Portland State University

PDXScholar

University Honors Theses | University Honors College

Winter 3-2022

Minimality of Integer Bar Visibility Graphs

Emily DeHoff
Portland State University

Follow this and additional works at: https://pdxscholar.library.pdx.edu/honorstheses

 Part of the Discrete Mathematics and Combinatorics Commons

Let us know how access to this document benefits you.

Recommended Citation
DeHoff, Emily, "Minimality of Integer Bar Visibility Graphs" (2022). *University Honors Theses.* Paper 1174.
https://doi.org/10.15760/honors.1233

This Thesis is brought to you for free and open access. It has been accepted for inclusion in University Honors Theses by an authorized administrator of PDXScholar. Please contact us if we can make this document more accessible: pdxscholar@pdx.edu.

Minimality of Integer Bar Visibility Graphs

by

Emily DeHoff

An undergraduate honors thesis submitted in partial fulfillment of the
requirements for the degree of
Bachelor of Science
in
University Honors
and
Mathematics

Thesis Adviser
Dr. John S. Caughman

Portland State University
2022

Contents

1 Introduction

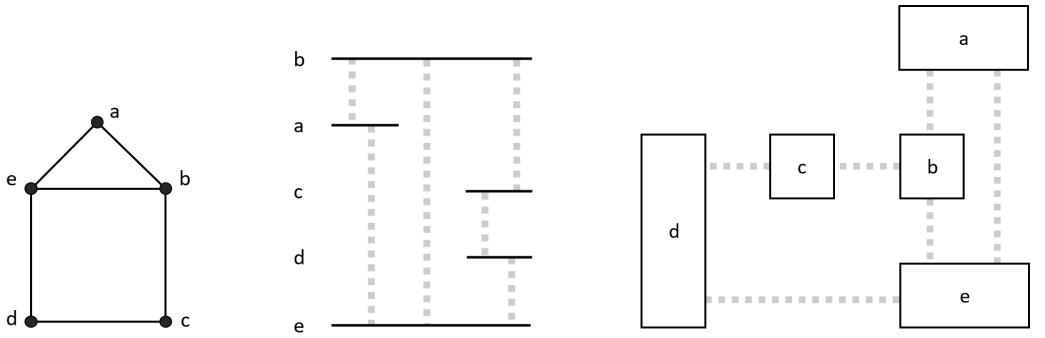
A graph is a mathematical structure often depicted as a collection of points in the plane (called vertices), together with a collection of simple curves (called edges) connecting pairs of these points.

Graphs are often used to study the relationships between objects, such as when mapping computer networks or modeling COVID transmission. In the 1970s, researchers found that graphs were a useful tool in the efficient design of integrated circuits, specifically in Very Large Scale Integration (VLSI). What began as an application of graph coloring [2] led to an in-depth study of visibility graphs [4, 6].

1.1 Visibility Graphs

A **visibility representation** of a graph G is an association between the set of vertices in G and a set of objects in the plane such that two objects have an unobstructed, positive width line of sight between them if and only if their corresponding vertices are adjacent in G. These visibility representations are also referred to as line-of-sight graphs, or simply visibility graphs.

Visibility graphs are typically categorized based on the objects used. In this paper, we focus primarily on **bar visibility representations**, using horizontal line segments, called bars, to represent the vertices of a graph G and vertical lines of sight between bars to indicate adjacency between vertices in G (Figure 1b). We also refer to this representation as a bar visibility graph (BVG). If a graph has such a representation, we say it is **bar representable**. Visibility graphs using rectangles and both vertical and horizontal lines of sight are also common in the literature and are referred to as **rectangle visibility representations** (Figure 1c).



(a) example of a bar representable graph

(b) a corresponding bar visibility representation

(c) a corresponding rectangle visibility representation for the same graph

Figure 1

Notice that in Figure 1a the vertices a and b are adjacent while b and d are not. We can see in Figure 1b that there is a clear line of sight between the bars a and b but visibility between b and d is obstructed by c. Similarly, in Figure 1c the rectangles a and b can 'see' each other, but b and d cannot.

In the context of VLSI, visibility graphs were first introduced by Garey, Johnson, and So [2] in 1976 as part of a graph coloring approach to testing for short-circuits in printed circuit boards. Roughly 10 years later, Wismath [6] and Tamassia and Tollis [4] provided several conditions on a graph in order for it to be representable as a bar or rectangle visibility graph. These conditions include planarity and constraints on cut-vertices.

1.2 Integer Bar Visibility Graphs

A natural question to ask when studying bar visibility graphs is how to minimize their size. For this question to be meaningful, we narrow our focus to BVGs where the endpoints of each bar lie on non-negative integer coordinate points and are included in the bars. We also require that lines of sight between bars have positive non-zero width. We call these **integer bar visibility graphs**. In this paper, when a graph has n vertices, we draw each bar at a distinct y-coordinate in the set $\{1, 2, ..., n\}$, so the height of a BVG will always be the number of vertices in the graph G. This allows us to focus on width when minimizing the size of a bar visibility representation. When describing integer bar visibility graphs, we refer to each unit interval along the x-axis as a **column**.

We use the term **width** to refer to the minimum number of columns required to represent a given bar-representable graph G, and we denote this by $w(G)$. In particular, a graph G cannot have a bar visibility representation with width less than $w(G)$.

To date, relatively little research has been published on minimizing the width of bar visibility graphs. One notable exception is the paper by Kant, Liotta, Tamassia, and Tollis [3], in which the authors consider the area required for both bar and rectangle visibility representations of directed trees. In their work, they present lower and upper bounds for the width required to represent free and rooted trees with an upward visibility graph. Here we seek to generalize and expand upon their work by allowing visibility both upward and downward, and by considering more general families of graphs.

1.3 Overview of Thesis

In this paper we study characterizations of bar visibility representations for several families of graphs, and we establish bounds on their width. Our work is organized as follows. In Section 2 we define the graph theoretic terms and notation used throughout this paper. In Section 3 we cover essential theorems from previous scholarship. In Section 4 we prove some lower bounds and basic results on the width of general graphs. In Section 5 we find the exact width for paths and cycles along with a sharp lower bound for the width of trees. In Section 6 we present our main results, providing a necessary condition for a graph to have width k. In Section 7 we use our main results to prove a surprising result about trees and show that there does not exist an upper bound on the width of families of graphs with specific degree constraints. And, finally, in Section 8, we present questions for further research.

2 Notation and Definitions

In this section, we collect a few basic definitions concerning graphs. For a more thorough introduction to the topic, we refer the reader to [3, 5].

A **graph** G is defined by a finite set $V(G)$, whose elements are called **vertices**, and a set $E(G)$ of 2-element subsets of $V(G)$, called **edges**. The number of vertices in G is represented by n. A drawing, or **embedding**, of G assigns each vertex v in $V(G)$ to a

point in the plane and each edge in $E(G)$ to a simple curve whose endpoints are the pair of vertices in that edge (see Figure 1a). When two vertices are connected by an edge, we say they are **adjacent**. The **degree** of a vertex v is the number of vertices adjacent to it and the **maximum degree** of a graph G is the highest degree of all vertices in G, denoted by $\Delta(G)$. Any vertex with degree 1 is called a **leaf**.

A graph is considered **planar** if it *can* be drawn in such a way that no two edges cross. Such a drawing is called a **planar embedding**. Note that not every drawing of a planar graph is a planar embedding. A **face** of a planar embedding is any simple closed region in the complement of the embedding. Roughly speaking, then, a face is any region that is either bounded by edges (with no vertices or edges within it), or the unbounded region outside the drawing.

A **path** in a graph G is any sequence of distinct, consecutively adjacent vertices. The **length** of a path is one less than the number of vertices in the path. A graph G is **connected** if there exists a path between every pair of vertices in G. We say a graph H is a **subgraph** of a graph G whenever $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. When a graph is disconnected, it contains more than one **component**, or maximal connected subgraph of the original graph. A **cut vertex** is a vertex whose removal increases the number of components in the graph. Likewise, a **cut edge** is an edge whose removal increases the number of components in the graph.

There are a number of important families of graphs we will encounter frequently in this paper. A **path** P_n is a sequence of n vertices $v_1, v_2, ..., v_n$ such that each vertex is adjacent to the next. A **cycle** C_n is a path with the addition of an edge between the first and last vertices. A **tree** (also referred to as a **free tree**) is a connected graph that contains no cycles as subgraphs. A **star** is a special type of tree that has one vertex v of degree $n-1$ and $n-1$ leaves adjacent to v. A **rooted tree** is a free tree T along with a distinguished vertex called the **root** of T. The **height** of a rooted tree T, denoted by $h(T)$, is the length of a longest path from the root of T to a leaf.

3 Prior Work

It is natural to wonder if every graph is bar-representable. Indeed, it is readily verified that all graphs on up to 4 vertices have a bar representation. However, in considering all graphs on up to 7 vertices, it is known that bar-representability coincides with planarity. In other words, if $n \leq 7$, then G is bar-representable if and only if G is planar. For the general case, Tamassia and Tollis [4] (and separately Wismath [6]) show that, for any number of vertices n, planarity is a necessary condition for a graph to be bar-representable. It is interesting to note, however, that the converse is not necessarily true. Wismath [6] found the smallest planar graph that is not bar-representable (see Figure 2).

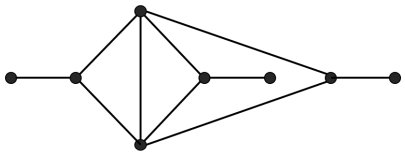


Figure 2: The smallest non-bar-representable planar graph.

Tamassia and Tollis [4] built on this by providing the necessary and sufficient conditions for a graph to be bar-representable. They found that a graph G is bar-representable if and only if it has a planar embedding where all cut vertices appear on the boundary of the same face. You can see in Figure 2 that the three cut vertices do not lie on the boundary of the same face, and no planar embedding exists such that they do share the same face. Thus this graph exemplifies the conditions laid out by Tamassia and Tollis.

We were able to find one piece of previous scholarship, by Kant et al [3], that considers the area required to represent graphs. Their work specifically deals with the area required to represent directed trees using the notion of *upward visibility*. For this kind of visibility, each directed edge (v_i, v_k), must correspond to a visibility where the bar for v_k is located above the bar for v_i. Their results are divided into two parts. The first looks at rooted trees, and the second at free trees.

Theorem 3.1 (Kant, Liotta, Tamassia, and Tollis, [3])**.** *Let T be a rooted tree with l leaves and height h. The area required by an upward bar-visibility representation of T is at least $(2l-1)\cdot h$.*

Theorem 3.2 (Kant, Liotta, Tamassia, and Tollis, [3])**.** *Let T be a free tree with n vertices, l leaves, and critical height h^*. The area required by a bar-visibility representation of T is $\Omega(l \cdot h^* + n - 1)$.*

We refer the interested reader to their paper for further details, including the definitions of critical height and Ω notation. Although these two results are relevant to our work in that they address the area (and by extension, the width) required to represent trees, their restriction to upward-visibility of directed trees reduces their applicability to the context of this paper.

4 Preliminary Results

For this project, we began our exploration of bar visibility graphs by constructing integer BVGs for every connected graph on up to 6 vertices (up to isomorphism). We used Python to write an algorithm that generates all possible bar visibility graphs with a given width and number of vertices (see Appendix). We used this to cross check our hand-drawn BVGs and reduce their size as appropriate until we found minimal representations of each graph. We then used this database of graphs and their respective bar visibility representations to identify patterns regarding the width of graphs. The following results came out of this initial investigation.

We begin with lower bounds for the width of any given graph. Our first result focuses on the number of leaves.

Lemma 4.1. *Given a bar-representable graph G with ℓ leaves, $w(G) \geq \lceil \frac{\ell}{2} \rceil$.*

Proof. Let G be a bar-representable graph with ℓ leaves. It is clear that each degree 1 vertex must have either an open line of sight upwards or an open line of sight downwards. This means each column has exactly two possible positions for a leaf, namely the top bar and the bottom bar. If we utilize every one of these leaf positions, this gives us a width of $\lceil \frac{\ell}{2} \rceil$. We can see in Figure 3 that if ℓ is odd, there is an open line of sight in one of the columns not being utilized by a leaf.

We want to show that G cannot be represented with a BVG narrower than $\lceil \frac{\ell}{2} \rceil$. By contradiction, suppose G can be represented with a BVG of width $\lceil \frac{\ell}{2} \rceil - 1$. Then by the

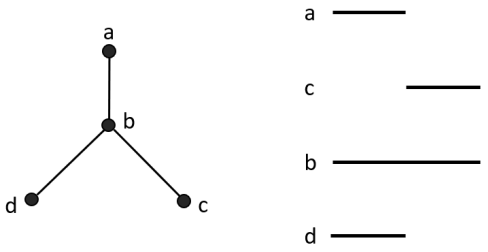


Figure 3: A graph with 3 leaves and width 2.

pigeonhole principle, there is some column that contains 3 vertices with degree 1. This is a contradiction because each degree 1 vertex must have a clear line of sight in one direction and each column has only two such lines of sight. Thus, a graph G with ℓ leaves has $w(G) \geq \lceil \frac{\ell}{2} \rceil$. □

Our second result gives a lower bound on the width of a single bar in a bar visibility representation based on the degree of its associated vertex.

Lemma 4.2. *The degree $d(v)$ of a vertex v forces a lower bound of $\lceil \frac{d(v)}{2} \rceil$ on the width of its associated bar.*

Proof. A bar with width r has $2r$ possible lines of sight. This means that a given vertex v in a graph G with degree $d(v)$ must be represented by a bar of width at least $\lceil \frac{d(v)}{2} \rceil$. □

As an example of this result, we can look at the vertex c in Figure 4. Notice that c has degree 4 and the width of its bar is 2. We build on this result by showing that the highest degree vertex in a graph G forces a lower bound on the width of the entire bar visibility graph of G.

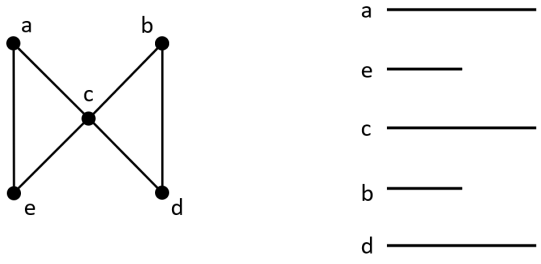


Figure 4: A graph with $\Delta(G) = 4$ and width 2.

Theorem 4.3. *Given a bar-representable graph G with maximum degree $\Delta(G)$, the width of G is bounded below by $w(G) \geq \lceil \frac{\Delta(G)}{2} \rceil$.*

Proof. Let G be a bar-representable graph with maximum degree $\Delta(G)$. Let v be a vertex with degree $d(v) = \Delta(G)$. By Lemma 4.2, we know that the width of the bar representing v is at least $\lceil \frac{\Delta(G)}{2} \rceil$. Thus, $w(G) \geq \lceil \frac{\Delta(G)}{2} \rceil$. □

Again, Figure 4 serves as an example. The maximum degree of this graph is 4, so the width of the BVG is at least 2. In this case, the width achieves its lower bound. We conclude this section with a result on the width of disconnected graphs.

Theorem 4.4. *Let G be the union of two disjoint bar-representable components, G_1 and G_2. Then $w(G) = w(G_1) + w(G_2)$.*

Proof. Let $G = G_1 \cup G_2$ be the union of two disjoint bar-representable components, G_1, G_2. We know that $w(G) \leq w(G_1) + w(G_2)$ since we can represent G by drawing a minimal BVG of G_1 next to a minimal BVG of G_2 with no lines of sight between.

For the opposite inequality, let us consider any minimum width representation of G. If any column contains bars for vertices from both G_1 and G_2, then some vertex in G_1 must be adjacent to some vertex in G_2. But this is a contradiction, since G_1 and G_2 are disjoint components. Thus, the bars for vertices in G_1 occupy a disjoint set of columns than the bars for vertices in G_2. Deleting each of these sets of columns in turn gives bar visibility representations of G_1 and G_2, respectively. By the definition of width, then, we must have $w(G) \geq w(G_1) + w(G_2)$. □

In Figure 5 we see an example that illustrates the situation for disconnected graphs. The components have width 3 and 1, while the entire graph has width 4.

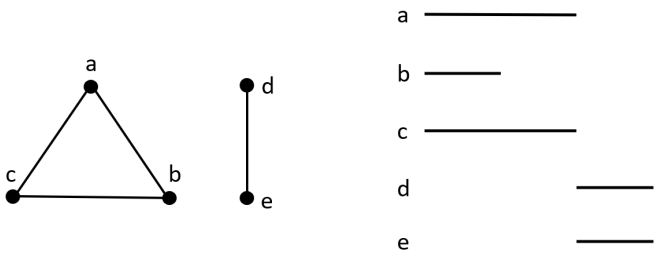


Figure 5: A graph with two disconnected components.

The previous result generalizes easily to any number of components.

Corollary 4.5. *Let G be the union of k disjoint, bar-representable components $G_1, ..., G_k$. Then $w(G) = \sum_{i=1}^{k} w(G_i)$.*

Proof. We can proceed by induction. The result is trivial when $k = 1$. Now fix any $k \geq 2$ and let $G = \bigcup_{i=1}^{k} G_i$. Define $G' = \bigcup_{i=1}^{k-1} G_i$ and, for our induction hypothesis, we assume that $w(G') = \sum_{i=1}^{k-1} w(G_i)$. Then by Theorem 4.4,

$$w(G) = w(G' \cup G_{k+1}) = \left(\sum_{i=1}^{k-1} w(G_i) \right) + w(G_k) = \sum_{i=1}^{k} w(G_i),$$

and the result follows. □

In light of this, we focus on connected graphs in this paper.

5 Families of Graphs

In this section, we look briefly at three families of graphs: trees, paths, and cycles. For each family, either we find an exact width for the graphs in that family, or we give bounds on the width for those graphs.

5.1 Trees

We begin our study of trees by showing that every tree is bar representable and by giving a lower bound on their width.

Theorem 5.1. *The width of a tree T with ℓ leaves is bounded below by $\lceil \frac{\ell}{2} \rceil$.*

Proof. Let T be a tree with ℓ leaves. Then T is clearly planar and, since T contains no cycles, every cut vertex must lie on the boundary of the same (unbounded) face. Thus T is bar representable. It follows from Lemma 4.1, $w(T) \geq \lceil \frac{\ell}{2} \rceil$. □

The graph shown in Figure 6 illustrates the bound of the previous theorem.

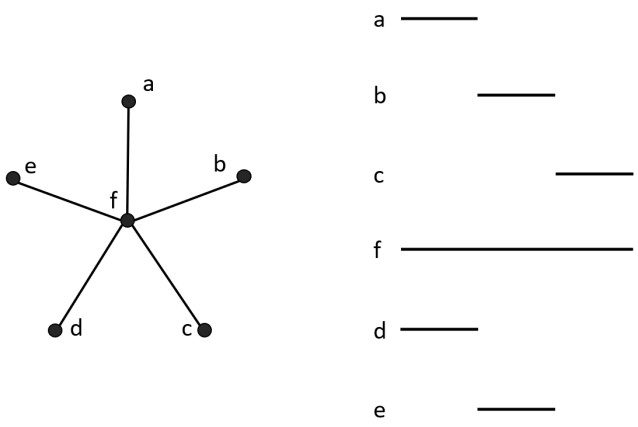


Figure 6: A star on 6 vertices with width 3.

From here, we are able to show that the width of a star will always equal this bound.

Corollary 5.2. *The width of a star S on n vertices is $\lceil \frac{n-1}{2} \rceil$.*

Proof. Let S be a star on n vertices. Then there are $n-1$ leaves, so we know that $w(S) \geq \lceil \frac{n-1}{2} \rceil$. Note that we can construct a bar representation of S by drawing the vertex of degree $n-1$ as a bar of length $\lceil \frac{n-1}{2} \rceil$. We then place $\lceil \frac{n-1}{2} \rceil$ bars of length 1 above this vertex and $\lfloor \frac{n-1}{2} \rfloor$ bars of length 1 below, as in Figure 6. This construction shows that stars can always be represented in a way that attains the lower bound for trees. □

We can use this result to show that there is no uniform upper bound on the width of all trees.

Corollary 5.3. *Let k be any positive integer. There exists a tree T with width $w(T) = k$.*

Proof. We know from Corollary 5.2 that the width of a star with ℓ leaves is $\lceil \frac{\ell}{2} \rceil$, so for any positive integer k, we can let T denote a star with $2k+1$ vertices. Then $w(T) = \lceil \frac{2k}{2} \rceil = k$. □

5.2 Paths

We begin our study of paths by first showing that every path is bar-representable and has a width of 1.

Lemma 5.4. *Given any path P_n, $w(P_n) = 1$.*

Proof. First note that a given path P_n is planar and there is only one face in any planar drawing of P_n, so every cut vertex necessarily lies on the boundary of the same face. Thus every path is bar representable.

We use induction to show that the width of P_n is 1. Let P_2 be a path on two vertices. We can represent P_2 with two width 1 bars, one directly above the other (Figure 7).

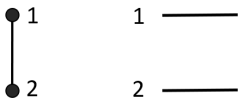


Figure 7: A path on 2 vertices with width 1.

Let P_k be a path of length k with a width 1 bar visibility representation (Figure 8).

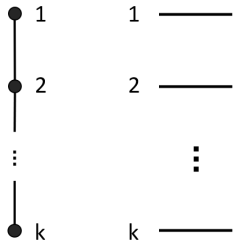


Figure 8: A path on k vertices with width 1.

Then a path P_{k+1} can be constructed by adding 1 vertex adjacent to kth vertex in P_k. Similarly, the BVG of P_{k+1} can be constructed by adding one width 1 bar under the kth bar (Figure 9).

Thus by induction, any path can be represented with a width 1 bar visibility graph. □

We follow this by showing that any graph with a width 1 bar visibility representation must be a path.

Lemma 5.5. *Given $w(G) = 1$, G must be a path.*

Proof. Let G be a graph such that $w(G) = 1$. When $n = 1$, G is simply a single vertex, or a path of length 0, so assume $n > 1$. Then $\Delta(G) = 2$. We also know G is connected. To see this, suppose G is not connected. Then there are two vertices in G with no path between them. Since the width of a path is 1 and the width of a graph with disconnected components is the sum of the widths of the components, $w(G)$ must be greater than 1. This is a contradiction, so G must be connected. Note that G has exactly 2 leaves, since there is only one column with an open upward line of sight from the top vertex and an open

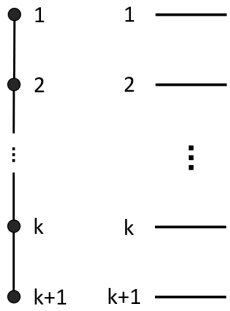


Figure 9: A path on $k+1$ vertices with width 1.

downward line of sight from the bottom vertex. So G is a connected graph with maximum degree 2 and exactly 2 leaves. By definition, this means G is a path. □

We conclude that having a width 1 bar visibility representation is a necessary and sufficient condition for a graph to be a path.

Theorem 5.6. *A graph G has $w(G) = 1$ if and only if G is a path.*

Proof. The proof of this theorem follows directly from Lemma 5.4 and Lemma 5.5. □

5.3 Cycles

Similar to our work with paths, we first show that every cycle can be represented with a width 2 bar visibility graph.

Lemma 5.7. *Given any cycle C_n, $w(C_n) = 2$.*

Proof. Let C_n be a cycle on n vertices. Clearly, C_n is planar and contains no cut vertices, so it is bar representable.

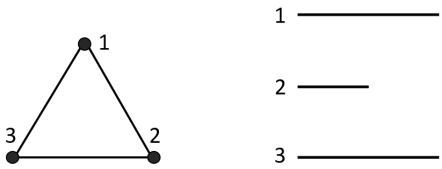


Figure 10: A cycle on 3 vertices with width 2.

We will use induction to show that $w(C_n) = 2$. Let C_3 be a cycle on 3 vertices. We can represent this with two width 2 bars and a width 1 bar between them (Figure 10). Suppose C_3 could be represented with a smaller BVG. Then $w(C_n) = 1$. By Theorem 5.6 this implies that C_3 is a path. This is a contradiction, so $w(C_3) = 2$.

Suppose C_k is a cycle on k vertices with a width 2 bar representation (Figure 11).

We can construct a cycle C_{k+1} on $k+1$ vertices by replacing the edge between vertices k and 1 with a length 2 path through a new vertex $k+1$. We can represent this new cycle as a bar visibility graph by inserting a width 1 bar between the bars k and 1 (Figure 12).

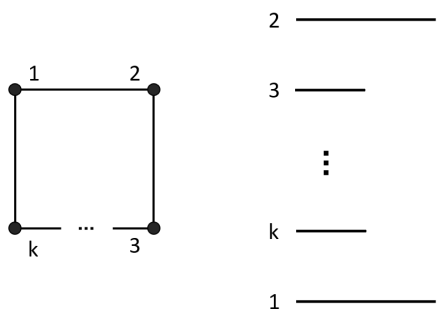


Figure 11: A cycle on k vertices with width 2.

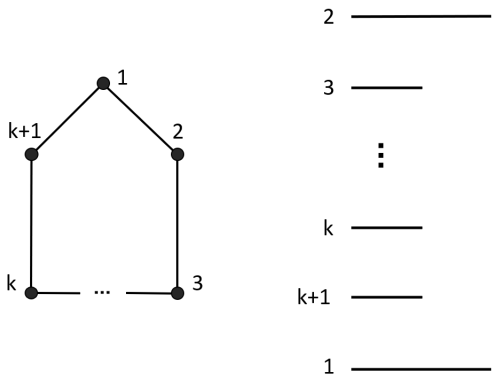


Figure 12: A cycle on $k+1$ vertices with width 2.

Notice that this does not increase the width of the BVG. Thus, any cycle can be represented with a width 2 bar visibility graph. □

Corollary 5.8. *In the above bar visibility representation of a cycle, any two adjacent vertices can be drawn as the top and bottom bars of the BVG.*

Proof. Given a cycle C_n, choose any two adjacent vertices, say, v_1 and v_2. Then there exists a path between v_1 and v_2 that does not use edge v_1v_2. We know from the construction in Lemma 5.4 that we can represent this with a stack of width 1 bars where v_1 is on one end and v_2 is on the other. The only unrepresented edge in C_n at this point is the edge directly between v_1 and v_2. So, we extend their respective bars to width 2 so they can see each other. This is now a bar visibility representation for the cycle C_n. □

In the following two corollaries, we extend the results of Lemma 5.7 to graphs composed of two cycles joined either by a cut vertex or a cut edge.

Corollary 5.9. *The width of two cycles sharing a cut vertex is 2.*

Proof. Let G be a graph composed of two cycles sharing a single cut vertex, like the one in Figure 4. We can label the two cycles C_1, C_2 and the shared cut vertex v. We know that G is planar and contains a single cut vertex, so G is bar representable.

By Theorem 5.6, we know that only paths have width 1, so a graph with a cycle subgraph cannot be represented with a width less than 2. Thus, $w(G) \geq 2$.

By Corollary 5.8, let v be identified with both the bottom width-2 bar in the representation of C_1 and the top width-2 bar in the representation of C_2. Then we can draw G by first creating a width-2 bar for v, then drawing the remaining bars in C_1 above v and the remaining bars in C_2 below v. Because v has width 2, none of the bars in C_1 have a clear line of sight to any of the bars in C_2. Thus, two cycles sharing only a cut vertex can be represented with a width-2 bar visibility graph. □

A similar argument can be given for graphs consisting of two cycles joined by a cut edge, as depicted in Figure 13.

Corollary 5.10. *The width of two cycles joined by a cut edge is 2.*

Proof. Let G be a graph composed of two cycles C_1, C_2 joined by a cut edge e, like the one in Figure 13. Label the endpoints of e as v_1 and v_2 (where v_1 is in C_1 and v_2 is in C_2). We know that G is planar and can be drawn such that the two cut vertices v_1 and v_2 lie on the boundary of the outside face. Thus G is bar representable.

By Theorem 5.6 we know that only paths have width 1. So, $w(G) \geq 2$.

By Corollary 5.8, let v_1 be the bottom width-2 bar in the representation of C_1 and let v_2 be the top width-2 bar in the representation of C_2. Then we can draw the BVG of C_1 above the BVG of C_2. Notice that v_1 can now see v_2. Thus, we can represent any graph composed of two cycles joined by a cut edge with a width-2 bar visibility graph. □

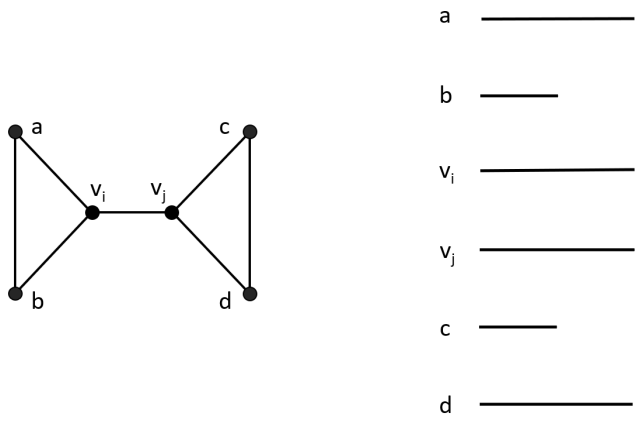


Figure 13: Two cycles joined by a cut edge with width 2.

6 Main Results

Our main results build on our work with paths and cycles in order to characterize all width 2 graphs and provide a necessary condition for a graph to be width k. We do this by orienting a graph and looking at its induced path subgraphs.

An **orientation** of a graph G assigns a direction to each edge in G. The first vertex of the edge becomes the tail and the second becomes the head. In an embedding of an oriented graph, each edge is drawn as an arrow from tail to head. An **acyclic orientation** of G is an orientation such that no induced subgraph of G contains a directed cycle.

A **directed path** is a path on vertices $v_1, v_2, ..., v_n$ with an orientation that satisfies the property that each edge is directed from v_i to v_{i+1} for all $1 \leq i < n$. We say two paths P_1 and P_2 are **coherently directed** if for every $v_j, v_k \in P_1$, $v_s, v_t \in P_2$ such that $v_j = v_s$ and $v_k = v_t$, then $j < k$ if and only if $s < t$. In other words, if two vertices appear in both P_1 and P_2, then those two vertices appear in the same order in both paths.

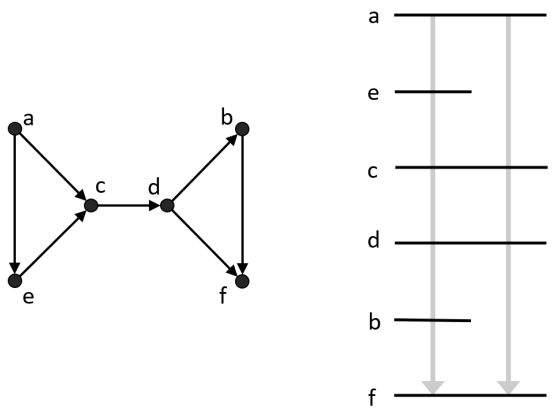


Figure 14: An orientation of the graph in Fig. 13 with 2 coherently directed paths.

We show that any graph with width 2 can be expressed as the union of 2 coherently directed paths. First we will look at an example. Consider the graph and its BVG in Figure 13. We know this graph has width 2. In Figure 14 we can see that each column in the BVG is a path within the graph. The path P_1 represented in the first column has vertices a, e, c, d, b, f. The path P_2 in the second column goes through the vertices a, c, d, f. Notice that the union of P_1 and P_2 gives us an acyclic orientation of the original graph in Figure 13. Observe that the edge between c and d is in both P_1 and P_2 and has the same direction in both paths, so there is no contradiction when taking the union.

Similarly, in Figure 15, we can see a graph with an acyclic orientation. Let P_1 be a path on the vertices f, e, a, b, c and P_2 a path on the vertices e, d, b. Notice that the union of P_1 and P_2 is the original graph. We can use these two paths to construct a width 2 BVG for this graph. We can see that the vertices e and b appear in the same order in both paths, so there is no conflict when constructing the BVG. We formalize our observations from these two examples in the following theorem.

Theorem 6.1. *Given a bar representable graph G, $w(G) = 2$ if and only if 2 is the fewest number of coherently directed paths whose union is an acyclic orientation of G.*

Proof. ($\Rightarrow$) Let G be a graph with width $w(G) = 2$. Then we can think about each column of the bar representation of G as a directed path going downwards through the BVG. Since there are two columns, we have two directed paths. Note that every edge in G is included in one or both of these paths. Thus the union of these two directed paths, $\hat{G}$, is an acyclic orientation of G.

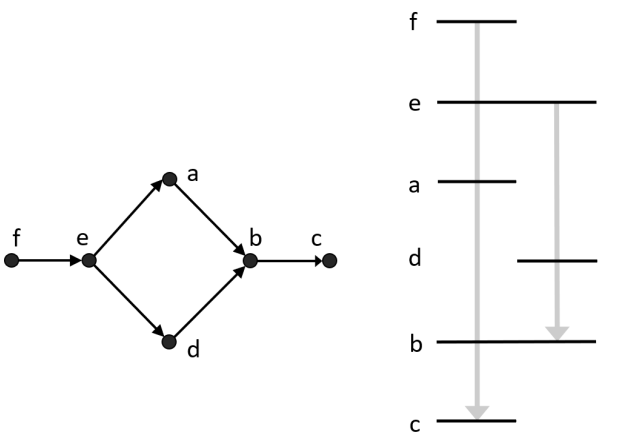


Figure 15: The union of two coherently directed paths with width 2.

Suppose we can use fewer than 2 paths. This would mean G is a single path. By Theorem 5.6, G has width 1. This is a contradiction, so 2 is the fewest number of paths we can use to build an acyclic orientation of G.

($\Leftarrow$) Let G be a graph with an acyclic orientation $\hat{G}$ such that $\hat{G} = P_1 \cup P_2$ where P_1, P_2 are coherently directed paths. We can begin constructing a bar representation of G by drawing P_1 as we did in Lemma 5.4. Then for each vertex included in both P_1 and P_2, extend the associated bar to width 2. Since the two paths are coherently directed, these extended bars are in the same order in both P_2 and P_1. Now we simply insert width 1 bars among the extended bars in the second column to complete P_2. Thus we have a width 2 representation of G. □

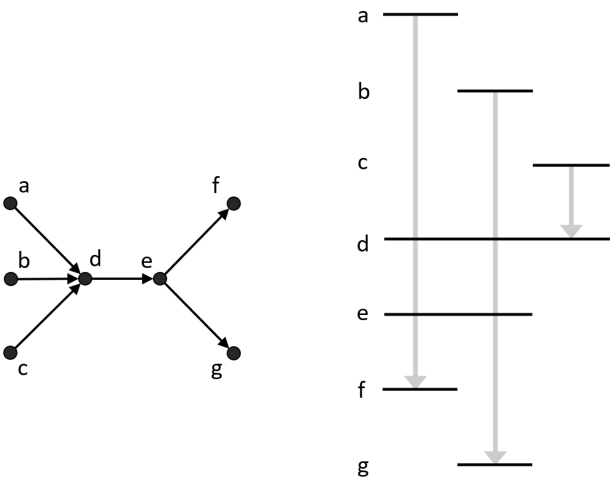


Figure 16: An acyclic orientation of a graph with width 3.

It is clear to see that the forward direction of this theorem scales up to $w(G) = k$. We can see an example with $k = 3$ in Figure 16. However we leave the other direction as a conjecture.

Corollary 6.2. *Given a bar representable graph G, if $w(G) = k$ then there exists an acyclic orientation $\hat{G}$ of G such that $\hat{G} = \bigcup_{i\in[k]} P_i$ where $P_1, P_2, ..., P_k$ are coherently directed paths.*

Conjecture 6.3. *Given a bar representable graph G, if k is the fewest number of coherently directed paths whose union is an acyclic orientation of G, then $w(G) = k$.*

7 Applications of Main Results

7.1 Trees with $w(T) > \lceil \frac{\ell}{2} \rceil$

We found in Theorem 5.1 that the width of a tree with ℓ leaves is bounded below by $\lceil \frac{\ell}{2} \rceil$. While the majority of trees we looked at meet this bound, we found that some require a slightly greater width. We use our main results to prove this.

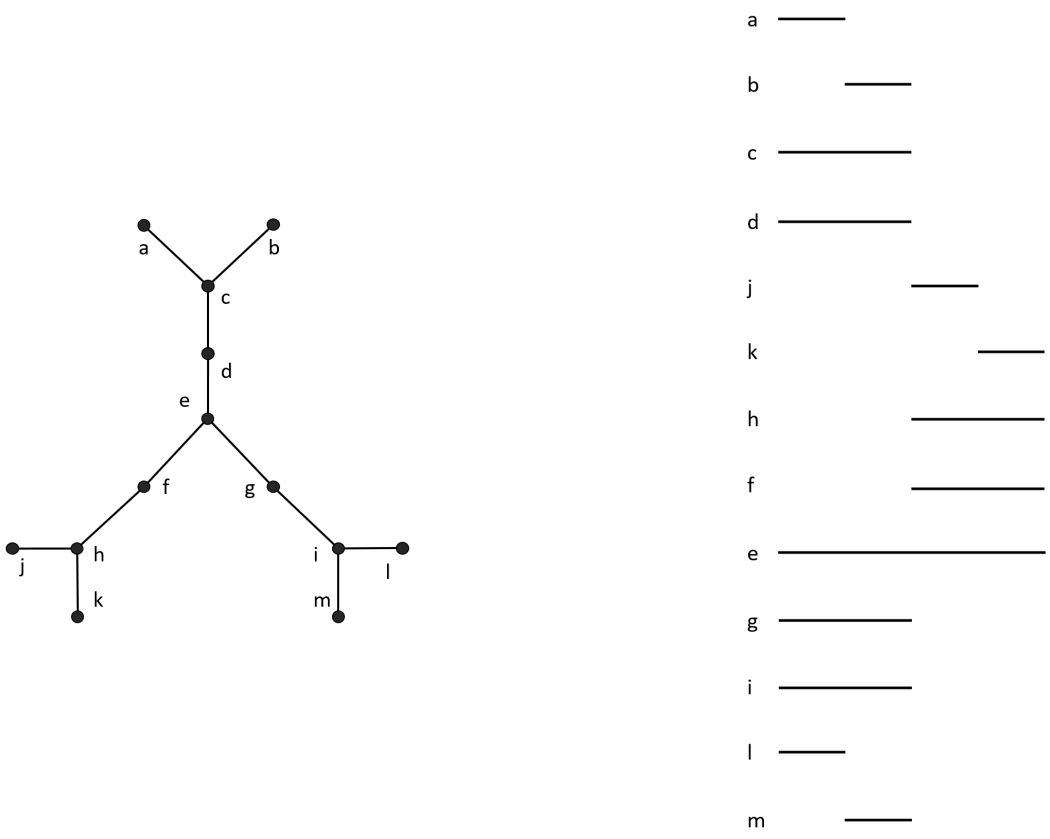


Figure 17: A tree with 6 leaves and width 4.

Theorem 7.1. *There are trees with ℓ leaves and width greater than $\lceil \frac{\ell}{2} \rceil$.*

Proof. Consider the tree in Figure 17. Note that this tree contains 6 leaves and is represented with a width 4 bar visibility graph. Suppose we could represent it with a width 3 BVG. Then by Corollary 6.2 we can find 3 coherently directed paths whose union is an acyclic orientation of this tree. In order to minimize the number of paths, we may assume that each path is maximal. Notice that for the paths to be coherently directed, the vertices a and b must either both be heads or both be tails. Likewise with the pair of vertices j and k and with the pair m and l. This means we have either four heads or four tails. Either way, we require at least four paths in order to build this tree with coherently directed paths. □

7.2 Graphs with Degree Constraints

We saw in Lemma 4.2 and Theorem 4.3 that we can use high degree vertices to force the bar visibility representation of a graph to be arbitrarily wide. Similarly, in Lemma 4.1 and Corollary 5.2 we saw that increasing the number of leaves in a graph forces an increase in the width of a graph. Here we focus on creating arbitrarily wide graphs by first eliminating high degree vertices, and then eliminating leaves as well.

We begin by restricting the maximum degree of a graph to 4. In the following result, we utilize leaves to create arbitrarily wide graphs.

Theorem 7.2. *Let k be any nonzero positive integer. Then there exists a graph G with $\Delta(G) = 4$ and $w(G) = k$.*

Proof. Consider the family of graphs in Figure 18. Notice that for each graph, there are $2k$ leaves, so by Lemma 4.1 $w(G) \geq k$. We can see from Figure 18 that each graph is representable with a width k BVG.

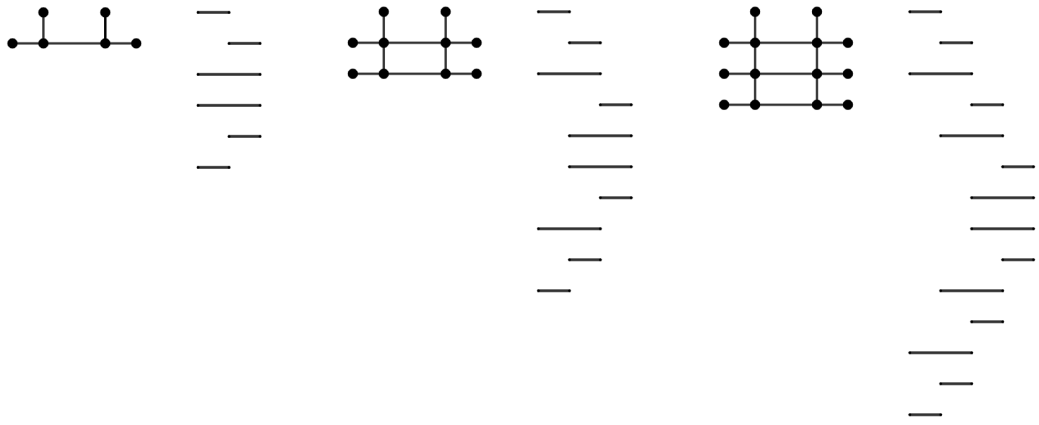


Figure 18: A family of graphs with $\Delta(G) = 4$ and arbitrary width.

□

In the following result, we construct a family of graphs that eliminates both high degree vertices and leaves. The degree of each vertex in these graphs is either 2 or 3. We show that we can find a graph in this family with width greater than any given width.

Theorem 7.3. *Let k be any nonzero positive integer. Then there exists a connected graph G with no leaves and $\Delta(G) = 3$ with width $w(G) \geq k$.*

Proof. Consider the family of graphs in Figure 19. If we think about the criteria in Corollary 6.2 for the width of a given graph, we want to show that given a width k, a graph in this family requires at least k coherently directed paths. Let us first consider using undirected paths. Notice that each 3-cycle requires two paths, since a path cannot start and end at the same vertex (this would be a cycle). In creating maximal paths, each path can include edges from one 3-cycle at one end and another 3-cycle at the other end. Thus, in the best case scenario, we can use two paths for every two 3-cycles in the graph. This means we need at least one path per 3-cycle to construct G. When looking at coherently directed paths, we need at least this many paths, if not more, to create an acyclic orientation of G. So, given a width k, a graph in this family with k 3-cycles will require at least k coherently directed paths. Thus the width of such a graph is at least k.

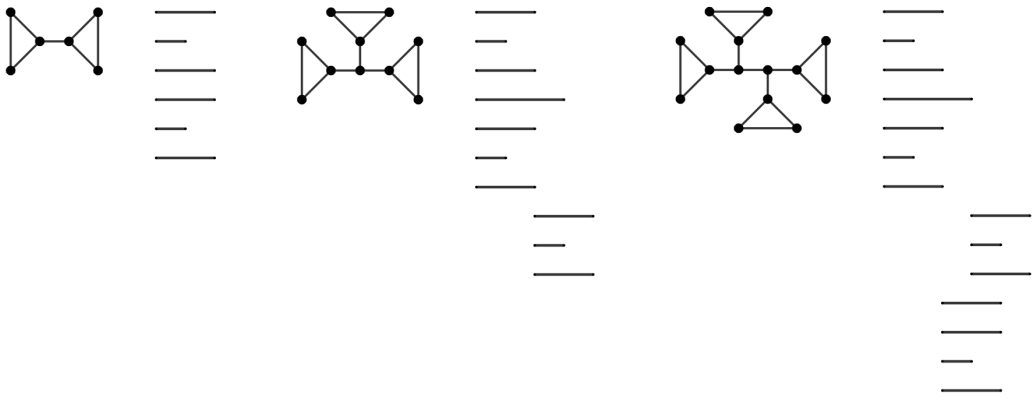


Figure 19: A family of graphs with $\Delta(G) = 3$, no leaves, and arbitrarily large width.

□

8 Future Work

In our initial work with BVGs, we found that the empty graph E_n on n vertices has width n. This follows directly from Corollary 4.5. We have checked all graphs on up to six vertices and have yet to find a connected graph with width greater than n, so we present the following conjecture on the upper bound of $w(G)$ for a connected graph G.

Conjecture 8.1. *Given a connected bar representable graph G on n vertices, $w(G) \leq n$.*

Limiting high degree vertices and leaves led us to ask whether a graph where every vertex has the same degree could also be arbitrarily wide.

Conjecture 8.2. *For any $k \geq 3$ there exists a family of k-regular connected bar representable graphs with arbitrarily large width.*

We think such a family of graphs might be of the form of the graph in Figure 20.

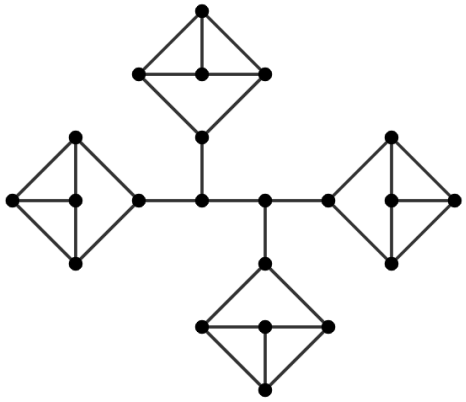


Figure 20: A potential family of 3-regular graphs with arbitrarily large width.

References

[1] D. Cvetkovic, M. Petric. A Table of Connected Graphs of Six Vertices. *Discrete Mathematics*, 50:37-49, 1984.

[2] M. Garey, D. Johnson, and H. So. An Application of Graph Coloring to Printed Circuit Testing. *IEEE Transactions on Circuits and Systems*, CAS23(10):591-599, 1976.

[3] G. Kant, G. Liotta, R. Tamassia, and I. Tollis. Area Requirement of Visibility Representations of Trees. *Information Processing Letters*, 62(2):81-88, 1997.

[4] R. Tamassia and I. Tollis. A Unified Approach to Visibility Representations of Planar Graphs. *Discrete & Computational Geometry*, 1(4):321-341, 1986.

[5] D.B. West. *Introduction to Graph Theory*, 2nd edition. Prentice Hall. 2001.

[6] S.K. Wismath. Characterizing bar line-of-sight graphs. *In Proceedings of the First Annual Symposium on Computational Geometry (SCG '85)*. Association for Computing Machinery, New York, NY USA, 147-152.

Appendix

The following is the code we used to generate all possible BVGs on a given number of vertices and with a given width.

```
from itertools import product
import numpy as np

vert = int()        # number of vertices
width = int()       # max width of bvgs
barset = set()      # list of possible bars given width (ordered pairs)
B = list()          # list of bvgs given vert and barset
D = list()          # list of bvgs with associated info (deg seq, traces, etc.)

# get number of vertices
vert = input('# vertices: ')
vert = int(vert)

# get maximum width
width = input('width: ')
width = int(width)

# create all possible bars given vert and add them to barset
for x_1 in range(width+1):
    for x_2 in range(x_1+1,width+1):
        bar = (x_1,x_2)
        barset.add(bar)

# create all possible bvgs on vertices from list of bars
B = list(z for z in product(barset, repeat = vert))
print(len(B))
```

We then created visual representations of the BVGs using matrices.

```
def grid(bvg):
    Grid = np.zeros((vert,width), dtype=int)
    for x_1 in range(vert):
        for x_2 in range(bvg[x_1][0],bvg[x_1][1]):
            Grid[x_1][x_2] = x_1+1
    return Grid
```

We used these matrices to generate the adjacency matrices, degree sequences, and traces in order to identify the graph that each BVG was representing.

```
# creates the adjacency matrix of a given bvg
def adj(Grid):
    A = np.zeros((vert,vert), dtype=int)
    # walks through columns of grid, deleting 0s
    for y in range(width):
```

```
        col = Grid[:,y]
        ncol = col[col != 0]
        # puts 1 in adjacency matrix for every pairwise adjacent vertex/bar
        for i in range(len(ncol)-1):
            A[ncol[i]-1][ncol[i+1]-1] = 1
            A[ncol[i+1]-1][ncol[i]-1] = 1
    return A

# generate degree sequences
def degseqlist(B):
    degseqs = set()
    start = int(input('start: '))
    stop = int(input('stop: '))
    # this loop walks through each bvg in the list B from index 'start'
    # to index 'stop'
    # 1. finds adjacency matrix adj(grid(B([x_1])))
    # 2. sums each row to create a deg seq (np.sum)
    # 3. sorts that deg seq (np.sort)
    # 4. turns the deg seq into a list (np.ndarray.tolist)
    # 5. inserts that list as the second entry in a tuple
    # (first entry is the associated bvg from the original list B)
    for i in range(start,stop):
        d = np.ndarray.tolist(np.sort(np.sum(adj(grid(B[i])), axis=0)))
        D[i][1] = d
        if i % 100 == 0:
            print(i)
        degseqs.add(str(d))
    degseqs = list(degseqs)
    degseqs.sort()
    for x in range(len(degseqs)):
        print(degseqs[x])

# find trace of A^2, A^3, A^4 of a given bvg
def trace(bvg):
    A = adj(grid(bvg))          # adjacency matrix
    A2 = np.matmul(A,A)         # A^2
    A3 = np.matmul(A2,A)        # A^3
    A4 = np.matmul(A3,A)        # A^4
    traces = [np.trace(A2), np.trace(A3), np.trace(A4)]
    return traces
```