots, v_n)$ with $\mathrm{pred}(v_i) \geqslant 2$ for every vertex v_i, $3 \leqslant i \leqslant n$, and with $\mathrm{succ}(v_i) \geqslant 1$ for every vertex v_i, $1 \leqslant i \leqslant n-1$. Kant [22] 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 [13]). Kant and He [23] show that if G is planar and 4-connected, then G has a vertex-ordering with every vertex v_i, $3 \leqslant i \leqslant n-2$, having $\mathrm{succ}(v_i) \geqslant 2$ and $\mathrm{pred}(v_i) \geqslant 2$. The next result follows.

Lemma 10. *An n-vertex m-edge* 4*-connected triangulated planar undirected graph has a vertex-ordering with total imbalance at most* $2m - 4n + 12$.

4.2. 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 11. *Any st-ordering of a biconnected undirected graph G with maximum degree at most* 3 *is optimal.*

Proof. Suppose G has n vertices. Clearly the result holds if $n = 2$. Assume from now on that $n \geqslant 3$. 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 $\phi(v) \geqslant 1$, it is easily seen that any vertex-ordering of G has total imbalance at least $n_3 + 4$. □

Theorem 12. *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 $\phi'(v) = 2\lfloor \frac{1}{2} |\mathrm{succ}(v) - \mathrm{pred}(v)| \rfloor$. 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 Lemmas 7 and 9, 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 12, an optimal ordering of a graph with maximum degree three can be determined in linear time. Hence, we have the following result.

Corollary 13. *Finding the optimal vertex-ordering of a graph is $\mathcal{NP}$-hard, even if given an optimal vertex-ordering of each biconnected component.*

5. Median placement algorithm

We now describe an algorithm for the balanced vertex-ordering problem. 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 et al. [5].

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. [6], and the algorithm of Dietz and Sleator [11] 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 14. *The algorithm* MEDIAN PLACEMENT *determines in linear time a vertex-ordering of a (directed) graph $G = (V, E)$ with total imbalance*

$$\sum_{v \in V} \phi(v) \leqslant |X| + \sum_{u \in V} \mathrm{succ}_I(u).$$

Proof. When a vertex u is inserted into the current ordering, the predecessors of u in I are precisely the neighbors of u that have already been inserted into I. Thus immediately after u is inserted, $\phi(u) = 0$ if $\mathrm{pred}_I(u)$ is even and $\phi(u) = 1$ if $\mathrm{pred}_I(u)$ 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 $\phi(u) \leqslant \mathrm{succ}_I(u)$ if $\mathrm{pred}_I(u)$ is even, and $\phi(u) \leqslant \mathrm{succ}_I(u) + 1$ if $\mathrm{pred}_I(u)$ is odd. Thus the total imbalance is at most $|X| + \sum_u \mathrm{succ}_I(u)$. □

5.1. Undirected graphs

Theorem 15. *The algorithm* MEDIAN PLACEMENT *determines in linear time a vertex-ordering of an n-vertex m-edge undirected graph with total imbalance*

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

Proof. That $\sum_v \phi(v) \leqslant m + |X|$ follows immediately from Lemma 14 since $\sum_u \mathrm{succ}_I(u) = m$ 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, $\phi(v) \leqslant \mathrm{succ}_I(v) - |X(v)|$ if $\mathrm{pred}_I(v) + |X(v)|$ is even, and $\phi(v) \leqslant 1 + \mathrm{succ}_I(v) - |X(v)|$ if $\mathrm{pred}_I(v) + |X(v)|$ is odd. Thus

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

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 15 provides an upper bound on the total imbalance that is tight in this case. Comparing the bound on the total imbalance established by the MEDIAN PLACEMENT algorithm (Theorem 15) versus the analogous bound for the imbalance of *st*-orderings of biconnected graphs (Lemma 6), the MEDIAN PLACEMENT algorithm is better for simple graphs with average degree at least five. On the other hand, for simple 4-connected triangulated planar graphs (which have average degree just under six), the bound in Lemma 10 is better than that in Theorem 15.

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

Lemma 16. *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 that minimizes $\phi(w_{(k+1)/2})$, we minimize the total imbalance. □

Recall that $\psi(v)$ denotes $\max\{\mathrm{succ}(v), \mathrm{pred}(v)\}$ for each vertex v in a vertex-ordering. As mentioned in Section 1, any vertex-ordering is a 2-approximation for the problem of minimizing $\sum_v \psi(v)$. This observation can be improved as follows.

Theorem 17. *There is a linear-time* 13/8*-approximation algorithm for the problem of determining a vertex-ordering of an undirected graph that minimizes* $\sum_v \psi(v)$.

Proof. We proceed by induction on $|V|$ with the hypothesis that every undirected graph $G = (V, E)$ with k vertices of odd degree, has a vertex-ordering with

$$\sum_{v\in V} \psi(v) \leqslant \frac{13}{8}\left(|E| + \frac{k}{2}\right).$$

This will imply the claimed approximation factor, since in every vertex-ordering of G,

$$\sum_{v\in V} \psi(v) \geqslant \sum_{v} \left\lceil \frac{\deg(v)}{2} \right\rceil = |E| + \frac{k}{2}.$$

First suppose that G has a vertex v of degree one. Let w be the neighbor of v. Let $G' = (V', E')$ be the subgraph of G induced by $V' = V\backslash\{v\}$. Say G' has k' vertices of odd degree. By induction, G' has a vertex-ordering with

$$\sum_{x\in V'} \psi(x) \leqslant \frac{13}{8}\left(|E| - 1 + \frac{k'}{2}\right).$$

Suppose that $\deg_G(w)$ is even. Then $\deg_{G'}(w)$ is odd, and $k' = k$. Insert v into the ordering of G' to minimize the resulting imbalance of w. Thus $\psi(w)$ is unchanged by the insertion

of v, and $\psi(v) = 1$. We obtain a vertex-ordering of G with

$$\sum_{x\in V} \psi(x) \leqslant 1 + \sum_{x\in V'} \psi(x) \leqslant 1 + \frac{13}{8}\left(|E| - 1 + \frac{k}{2}\right) < \frac{13}{8}\left(|E| + \frac{k}{2}\right).$$

Now suppose that $\deg_G(w)$ is odd. Then $\deg_{G'}(w)$ is even, and $k' = k-2$. Insert v arbitrarily into the ordering of G'. Thus $\psi(w)$ increases by at most one, and $\psi(v) = 1$. We obtain a vertex-ordering of G with

$$\sum_{x\in V} \psi(x) \leqslant 2 + \sum_{x\in V'} \psi(x) \leqslant 2 + \frac{13}{8}\left(|E| - 1 + \frac{k-2}{2}\right) < \frac{13}{8}\left(|E| + \frac{k}{2}\right).$$

This completes the case in which G has a vertex of degree one.

Now suppose that G has a vertex v of degree two. Let u and w be the neighbors of v. Let $G' = (V', E')$ be the graph obtained from G by contracting v. That is, $V' = V\backslash\{v\}$ and $E' = (E\backslash\{vu, vw\}) \cup \{uw\}$. Observe that G' has k vertices of odd degree, and $|E'| = |E| - 1$. By induction, G' has a vertex-ordering with

$$\sum_{x\in V'} \psi(x) \leqslant \frac{13}{8}\left(|E| - 1 + \frac{k}{2}\right).$$

Insert v into the ordering of G' between u and w. Thus $\psi(u)$ and $\psi(w)$ are unchanged, and $\psi(v) = 1$. We obtain a vertex-ordering of G with

$$\sum_{x\in V} \psi(x) \leqslant 1 + \sum_{x\in V'} \psi(x) \leqslant 1 + \frac{13}{8}\left(|E| - 1 + \frac{k}{2}\right) < \frac{13}{8}\left(|E| + \frac{k}{2}\right).$$

Now suppose that G has minimum degree three. By Theorem 15, the algorithm MEDIAN PLACEMENT determines a vertex-ordering of G with total imbalance $\sum_v \phi(v) \leqslant |E| + |V|/2$. By (1), $\phi(v) = 2\psi(v) - \deg(v)$. It follows that,

$$\sum_v \psi(v) \leqslant \frac{3|E|}{2} + \frac{|V|}{4}. \tag{3}$$

Let n_3 be the number of vertices in G with degree exactly three. Since the minimum degree is three,

$$2|E| = \sum_v \deg(v) \geqslant 3n_3 + 4(|V| - n_3) = 4|V| - n_3.$$

Hence $4|V| \leqslant 2|E| + n_3 \leqslant 2|E| + k \leqslant 2|E| + 13k$, and $24|E| + 4|V| \leqslant 26|E| + 13k$. Thus by (3),

$$\sum_v \psi(v) \leqslant \frac{3|E|}{2} + \frac{|V|}{4} \leqslant \frac{13}{8}\left(|E| + \frac{k}{2}\right),$$

as desired. The above approach can be implemented in linear time using the MEDIAN PLACEMENT algorithm, by placing the low degree vertices at the end of the insertion ordering. □

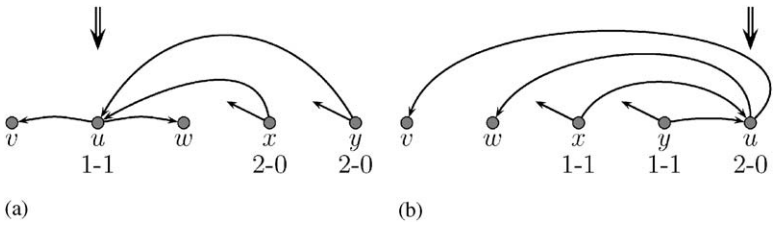

Fig. 3. Inserting vertex u into a vertex-ordering of a directed graph: (a) median placement insertion, (b) minimum imbalance insertion.

5.2. Directed graphs

We now analyze the MEDIAN PLACEMENT algorithm in the general case of directed graphs. For undirected graphs, Lemma 16 proves that the MEDIAN PLACEMENT algorithm inserts each vertex to minimize the increase in the total imbalance. The example in Fig. 3 shows that this property does not hold for directed graphs. Using the MEDIAN PLACEMENT algorithm the total imbalance becomes four, whereas there exists a position, illustrated in Fig. 3b, to insert u with total imbalance two.

Lemma 14 suggests that a good insertion ordering for the MEDIAN PLACEMENT algorithm applied to a directed graph, is one with small $\sum_u \mathrm{succ}(u)$. For any vertex-ordering of a directed graph, $\sum_u \mathrm{succ}(u)$ or $\sum_u \mathrm{pred}(u)$ 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 $\mathrm{succ}(u) = 0$ for all vertices u. Since such an ordering can be determined in linear time (see [10] 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 18. *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 $G \backslash F$ 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 $\sum_u \mathrm{succ}(u) = |F|$, finding a vertex-ordering with minimum $\sum_u \mathrm{succ}(u)$ is equivalent to finding a minimum feedback arc set, which is $\mathcal{NP}$-hard [25].

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 $\sum_u \mathrm{succ}(u)$ (taken over all vertex-orderings) is $\frac{m}{2} - \Theta(m/\sqrt{\Delta})$, and a vertex-ordering with $\sum_u \mathrm{succ}(u) = \frac{m}{2} - \Theta(m/\sqrt{\Delta})$ can be determined in $O(mn)$ time. Using this as the insertion ordering in algorithm MEDIAN PLACEMENT, by Lemma 14 with $|X| \leqslant n$, we obtain the following result.

Theorem 19. *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*

$$\sum_v \phi(v) \leqslant n + \frac{m}{2} - \Theta\left(\frac{m}{\sqrt{\Delta}}\right).$$

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. [16]. They give a linear time greedy heuristic for finding a feedback arc set, and prove an exact bound on $\sum_u \mathrm{succ}(u)$, 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 $\sum_u \mathrm{succ}(u) \leqslant \frac{m}{2} - \frac{n}{6}$. 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 $\sum_u \mathrm{succ}(u) \leqslant \frac{m}{2} - \frac{n}{6}$ holds. Using this ordering as the insertion ordering in algorithm MEDIAN PLACEMENT, by Lemma 14 with $|X| \leqslant n$, we obtain the following result.

Theorem 20. *Every n-vertex m-edge directed graph has a vertex-ordering, which can be computed in linear time, with total imbalance at most $\frac{m}{2} + \frac{5n}{6}$.*

The above result can be improved by the following randomized approach.

Theorem 21. *Every directed graph G with n vertices and m edges has a vertex-ordering with total imbalance $\frac{m+n}{2}$.*

Proof. Take a random permutation π of the vertices as the ordering. Consider a vertex v of (out-)degree d. We claim that in π, $\mathrm{succ}(v) = i$ and $\mathrm{pred}(v) = d - i$ with probability $\frac{1}{d+1}$. To prove this, we only need consider permutations of v and its neighbors. (There are equal numbers of permutations of the whole vertex set for each permutation of v and its neighbors.) Now, if v is placed in the $(i+1)$-st position, then $\mathrm{succ}(v) = i$ and $\mathrm{pred}(v) = d - i$. There are $d!$ such permutations. Thus with probability $d!/(d+1)! = 1/(d+1)$, we have $\mathrm{succ}(v) = i$ and $\mathrm{pred}(v) = d - i$, as claimed.

Define $\psi(v) = \max\{\mathrm{pred}(v), \mathrm{succ}(v)\}$. Thus

$$E[\psi(v)] = \sum_{i=0}^{d} \frac{\max(i, d-i)}{d+1} = \frac{1}{d+1}\left(\sum_{i=0}^{\lfloor d/2 \rfloor} (d-i) + \sum_{i=\lfloor d/2 \rfloor + 1}^{d} i\right).$$

For even d,

$$E[\psi(v)] = \frac{1}{d+1}\left(\frac{d}{2} + \frac{d}{2}\left(d + \frac{d}{2} + 1\right)\right) < \frac{3d+1}{4}.$$

For odd d,

$$E[\psi(v)] = \frac{1}{d+1}\left(\frac{d+1}{2}\left(d + \frac{d+1}{2}\right)\right) = \frac{3d+1}{4}.$$

Thus,

$$E\left[\sum_v \psi(v)\right] \leqslant \frac{3}{4}\sum_v \deg(v) + \frac{n}{4} = \frac{3m}{4} + \frac{n}{4}.$$

Thus there exists an ordering with $\sum_v \psi(v) \leqslant \frac{3m}{4} + \frac{n}{4}$. By (1), it follows that $\sum_v \phi(v) \leqslant \frac{m+n}{2}$. □

We can derandomize the proof of Theorem 21 using the method of conditional expectations to obtain a polynomial time algorithm. For details on this standard method of derandomization we refer the reader to the monograph of Motwani and Raghavan [30]. For undirected graphs G, Theorem 21 applied to the symmetric directed graph of G, matches the result in Theorem 15. In one sense, however, the median placement algorithm is superior to the randomized approach. Using conditional probabilities one has to choose the vertex that minimizes the increase in the total imbalance as the next vertex to be inserted, whereas Theorem 21 can be obtained using the MEDIAN PLACEMENT algorithm regardless of the insertion ordering.

Applying Theorem 21 with the algorithm of Biedl and Kaufmann [3] for orthogonal graph drawing with bounded aspect ratios, yields an improved bound of $\left(\frac{3}{4}m + \frac{1}{4}n\right) \times \left(\frac{3}{4}m + \frac{1}{4}n\right)$ for the area, compared with area $\left(\frac{3}{4}m + \frac{1}{2}n\right) \times \left(\frac{3}{4}m + \frac{1}{2}n\right)$ as stated in [3, Theorem 5].

6. Partially fixed orderings of bipartite graphs

We have seen that the MEDIAN PLACEMENT algorithm 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 algorithm, whereas (surprisingly so) the third problem is $\mathcal{NP}$-complete.

6.1. Total imbalance and imbalance in B

If only the final imbalance of vertices in B counts, then the MEDIAN PLACEMENT algorithm 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 algorithm determines an optimal vertex-ordering if we count the imbalance of all vertices.

Theorem 22. *Given a bipartite graph $G = (A, B; E)$ and a fixed vertex-ordering of A, there is a linear time algorithm that determines an optimal vertex-ordering of G.*

Proof. It follows from the same technique used in the proof of Lemma 16 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 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 algorithm 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. (This is similar to the partitioning step in the WEIGHTED TREE ORDERING algorithm in Section 3.) 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 22 with this ordering of A fixed. By Theorem 22 this algorithm will compute an optimal vertex-ordering of G. We therefore have the following result.

Corollary 23. *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 [12]) this result is of some interest. While the balanced ordering problem is $\mathcal{NP}$-complete for bipartite graphs, if the number of vertices in one color class is constant, the problem becomes fixed parameter tractable.

6.2. Imbalance in A

Theorem 24. *Given a bipartite graph $G = (A, B; E)$, it is $\mathcal{NP}$-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 the other color class, whose vertex-ordering is fixed to

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

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 every vertex in one color class is balanced. Therefore the problem is $\mathcal{NP}$-complete. □

While the above problem is $\mathcal{NP}$-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 25. *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 $C = (v_1, u_1, \ldots, v_k, u_k)$, 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 $1 \leqslant i \leqslant k$. 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, $2 \leqslant i \leqslant k$, 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. □

7. Conclusion and open problems

We have considered the problem of determining a balanced ordering of the vertices of a graph. This problem is shown to be $\mathcal{NP}$-hard, and remains $\mathcal{NP}$-hard for bipartite simple graphs with maximum degree six. Note that Kára et al. [24] have recently extended the method developed in this paper to prove that the balanced ordering problem is $\mathcal{NP}$-hard for graphs of maximum degree four, and for planar graphs. We then described and analyzed a number of methods for determining a balanced vertex-ordering, obtaining optimal orderings

for trees, directed acyclic graphs and graphs with maximum degree three. We presented a 13/8-approximation algorithm for the problem on undirected graphs. Obtaining a good approximation algorithm for directed graphs, and improving the approximation factors for undirected graphs are challenging open problems. Linear or semi-definite programming would seem a potential approach. However, we have found that these methods tend to give an approximation factor that is at least logarithmic in the size of the graph.

Acknowledgements

Thanks to Rudolf Fleischer, Erik Demaine, and Ulrik Brandes for helpful observations. We especially thank Mohammad Mahdian for fruitful discussions that lead to some of the results in Section 5.

References

[1] B. Berger, P.W. Shor, Tight bounds for the maximum acyclic subgraph problem, J. Algorithms 25 (1997) 1–18.
[2] T. Biedl, G. Kant, A better heuristic for orthogonal graph drawings, Comput. Geom. 9 (3) (1998) 159–180.
[3] T.C. Biedl, M. Kaufmann, Area-efficient static and incremental graph drawings, in: R.E. Burkhard, G.J. Woeginger (Eds.), Proceedings of the 5th Annual European Symposium on Algorithms (ESA '97), Lecture Notes in Computer Science, vol. 1284, Springer, Berlin, 1997, pp. 37–52.
[4] T. Biedl, T. Chan, Y. Ganjali, M. Hajiaghayi, D.R. Wood, Balanced vertex-orderings of graphs, Tech. Rep. TR-2002-01, School of Computer Science, Carleton University, Ottawa, Canada, 2002.
[5] T.C. Biedl, B.P. Madden, I.G. Tollis, The three-phase method: a unified approach to orthogonal graph drawing, International J. Comp. Geometry Appl. 10 (6) (2000) 553–580.
[6] M. Blum, R.W. Floyd, V. Pratt, R.L. Rivest, R.E. Tarjan, Time bounds for selection, J. Comput. System Sci. 7 (1973) 448–461.
[7] U. Brandes, Eager *st*-ordering, in: R.H. Möhring, R. Raman (Eds.), Proceedings of the 10th Annual European Symposium on Algorithms (ESA '02), Lecture Notes in Computer Science, vol. 2461, Springer, Berlin, 2002, pp. 247–256.
[8] J. Cheriyan, J.H. Reif, Directed *s*-*t* numberings, rubber bands, and testing *k*-vertex connectivity, Combinatorica 14 (4) (1994) 435–451.
[9] M. Chrobak, D. Eppstein, Planar orientations with low out-degree and compaction of adjacency matrices, Theoret. Comput. Sci. 86 (2) (1991) 243–266.
[10] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, McGraw-Hill, 1990.
[11] P.F. Dietz, D.D. Sleator, Two algorithms for maintaining order in a list, in: Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC'87), ACM, 1987, pp. 365–372.
[12] R.G. Downey, M.R. Fellows, Parameterized Complexity, Springer, Berlin, 1999.
[13] H. de Fraysseix, P. Ossona de Mendez, Regular orientations, arboricity, and augmentation, in: R. Tamassia, I.G. Tollis (Eds.), Proceedings of the DIMACS International Workshop on Graph Drawing (GD '94), Lecture Notes in Computer Science, vol. 894, Springer, Berlin, 1995, pp. 111–118.
[14] H. de Fraysseix, P. Ossona de Mendez, P. Rosenstiehl, Bipolar orientations revisited, Discrete Appl. Math. 56 (2–3) (1995) 157–179.
[15] H. de Fraysseix, J. Pach, R. Pollack, How to draw a planar graph on a grid, Combinatorica 10 (1) (1990) 41–51.
[16] P. Eades, X. Lin, W.F. Smyth, A fast and effective heuristic for the feedback arc set problem, Inform. Process. Lett. 47 (6) (1993) 319–323.
[17] P. Eades, A. Symvonis, S. Whitesides, Three dimensional orthogonal graph drawing algorithms, Discrete Appl. Math. 103 (2000) 55–87.

[18] J. Ebert, *st*-Ordering the vertices of biconnected graphs, Computing 30 (1) (1983) 19–33.
[19] S. Even, R.E. Tarjan, Computing an *st*-numbering, Theoret. Comput. Sci. 2 (3) (1976) 339–344.
[20] S. Even, R.E. Tarjan, Corrigendum: computing an *st*-numbering, Theoret. Comput. Sci. 2 (1976) 339–344; S. Even, R.E. Tarjan, Theoret. Comput. Sci. 4 (1) (1997) 123.
[21] M.R. Garey, D.S. Johnson, Computers and Intractability: a Guide to the Theory of NP-Completeness, Freeman, New York, 1979.
[22] G. Kant, Drawing planar graphs using the canonical ordering, Algorithmica 16 (1) (1996) 4–32.
[23] G. Kant, X. He, Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems, Theoret. Comput. Sci. 172 (1–2) (1997) 175–193.
[24] J. Kára, J. Kratochvíl, D.R. Wood, On the complexity of the balanced vertex ordering problem, 2004, Manuscript.
[25] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Complexity of Computer Communications, Plenum Press, New York, 1972, pp. 85–103.
[26] J. Kratochvíl, Z. Tuza, On the complexity of bicoloring clique hypergraphs of graphs, J. Algorithms 45 (1) (2002) 40–54.
[27] A. Lempel, S. Even, I. Cederbaum, An algorithm for planarity testing of graphs, in: Proceedings of the International Symposium on Theory of Graphs, Gordon and Breach, London, 1967, pp. 215–232.
[28] Y. Liu, A. Morgana, B. Simeone, A linear algorithm for 2-bend embeddings of planar graphs in the two-dimensional grid, Discrete Appl. Math. 81 (1–3) (1998) 69–91.
[29] Y. Maon, B. Schieber, U. Vishkin, Parallel ear decomposition search (EDS) and *st*-numbering in graphs, Theoret. Comput. Sci. 47 (3) (1986) 277–298.
[30] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, Cambridge, 1995.
[31] A. Papakostas, I.G. Tollis, Algorithms for area-efficient orthogonal drawings, Comput. Geom. 9 (1998) 83–110.
[32] P. Rosenstiehl, R.E. Tarjan, Rectilinear planar layouts and bipolar orientations of planar graphs, Discrete Comput. Geom. 1 (4) (1986) 343–353.
[33] T.J. Schaefer, The complexity of satisfiability problems, in: Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC'78), ACM, 1978, pp. 216–226.
[34] R. Tamassia, I.G. Tollis, Planar grid embedding in linear time, IEEE Trans. Circuits Systems 36 (9) (1989) 1230–1234.
[35] R.E. Tarjan, Two streamlined depth-first search algorithms, Fund. Inform. 9 (1) (1986) 85–94.
[36] D.R. Wood, Multi-dimensional orthogonal graph drawing with small boxes, in: J. Kratochvil (Ed.), Proceedings of the 7th International Symposium on Graph Drawing (GD '99), Lecture Notes in Computer Science, Springer, Berlin, 1999, pp. 311–322.
[37] D.R. Wood, Optimal three-dimensional orthogonal graph drawing in the general position model, Theoret. Comput. Sci. 299 (1–3) (2003) 151–178.
[38] D.R. Wood, Minimising the number of bends and volume in three-dimensional orthogonal graph drawings with a diagonal vertex layout, Algorithmica 39 (3) (2004) 235–253.