CYCLEWIDTH: AN ANALOGY OF TREEWIDTH

NICOLE WEIN
Advisor: Glencora Borradaile

Abstract. In this work we study cyclewidth, a generalization of treewidth and pathwidth to cycles. Although treewidth and pathwidth have been extensively studied, cyclewidth has only been mentioned in a single paper, to the best of the author's knowledge. We prove that the decision problem of determining whether the cyclewidth of a graph is at most k is NP-complete. We also present an algorithm for finding the cyclewidth of cactus graphs and prove that the pathwidth of a given cactus graph is at most one greater than its cyclewidth. Additionally, we characterize edge-maximal graphs of pathwidth k and leave the problem of recognizing them unsolved.

1 Introduction

Treewidth was introduced independently by Halin [9] and Robinson and Seymour [18], and pathwidth was introduced by Robertson and Seymour [17]. One primary value of treewidth and pathwidth is that many NP-hard problems can be solved in polynomial time on graphs of bounded treewidth or pathwidth. Treewidth and pathwidth have also proved useful in electrical networks [3], graph drawing [16], and searching games [6]. We generalize treewidth and pathwidth to a concept we call cyclewidth. As far as the author can tell, cyclewidth only appears in a single paper, by Sharp and Kozen [14]. Sharp and Kozen use cyclewidth peripherally to prove the following result about graph coloring: (d, k)-colorable graphs for $k \leq \lfloor 3d/2 \rfloor$ have cyclewidth at most d. We investigate the properties of cyclewidth centrally, especially in relation to treewidth and pathwidth.

Notation. Let G be an graph with vertex set $V(G)$ and edge set $E(G)$. All graphs are finite, undirected, and unweighted.

Notation. Let $A \uplus B$ denote the disjoint union of A and B.

Definition 1.1. *A* decomposition (H, X) *of a graph G consists of an overlay graph H and a family of subsets $X = \{X_i | i \in V(H)\}$ of $V(G)$ satisfying the following properties.*

(i) Vertex containment*: $V(G) = \cup_i X_i$.*

(ii) Edge containment*: for all edges $uv \in E(G)$, there exists $i \in V(H)$ such that $u, v \in X_i$.*

(iii) Vertex connectivity*: for all $v \in V(G)$, the subgraph of H induced by $\{i \in V(H) | v \in X_i\}$ is connected.*

For clarity, we will refer to vertices *of G and* nodes *of H. We will refer to X as* bags.

A *path* is a walk with no repeated vertices, a *cycle* is a closed path, and a *tree* is a connected acyclic graph. *Tree decompositions*, *path decompositions*, and *cycle decompositions* are decompositions in which the overlay graph is a tree, a path, and a cycle, respectively.

Definition 1.2. *The* width *of a decomposition is* $\max_i |X_i| - 1$. *The* treewidth, pathwidth, *and* cyclewidth *of a graph G is the minimum width over all tree, path, and cycle decompositions of G, respectively.*

In Section 2, we prove a collection of lemmas about decompositions, with a focus on cycle decompositions. In Section 3, we prove that the decision problem of whether the cyclewidth of a graph is at most k in NP-complete. In Section 4, we give a polynomial time algorithm for the cyclewidth of cacti and prove that if G is a cactus, $\text{pathwidth}(G) - \text{cyclewidth}(G) \in \{0, 1\}$. In Section 5, we characterize edge-maximal graphs of cyclewidth k. We leave unsolved the problem of recognizing edge-maximal graphs of cyclewidth k, but provide speculation in Appendix A.

2 Cycle Decompositions

Notation. Let G be a graph with decomposition (H, X). For $S \subseteq V(G)$, let $G[S]$ denote the subgraph of G induced by S, let $X_i[S]$ denote $X_i \cap S$ for all $i \in V(H)$ and let $H[S]$ denote $\{i \in V(H) | X_i[S] \neq \varnothing\}$.

Observe that if G is a graph with decomposition (H, X), then for all $S \subseteq V(G)$, $(H[S], X[S])$ is a valid decomposition.

Lemma 2.1. *If G is a connected graph and (H, X) is a decomposition of G, then H is connected.*

Proof. Equivalently, we will show that for all $u, v \in V(G)$ and all $i, j \in V(H)$, if $u \in X_i$ and $v \in X_j$, then i and j are connected in H. Let $d(u, v)$ denote the distance between u and v. The following is a proof by strong induction on $d(u, v)$.
Base Case: $d(u, v) = 1$. Let h be a node in H such that $u, v \in X_h$. By edge containment of (H, X), such a node exists. By vertex connectivity of (H, X), $H[\{u\}]$ and $H[\{v\}]$ both form connected components. Then, since $h \in (H[\{u\}] \cap H[\{v\}])$, $H[\{u\}] \cup H[\{v\}]$ forms a connected component.
Inductive Hypothesis: Suppose that if $u \in X_i$, $v \in X_j$, and $d(u, v) \leq k$, then i and j are connected in H.
Inductive Step: Suppose $d(u, v) = k + 1$. Choose $w \in V(G)$ on the shortest path between u and v. Then $d(u, w) \leq k$ and $d(v, w) \leq k$. By the inductive hypothesis, if $u \in X_i$, $v \in X_j$, and $w \in X_l$ then i and l are connected and j and l are connected. Then, by transitivity of connectivity, i and j are connected. □

Let G be a graph with decomposition (H, X), let j be a node in $V(H)$, let $H' = H \backslash j$, and for all $i \in V(H')$ let $X'_i = X_i \cup X_j$. We say (H', X') is the result of *distributing* i *over* (H, X). Observe that if H' is connected, then (H', X') is a valid decomposition of G.

Lemma 2.2. *For any graph* G, $\frac{1}{2}\,\text{pathwidth}(G) \leq \text{cyclewidth}\, G \leq \text{pathwidth}(G)$

Proof. Given a cycle decomposition (C, X) of G, construct a path decomposition (P, X') of G by distributing any node $i \in V(C)$ over (C, X). $\text{width}((P, X')) = \text{width}((C, X)) + |X_j| \leq 2(\text{width}((C, X)))$, so $\frac{1}{2}\,\text{pathwidth}(G) \leq \text{cyclewidth}\, G$.

Given a path decomposition (P, X) of G, construct a cycle deomposition (C, X) of G by adding an edge between the two endpoints of P. $\text{width}((C, X)) = \text{width}((P, X))$, so $\text{cyclewidth}(G) \leq \text{pathwidth}(G)$. □

An *interval graph* is a graph for which each vertex can be associated with an interval on a line such that two intervals overlap if and only if their associated vertices share an edge. The *interval thickness* of an interval graph is the maximum number of intervals in its interval representation that contain the same point on the line. The interval thickness of a graph G is the minimum interval thickness over all interval graphs $G' \supseteq G$.

A *circular arc graph* is a graph for which each vertex can be associated with an arc on a circle such that two arcs overlap if and only if their associated vertices share an edge. The *arc thickness* of a circular arc graph is the maximum number of arcs in the circular arc representation that contain the same point on the circle. The arc thickness of a graph G is the minimum arc thickness over all circular arc graphs $G' \supseteq G$.

Note that the interval representation of an interval graph is unique while the circular arc representation of a circular arc graph is not.

Notation. In a circular arc representation of a graph G, arbitrarily number the vertices of G from 0 to $|V(G)| - 1$, let $A[i]$ denote the arc corresponding to the i^{th} vertex, and for all i, let $A[i]_s$ and $A[i]_t$ be the *starting* (counterclockwisemost) and *ending* (clockwisemost) points of $A[i]$, respectively.

A circular arc graph can be completely characterized by a *circular arc representation list*, a list containing $A[i]_s$ and $A[i]_t$ for all i in order of appearance starting from an arbitrary point on the circle and moving clockwise.

Lemma 2.3. *(Theorem 29 [1]) The pathwidth of a graph* G *is at most* $k-1$ *if and only if the interval thickness of* G *is at most* k.

The following is a construction to convert between a graph G with width-$k-1$ cycle decomposition (C, X) and circular arc graph $G' \supseteq G$, with $V(G') = V(G)$ and arc thickness k. It is based on Bodlaender's proof of Lemma 2.3 [1].

Given (C, X), associate to each $v \in V(G)$ an arc beginning at the counterclockwisemost bag containing v and ending at the clockwisemost bag containing v. The result is a circular arc graph $G' \supseteq G$ with $V(G') = V(G)$ and arc thickness k. Note that $G = G'$ if and only if every bag in (C, X) is a clique.

Given a circular arc representation of $G' \supseteq G$ with $V(G') = V(G)$, let a_v be the arc corresponding to the vertex $v \in V(G')$. Construct one bag at each endpoint of each arc on the circle and connect the bags in circular order. For all $v \in V(G')$, add v to every bag that shares a point on the circle with a_v. The result is a width-$k-1$ cycle decomposition of G. Thus, we have,

Lemma 2.4. *The cyclewidth of a graph G is at most $k-1$ if and only if its arc thickness is at most k.*

Now we consider the effect of cliques on decompositions.

Lemma 2.5. *(Lemma 4 [1]) If (T, X) is a tree decomposition of a graph G and $K \subseteq G$ is a clique, then there exists $i \in V(T)$ such that $K \subseteq X_i$.*

Lemma 2.6. *Let (C, X) be a cycle decomposition of a graph G and let $K_n \subseteq G$ be a clique on n vertices. Then, for all $i \in V(C)$, there exists $j \in V(C)$ such that and $X_i[K_n] \cup X_j[K_n] = V(K_n)$.*

Proof. Fix $i \in V(C)$ and let $S = V(K_n) \backslash X_i[K_n]$. By Lemma 2.1, since S is connected, $C[S]$ must be connected. Then since $X_i \cap S = \varnothing$, $(C[S], X[S])$ must be a path decomposition. By Lemma 2.5, since S is a clique, there exists $j \in V(C)$ such that $S \subseteq X_j$. Thus, $X_j[K_n] \subseteq S$, so $X_i[K_n] \cup X_j[K_n] = V(K_n)$. □

Lemma 2.7. *Let K be a clique with cycle decomposition (C, X). Then* $\text{width}((C, X)) \geq \max_{i,j \in V(C)} |X_i \backslash X_j|$

Proof. Let i and j be arbitrary nodes in C. Let $X' = X[K \backslash (X_i \cap X_j)]$. Moving clockwise in C from j, let j_1 be the first node for which $X'_j \neq X'_{j_1}$ and let j_2 be the counterclockwise neighbor of j_1. Let v be a vertex in K such that $v \in (X'_j \cap X'_{j_2}) \backslash X'_{j_1}$. Vertex v must only appear in bags corresponding to nodes on the counterclockwise path P in C between between j_2 and i. By edge containment of (C, X), for all $u \in X'_i$, there must exist a node $i_u \in V(P)$ such that $u, v \in X'_{i_u}$. By vertex connectivity of (C, X), for all $u \in X'_i$, for every node l on the clockwise path in C from i to i_u, $u \in X'_l$. Let w be the vertex in X'_i for which i_w appears first on the counterclockwise path in C from i. Then, $X'_i \cup \{v\} \subseteq X'_{i_w}$, so $|X_{i_w}| \geq |X'_i| + 1 = |X_i \backslash X_j| + 1$, so $\text{width}((C, X)) \geq |X_i \backslash X_j|$. □

Lemma 2.8. *The largest clique with cyclewidth at most k is a $2k+1$-clique.*

Proof. Let K be a $2k+2$-clique with cycle decomposition (C, X). Suppose, by way of contradiction, that $\text{width}(C, X) \leq k$. By Lemma 2.6, if there exists a node g such that $|X_g| \leq k+1$, then there exists a node h such that $|X_h| \geq k+1$, so for all $i \in V(C)$, $|X_i| = k+1$. Then, for all $j \in V(C)$, there exists a node $l \in V(C)$ such that $X_j \cap X_l = \varnothing$. Fix j and l. By Lemma 2.7, $\text{width}((C, X)) \geq \max(|X_j|, |X_l|) = k+1$, a contradiction.

Let K' be a $2k+1$-clique. We will describe the construction a width-k cycle decomposition (C', X') of K'. Each X'_i must be a clique, so we will recognize (C', X') as a circular arc graph. Construct a circular arc representation list of K' by iteratively appending $A[i]_s$ followed by $A[(i+k+1) \mod (2k+1)]_t$ to the list for i from 0 to $2k$. The resulting circular arc graph corresponds to a width-k cycle decomposition of K'. □

Lemma 2.9. *Let G be a graph with cyclewidth k and $|V(G)| \geq 2k+2$. G may contain a $2k$-clique, but not a $2k+1$-clique.*

Proof. Let K be a $2k+1$ clique. By Lemma 2.8, cyclewidth$(K) = k$. Subdivide a single arc of a thickness-$k+1$ circular arc representation of K into $p > 2$ pieces. The result represents a graph with cyclewidth k, at least $2k+2$ vertices, and a $2k$ clique. Thus, G may contain a $2k$-clique.

Suppose, by way of contradiction, that G contains a $2k+1$-clique K'. Let (C, X) be a width-k cycle decomposition of G. We claim that for all $i \in C$, $|X_i[K']| = k+1$. As a consequence, any proper supergraph of a $2k+1$-clique has cyclewidth at least $k+1$, a contradiction.

Proof of claim: Suppose by way of contradiction there exists $i \in C$ such that $|X_i[K']| \leq k+1$. Fix i. By Lemma 2.6, there exists $j \in C$ such that $|X_j \backslash X_i| \geq 2k+1-(k+1) = k$. Then, by Lemma 2.7, width$((C, X)) \geq k+1$. □

Definition 2.1. *Let G be a graph with cycle decomposition (C, X) and let G' be a subgraph of G. We say G'* spans *(C, X) if $H[G'] = H$.*

Lemma 2.10. *Let (C, X) be a cycle decomposition of a graph G and let $K \subseteq G$ be a clique. Then either K spans (C, X) or there exists $i \in V(C)$ such that $V(K) \subseteq X_i$.*

Proof. Suppose K does not span (C, X). Then $(C[K], X[K])$ is a path decomposition, so by Lemma 2.5, there exists $i \in V(C)$ such that $V(K) \subseteq X_i$. □

Let G and G' be graphs with decompositions (H, X) and (H', X') and let F and F' be isomorphic subgraphs of H and H', respectively. For all $i \in V(H \backslash F)$, let $X''_i = X_i$ and for all $i \in V(H' \backslash F')$, let $X''_i = X'_i$. For all $i \in V(F)$, identify i with its corresponding node $i' \in V(F')$ and let H'' be the resulting graph. For all $i \in V(F)$, let $X''_i = X_i \cup X'_{i'}$. We say that (H'', X'') is the result of *merging* (H, X) at F with (H', X') at F'. Observe that if G and G' are disjoint, (H'', X'') results in a valid decomposition of $G \uplus G'$.

Lemma 2.11. *Let G be a graph with a cycle decomposition (C, X) of width k with no spanning cycle. Then,* pathwidth$(G) \leq k$.

Proof. If G is disconnected, each of its connected components can be considered separately, so suppose without loss of generality, that G is connected. Consider an arbitrary cycle $C_1 \in G$. Let $\{G_i\}$ be the set of connected components of $G \backslash C_1$. Let (P, X) be the result of removing an edge gh from C such that at most one of $\{X_g, X_h\}$ contain at least one vertex in C_1. Since C is not spanning, gh exists.

For each G_j, construct a path decomposition (P^j, X^j) of $C_1 \uplus G_j$ as follows. Let $S_j = V(G_j) \cap X_g \cap X_h$. If S_j is empty, then let $(P^j, X^j) = (P, X)$. Otherwise, for all $v \in S_j$, $P[\{v\}]$ is the disjoint union of two paths P_{v1} and P_{v2} because if $P[\{v\}]$ were connected, then all $v \in S_j$ would trivially be spanning cycles. For all $v \in S_j$, replace all occurrences of v in $\{X_p | p \in V(P_{v1})\}$ with a new vertex v_1, replace all occurrences of v in $\{X_p | p \in V(P_{v2})\}$ with a new vertex v_2, and let (P'^j, X'^j) be the result. For all $v \in S_j$, add edges $v_1 w$ and $v_2 w$

to $((C_1 \uplus G_j)\backslash\{v\}) \uplus \{v_1\} \uplus \{v_2\}$ if $vw \in E(G)$ and there exists i such that $v_1, w \in X'_i$ and $v_2, w \in X'_i$, respectively. Let G' be the resulting graph. G' must be disconnected because if G' were connected, there would exist a path between v_1 and v_2, so (C, X) would have a spanning cycle. Let G'_1 be a connected component of G' for which G'_1 is connected to C_1. The other connected component G'_2 of G' must be disconnected from C_1 because otherwise, (C, X) would have a spanning cycle. Orient P'^j linearly so that its leftmost node contains vertices in G'_2 and its rightmost node contains vertices in G'_1. Let (P''^j, X''^j) be the result of adding an edge between the leftmost node of $(P'^j[G'_2], X'^j[G'_2])$ and the rightmost node of $(P', \{X'_i\backslash G'_2\})$. Let (P^j, X^j) be the result of replacing every occurrence of v_1 or v_2 in X''^j with v.

For all i, $(P^i[C_1], X^i[C_1]) = (C[C_1], X[C_1])$. For each i, orient P^i linearly so that all $(P^i[C_1], X^i[C_1])$ are oriented consistently. Let (P'', X'') be the result of merging all (P^i, X^i) at $P^i[C_1]$, respectively. We claim that (P'', X'') is a valid path decomposition of width at most k.

By construction, (P'', X'') satisfies vertex containment. Consider $\uplus_j S_j$ and $G\backslash(\uplus_j S_j)$ separately. By construction, vertex connectivity holds for all $v \notin \uplus_j S_j$ and edge containment holds for all uv such that $u, v \notin \uplus_j S_j$. Consider an arbitrary vertex $v \in S_j$ for some j. For all $u \in V(G)$, if $uv \in E(G)$, then uv_1 or uv_2 in (P'^j, X'^j) and (P''^j, X''^j). Replacing v_1 and v_2 with v to form (P^j, X^j) ensures that edge containment holds for uv. One endpoint of P'^j corresponds to a bag containing v_1 and the other endpoint corresponds to a bag containing v_2, so constructing (P''^j, X''^j) and replacing v_1 and v_2 by v to form (P^j, X^j) ensures vertex connectivity for v. Since every vertex belongs to C_1 or exactly one G_i, merging all (P^j, X^j) does not violate vertex connectivity.

By construction, for all j every bag in (P^j, X^j) is a subset of a bag in (C, X). Since all (P^i, X^i) are merged with $P^i[C_1]$ aligned, every bag in (P'', X'') is also a subset of a bag in (C, X). Thus, $\text{width}((P'', X'')) \leq k$. □

From Lemmas 2.2 and 2.11 we get:

Corollary 2.12. *Let T be a tree. Then* $\text{cyclewidth}(T) = \text{pathwidth}(T)$.

Lemma 2.13. *(Theorem 3 [19]) There is a linear time algorithm to determine the pathwidth of a tree.*

From Corollary 2.12 and Lemma 2.13 we get:

Corollary 2.14. *There is a linear time algorithm to determine the cyclewidth of a tree.*

3 The Cyclewidth Problem is NP-Complete

Let PWP (Pathwidth Problem) be the decision problem of whether the pathwidth of a graph is at most k. It is known that PWP is NP-complete [13]. Let CWP (Cyclewidth Problem) be the decision problem of whether the cyclewidth of a graph is at most k.

Theorem 3.1. *CWP is NP-complete*

Proof. We begin by showing that CWP is in NP. The solution to a "yes" instance of CWP can be represented by a cycle decomposition of width k. To check that this decomposition is valid, we must verify the three properties in the definition of decomposition, which takes polynomial time.

We establish a reduction from PWP to CWP as follows. Consider an arbitrary instance of PWP on a graph G. Let K be a $k+1$-clique disjoint from G and let $G' = G \cup K$. Consider CWP on G'.

We now establish that the reduction is correct. Consider an arbitrary instance I of PWP and its corresponding instance I' of CWP. We will show that I is a "yes" instance if and only if I' is also a "yes" instance.

Suppose I is a "yes" instance. Then G has a path decomposition (P, X) of width k. Construct a cycle decomposition (C, X') of G' as follows. Let H' be the result of inserting an edge from each end of the path P to an additional node j. Let $X'_i = X_i$ for all $i \in V(P)$ and let $X_j = V(K)$. (C, X') is a valid width-k cycle decomposition of G', so I' is a "yes" instance.

Suppose I' is a "yes" instance. Then G' has a cycle decomposition (C, X') of width k. Figure 1 shows a possible (C, X') with the following variables labeled.

$$
\begin{aligned}
l &= \text{argmin}_i(|X'_i[K]|) \\
h &= \text{argmin}_i(|X'_i[G]|) \\
k_{min} &= |X'_l[K]| \\
g_{min} &= |X'_h[G]| \\
k_{max} &= |\max_i(|X'_i[K]|)| \\
g_{max} &= |\max_i(|X'_i[G]|)|
\end{aligned}
$$

Since $max_i|X'(i)| = k+1$,

$$g_{max} \leq k + 1 - k_{min} \tag{1}$$

and

$$g_{min} \leq k + 1 - k_{max} \tag{2}$$

Let $S = V(K) \backslash X'_l$. By Lemma 2.6, there exists $j \in V(C)$ such that $S \subseteq X'_j$. $k_{min} = k + 1 - |S|$ so,

$$k_{min} \geq k + 1 - k_{max} \tag{3}$$

Combining (2) and (3),

$$g_{min} \leq k_{min} \tag{4}$$

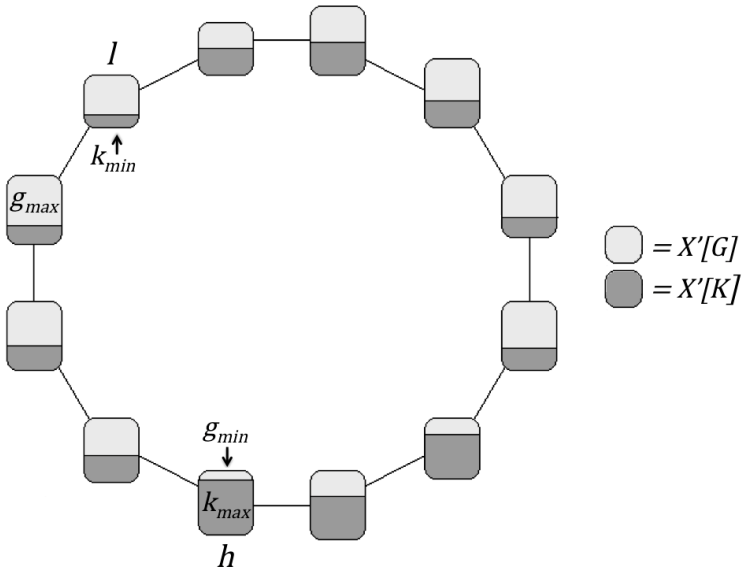

Figure 1: A possible division of (C, X') based on the numbers of vertices in each bag of $X'[K]$ and $X'[G]$.

Let path decomposition (P, X) of G be the result of distributing a h over (C, X'). Calculate the width of (P, X) by the following.

$$\begin{aligned} max_i|X_i| &= g_{min} + g_{max} \\ &\leq k_{min} + k + 1 - k_{min} \qquad \text{by (1) and (4)} \\ &= k + 1 \end{aligned}$$

Thus, width$((P, X)) \leq k$, so I is a "yes" instance. □

A natural next inquiry concerns approximation schemes for cyclewidth. The best known approximation ratio of a polynomial time approximation algorithm for pathwidth is $O((\log n)^{\frac{3}{2}})$ [5], so by Lemma 2.2, this is also the case for cyclewidth.

4 A Polynomial Time Algorithm for the Cyclewidth of Cacti

A *cactus* is connected graph in which every edge belongs to at most one cycle.

Lemma 4.1. *(Corollary 7.4 [2]) The pathwidth of a cactus can be determined in polynomial time.*

Bodlaender and Kloks list cacti among a group of graph classes for which pathwidth can be determined in polynomial time [2]. The order of the polynomial for cacti, specifically, is not stated. Let $f(n)$ denote the function for which the pathwidth of a cactus can be determined in $\theta(f(n))$.

Let G be a cactus and let $C_1, \dots C_n$ be a set of cycles in G. Define the *components with respect to* $\{C_1, \dots C_n\}$ as the set of connected components of $G \backslash (\cup_i C_i)$.

Let (H, X) be a decomposition with adjacent nodes i and j. Let *stretching* (H, X) *between* i *and* j *by* k be the process of subdividing ij with k nodes and letting each correspond to the bag $X_i \cup X_j$. Observe that stretching a decomposition between any two adjacent nodes results in a valid decomposition.

We now introduce a method to construct a cycle decomposition of a cactus G with exactly one spanning cycle C_1.

Construction 4.1. Let $\{G_i\}$ be the set of components of G with respect to C_1. By the definition of cactus, for every component G_i, C_1 contains exactly one vertex w_i adjacent to G_i. We will build a decomposition of G by iteratively adding the vertices of each component to a constantly updated decomposition (C, X), initialized to a width 1 cycle decomposition of C_1. For each component G_i, compute a minimum-width path decomposition (P, X') of G_i. Let l_i be $|V(H')|$. Let h and j be two adjacent nodes in (C, X) such that $X'_h \cap X'_j = w_i$. Stretch (C, X) between h and j by l_i and let Q be the path of added nodes. Merge (P, X') at P with (C, X) at Q. This construction can be completed $f(n)$ time.

Observe that Construction 4.1 yields a valid cycle decomposition. Furthermore, observe that if a cycle decomposition (C, X) of G is constructed as in Construction 4.1, for every vertex $v \in V(C_1)$, there exist adjacent nodes $i, j \in V(C)$ such that $X_i \cap X_j = \{v\}$.

Lemma 4.2. *The result of Construction 4.1 is a minimum-width cycle decomposition* (C, X) *of* G *over all cycle decompositions with spanning cycle* C_1*.*

Proof. Let (C', X') be a minimum-width cycle decomposition of G over all cycle decompositions with spanning cycle C_1. No component G'_i with respect to C' contains a spanning cycle, so by Lemma 2.11, $\text{pathwidth}(G'_i) \leq \text{width}((C'[G_i], X'[G_i]))$. Thus,

$$\begin{aligned}
\text{width}((C', X')) &\geq \text{cyclewidth}(C_1) + \max_i(\text{width}((C'[G_i], X'[G_i])))\\
&\geq 1 + \max_i(\text{pathwidth}(G_i))\\
&= \text{width}((C, X))
\end{aligned}$$

□

Lemma 4.3. *Every cactus has a minimum-width cycle decomposition with at most one spanning cycle.*

Proof. Suppose a cactus G has a minimum-width cycle decomposition (C, X) with at least two spanning cycles. Let $\{C_i\}$ be the set of cycles in G, let $\{G_{ij}\}$ be the set of components with respect to C_i, and let v_{ij} be the vertex in $V(C_i)$ adjacent to G_j. Choose C_a and C_b to be spanning cycles in (C, X) such that there exists exactly one j such that $G_{aj} \uplus v_{aj}$ is spanning and there exists exactly one k such that $G_{bk} \uplus v_{bk}$ is spanning. Fix j and k. We

will construct a cycle decomposition (C', X') of G with width at most width$((C, X))$ such that neither C_a nor C_b is spanning.

Let $S_a = G \backslash G_{aj}$ and let $S_b = G \backslash G_{bk}$. First, we construct a path decomposition of $S_a \cup S_b$. Use Construction 4.1 to construct a cycle decomposition (C'_a, Y) of S_a and a cycle decomposition (C'_b, Z) of S_b. Let i_a and i'_a be two adjacent nodes in C'_a such that $X_{i_a} \cap X_{i'_a} = \{v_{aj}\}$ and let i_b and i'_b be two adjacent nodes in C'_b such that $X_{i_b} \cap X_{i'_b} = \{v_{bk}\}$. Let (P_a, Y') be the result of stretching C'_a between i_a and i'_a by 1 and distributing the added node over (C'_a, Y). Let (P_b, Z') be the result of stretching C'_b between i_b and i'_b by 1 and distributing the added node over (C'_b, Z). Let (P_{ab}, Y'') be the result of adding an edge between one end of P_a and one end of P_b. Let *endpoint a* and *endpoint b* be the endpoints of P_{ab} that are in P_a and P_b, respectively.

Let $S = G_{aj} \cap G_{bk}$ and let $S' = S \uplus v_{aj} \uplus v_{bk}$. Figure 2 shows a cactus labeled with S and S' given a valid choice of C_a and C_b. If $S = \varnothing$, let (C', X') be the result of adding an

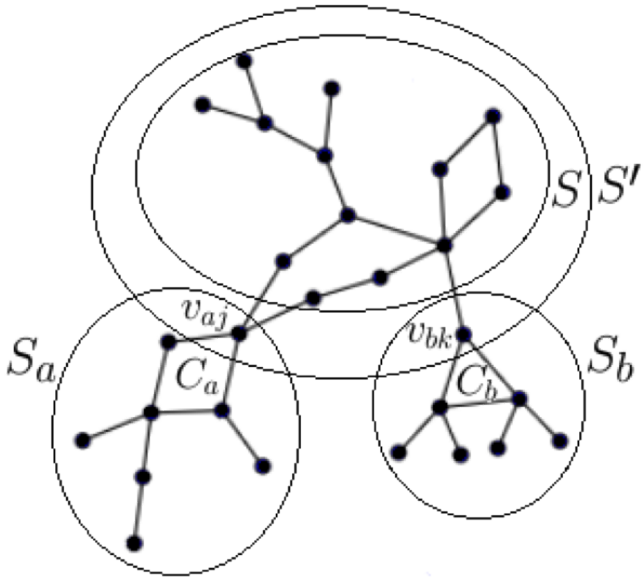

Figure 2: Subsets of the vertices of a cactus given a valid choice of C_a and C_b.

edge between endpoint a and endpoint b in P_{ab}. Otherwise, construct (C', X') as follows. Let $h = \text{argmin}(|X[S]|)$. Starting from h and moving counterclockwise in C, let a_{cc} and b_{cc} be the first nodes whose corresponding bags contain v_{aj} and v_{bk}, respectively. Similarly, starting from h and moving clockwise, let a_c and b_c be the first nodes whose corresponding bags contain v_{aj} and v_{bk}, respectively. Let (C'', X'') be the result of replacing h with a length $|V(P_{ab})|$ path P of nodes each corresponding to the bag X_h. Let (C', X') be the result of merging (P_{ab}, Y'') at P_{ab} with $(C''[S'], X''[S'])$ at P with P_{ab} oriented as follows. If a_{cc} appears before b_{cc} moving counterclockwise from h, orient (P_{ab}, Y'') so that endpoint a is the counterclockwisemost bag of (P_{ab}, Y''). In this case, add v_{aj} to every bag in (C', X') corresponding to a node in the counterclockwise path from the bag merged with endpoint

a to a_{cc} and add v_{bk} to every bag corresponding to a node in the clockwise path from the bag merged with endpoint b to b_c. Otherwise, orient (P_{ab}, Y'') such that endpoint a is the clockwisemost bag of (P_{ab}, Y''). In this case, add v_{aj} to every bag in (C', X') corresponding to a node in the clockwise path from the bag merged with endpoint a to a_c and add v_{bk} to every bag corresponding to a node in the counterclockwise path from the bag merged with endpoint b to b_{cc}. Figure 3 shows an example of this merging process.

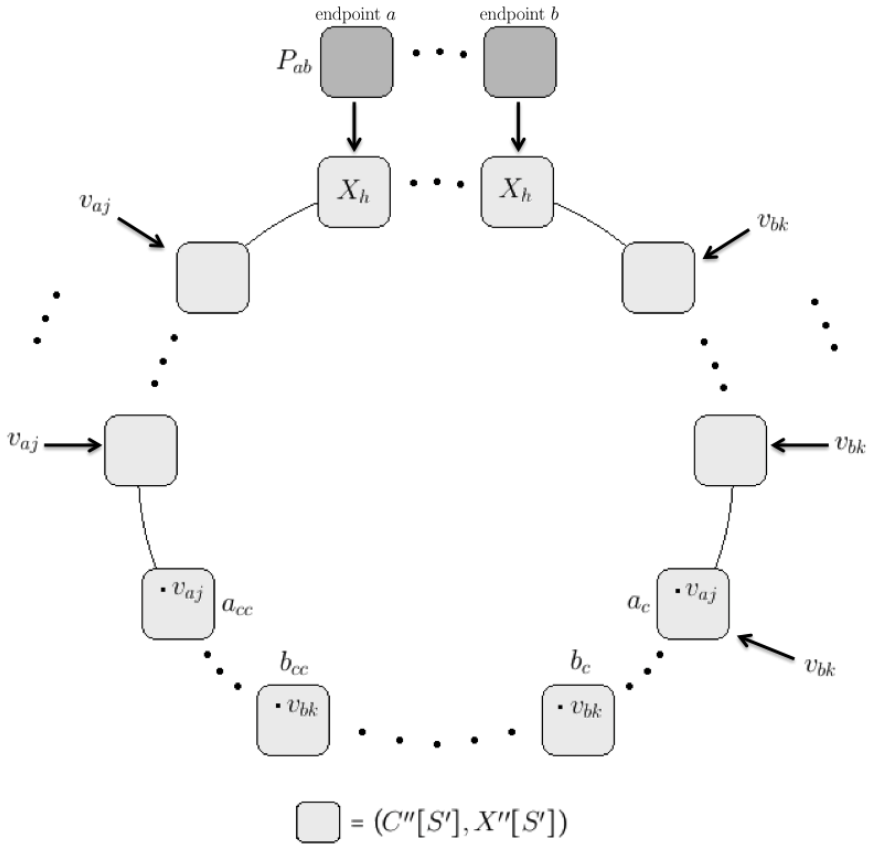

Figure 3: (C', X') formed from merging $(C''[S'], X''[S'])$ and P_{ab}, and adding v_{aj} and v_{bk} to certain bags.

We argue that neither C_a nor C_b is spanning in (C', X') unless $v_{aj} = v_{bk}$ and every bag contains v_{aj}. In this case, modify (C', X') by stretching between any pair of adjacent nodes by 1 and removing v_{aj} from the bag corresponding to the added node.

Now, we show that (C', X') is a valid decomposition. (C', X') is the result of merging a decomposition (P_{ab}, Y'') of $S_a \cup S_b$ with a decomposition $(C''[S'], X''[S'])$ of S'. $S' \cup S_a \cup S_b = G$, so (C', X') satisfies vertex containment and edge containment. Consider an arbitrary vertex $v \in V(G)$. If v is in exactly one of $\{S', S_a, S_b\}$ then vertex connectivity holds for v. Otherwise, v must be either v_{aj} or v_{bk}. After merging (P_{ab}, Y'') with $(C''[S'], X''[S'])$, we explicitly add v_{aj} and v_{bk} to a set of bags to ensure that the subgraph of C' induced by $\{i \in V(C') | v_{aj} \in X_i\}$ and $\{i \in V(C') | v_{bk} \in X_i\}$, respectively, are connected.

Now, we show that $\text{width}((C', X')) \leq \text{width}((C, X))$. Let $a_{max} = \text{argmax}(|X_i[S_a]|)$ and let $b_{max} = \text{argmax}(|X_i[S_b]|)$. If a_{max} contains v_{aj}, then either v_{aj} can be easily removed from $X_{a_{max}}$ or $|X_{a_{max}}[S_a]| = 2$, so we suppose, without loss of generality, that $v_{aj} \notin X_{a_{max}}$. Since C_b is spanning, $X_{a_{max}}$ must contain at least one vertex in C_b. Since $X_{a_{max}} \cap C_b = \varnothing$, $|X_{a_{max}}| \geq \min_i(|X_i[S]|) + (\text{cyclewidth}(S_a) + 1) + 1$. A symmetric argument about b_{max} allows us to conclude,

$$\begin{aligned}\text{width}(C, X) \geq \max\Big(&\min_i |X_i[S]| + \text{cyclewidth}(S_a) + 1, \\ &\min_i |X_i[S]| + \text{cyclewidth}(S_b) + 1, \\ &\text{width}((C[S'], X[S']))\Big)\end{aligned} \tag{5}$$

Consider $C'[G \backslash S']$ and $C' \backslash (C'[G \backslash S'])$ separately. Recall that P is the path at which (P_{ab}, Y'') is merged with $(C[S'], X[S'])$. By construction,

$$\forall p \in V(P), |X_p| = \min_i |X_i[S]| \tag{6}$$

Recall that (P_{ab}, Y'') is a concatenation of (P_a, Y') and (P_b, Z'), so $\text{width}((P_{ab}, Y'')) = \max\Big(\text{width}((P_a, Y')), \text{width}((P_b, Z'))\Big)$. (P_a, Y') and (P_b, Z') are both constructed by distributing a node over a cycle decomposition constructed as in Construction 4.1 so, $\text{width}((P_a, Y')) \leq \text{cyclewidth}(S_a) + 1$ and $\text{width}((P_b, Z')) \leq \text{cyclewidth}(S_b) + 1$. Thus,

$$\text{width}((P_{ab}, Y'')) \leq \max(\text{cyclewidth}(S_a) + 1, \text{cyclewidth}(S_b) + 1) \tag{7}$$

By (6) and (7),

$$\begin{aligned}\text{width}((C'[G \backslash S], X')) &\leq \min_i |X_i[S]| + \max(\text{cyclewidth}(S_a) + 1, \text{cyclewidth}(S_b) + 1) \\ &\leq \text{width}((C, X)) \quad \text{by (5)}\end{aligned} \tag{8}$$

By construction,

$$\begin{aligned}\text{width}((C' \backslash (C'[G \backslash S'], X')) &= \text{width}((C[S'], X[S'])) \\ &\leq \text{width}((C, X)) \quad \text{by (5)}\end{aligned} \tag{9}$$

Thus,

$$\begin{aligned}\text{width}((C', X')) &= \max\Big(\text{width}((C'[G \backslash S], X')), \text{width}((C' \backslash (C'[G \backslash S'], X'))\Big) \\ &\leq \text{width}((C, X)) \quad \text{by (8) and (9)}\end{aligned}$$

If (C', X') has at least two spanning cycles, the above process can be repeated until the result has at most one spanning cycle. Thus, every cactus has a minimum-width cycle decomposition with at most one spanning cycle. □

From this point forward, we will only consider cycle decompositions of cacti with at most one spanning cycle. That is, when referring to a cycle decomposition of a cactus with spanning cycle C, it is implied that C is the only spanning cycle.

From Lemmas 4.2 and 4.3, we get:

Corollary 4.4. *Let G be a cactus. For some cycle $C \in G$, the result of Construction 4.1 is a minimum-width cycle decomposition of G.*

Theorem 4.5. *Let G be a cactus. Then* $\text{pathwidth}(G) - \text{cyclewidth}(G) \in \{0, 1\}$.

Proof. By Lemma 2.2, $\text{pathwidth}(G) - \text{cyclewidth}(G) \geq 0$.

Let (C, X) be a minimum-width cycle decomposition of G. By Corollary 4.4, we can suppose, without loss of generality, that (C, X) was constructed as in Construction 4.1. Consider an arbitrary vertex $v \in V(G)$ and adjacent nodes $i, j \in V(C)$ such that $X_i \cap X_j = \{v\}$. Stretch between i and j by 1 and distribute the added vertex over (C, X). The result is a path decomposition of width at most $\text{cyclewidth}(G)+1$, so $\text{pathwidth}(G)-\text{cyclewidth}(G) \leq 1$. □

Lemma 4.6. *Let (C, X) be a minimum-width cycle decomposition of a cactus G over all cycle decompositions with spanning cycle C_1. For each component G_i with respect to C_1, let v_i be the vertex in C_1 adjacent to G_i. Let G_j be the component with the greatest pathwidth, let C_2 be a cycle in $G_{k \neq j}$, and let (C', X') be a cycle decomposition of G in which C_2 is spanning. Then* $\text{width}((C, X)) \leq \text{width}((C', X'))$.

Proof. By Lemma 4.2, $\text{width}((C, X)) = 1 + \text{pathwidth}(G_j)$. Let $\{G'_i\}$ be the set of components with respect to C_2. Then, by the definition of cactus, there exists k such that $G_j \subseteq G'_k$. Thus, $\text{pathwidth}(G'_k) \geq \text{pathwidth}(G_j)$ so by Lemma 4.2, $\text{width}((C', X')) \geq 1 + \text{pathwidth}(G_j) = \text{width}((C, X))$. □

Theorem 4.7. *The cyclewidth of a cactus G can be determined in $O(f(n) \log n)$ time.*

Proof. The idea of the algorithm is to perform a binary search on the cycles in G to find the cycle C for which a minimum-width cycle decomposition has C as a spanning cycle. We must also consider the case in which a minimum-width cycle decomposition has no spanning cycles.

Let $\{C_i\}$ be the set of cycles in G and let $\{G_{ij}\}$ be the set of components with respect to C_i. Let v_{ij} denote the vertex in C_i adjacent to G_{ij}. Define the *center cycle* of a cactus as the cycle C_i that minimizes the maximum number of cycles in $G_{ij} \cup \{v_{ij}\}$ over all j. Let G_{im} be the maximum pathwidth component of C_i for all i. Let S be the iteratively updated set of indices i for which C_i has been tested as a spanning cycle. Initially, $S = \varnothing$.

One iteration of the algorithm is as follows. Find the center cycle C_j of $\cap_{i \in S}(G_{im} \cup \{v_{im}\})$. Use Construction 4.1 to find a minimum-width cycle decomposition (C^j, X^j) of G with C_j as the spanning cycle. Add j to S.

The complete algorithm is as follows. Perform repeated iterations until $\cap_{i \in S}(G_{im} \cup \{v_{im}\})$ contains no cycles. Then find the pathwidth of G and conclude that $\text{cyclewidth}(G) =$

$\min(\text{pathwidth}(G), \min_{i \in S} \text{width}((C^i, X^i)))$. By Lemma 4.6, the algorithm checks all spanning cycles that could possibly be part of a minimum-width cycle decomposition of G.

Since Construction 4.1 takes $O(f(n))$ time and the number of iterations in the algorithm is $O(\log n)$, the algorithm takes $O(f(n) \log n)$ time. □

5 Maximal Graphs of Cyclewidth k

A k-tree is an edge-maximal graph of treewidth k, equivalently defined as a chordal graph in which all maximal cliques are size $k+1$ and all minimal clique separators are size k. In particular, a 1-tree is a exactly a tree. Note that a tree decomposition (T, X) with no two identical bags is a width-k decomposition of a k-tree if and only if for all i, X_i is a $k+1$-clique with exactly k vertices in common with each of the bags corresponding to the neighbors of i.

A *k-path* is either a k-tree with exactly two vertices of degree k or a $k+1$-clique. I have k-path In particular, a 1-path is a exactly a path. Note that a path decomposition (P, X) with no two identical bags is a width-k decomposition of a k-path if and only if for all i, X_i is a $k+1$-clique with exactly k vertices in common with each of the bags corresponding to the neighbors of i and every vertex is either in at least two bags or a bag corresponding to an endpoint of P.

A k-caterpillar is an edge-maximal graph of pathwidth k, equivalently defined as a k-tree composed of a k-path with additional k-leaves adjacent to some of its separator k-cliques. In particular, a 1-caterpillar is exactly a *caterpillar tree*, a tree in which every vertex is at most a distance 1 from a central path. Note that a path decomposition (P, X) with no two identical bags is a width-k decomposition of a k-caterpillar if and only if for all i, X_i is a $k+1$-clique with exactly k vertices in common with each of the bags corresponding to the neighbors of i.

We now characterize edge-maximal graphs of cyclewidth k.

Definition 5.1. *A* k-circular caterpillar *is an edge-maximal graph of cyclewidth* k.

A *normal* circular arc representation is a circular arc representation of a graph in which no two arcs cover the entire circle.

Theorem 5.1. *A k-circular caterpillar is alternately defined as a circular arc graph with a normal circular arc representation in which every point on the circle is covered by either k or $k+1$ arcs. All k-circular caterpillars have exactly nk edges.*

Proof. Let G be a k-circular caterpillar. $V(G) \geq 2k+1$ because otherwise, by Lemma 2.8, $\text{cyclewidth}(G) < k$. Since G is edge-maximal, every bag of any cycle decomposition of G must be a $k+1$-clique, so G must be a circular arc graph with a circular arc representation in which every point on the circle is covered by either k or $k+1$ arcs. Let L be a such a circular arc representation list. L must alternate between starting points (of the form $A[i]_s$) and ending points (of the form $A[i]_t$). For all i, let S_i be the subset of $V(G)$ such that $v \in S_i$

if and only if v is the j^{th} vertex in G and $A[j]_s$ is among the first i starting points L. For all i, let M_i be the number of edges with at least one endpoint in S_i. For all i, $A[i]_s$ is contained in exactly k other arcs so $M_1 = k$ and for all i from 2 to $|V(G)|$, $M_i - M_{i-1} \leq k$. Since G is edge maximal, $M_i - M_{i-1} = k$, which is true if and only if there do not exist i and j such that the arc corresponding to the i^{th} vertex contains the point $A[j]_s$ and the arc corresponding to the j^{th} vertex contains the point $A[i]_s$. If such i and j exist, the arcs corresponding to i and j, respectively, cover the entire circle and the circular arc graph representation is not normal. Thus, a k-circular caterpillar is a circular arc graph with a normal circular arc representation in which every point on the circle is covered by either k or $k+1$ arcs. Since $M_1 = k$ and for all i from 2 to $|V(G)|$, $M_i - M_{i-1} = k$, G has exactly nk edges. □

Let (C, X) be a a width k cycle decomposition of a k-circular caterpillar G. Let j and l be adjacent nodes is C and remove the edge between them. Insert a new node h adjacent to only l, let X_h contain $k+1$ vertices disjoint from $V(G)$, and call the result (P, X'). Arbitrarily define a one-to-one correspondence between X'_i and X'_h. For all $g \in V(C)$, for all $v \in X'_j \cap X'_h$, if j and h are in different connected components of the subgraph of P induced by $\{i \in V(P) | v \in X_i\}$, replace v in X_g with its corresponding vertex in X_h. After all such replacements, let (P', X'') be the result. We say that (P', X'') is the result of *unidentifying* (C, X) at i.

Definition 5.2. *A* k-cycle *is a* k*-circular caterpillar with no vertices of degree* k*.*

In particular, a 1-cycle is exactly a cycle and a 1-circular caterpillar is exactly a *circular caterpillar* a graph in which every vertex is distance at most one from a central cycle.

Note that if (C, X) is a width-k cycle decomposition of a graph G such that every bag in (C, X) is unique, G is a k-circular caterpillar if and only if for all i, X_i is a $k+1$-clique with exactly k vertices in common with each of the bags corresponding to the neighbors of i. Thus, unidentifying (C, X) at any node in $V(C)$ results in a path decomposition of a k-caterpillar.

Note that if (C, X) is a width-k cycle decomposition of a graph G such that every bag in (C, X) is unique, G is a k-cycle if and only if for all i, X_i is a $k+1$-clique with exactly k vertices in common with each of the bags corresponding to the neighbors of i and every vertex is in at least two bags. Thus, unidentifying (C, X) at any node in $V(C)$ results in a path decomposition of a k-path.

We can recognize k-trees, k-paths, and k-caterpillars simply by iteratively removing cliques containing at least one vertex of degree k; however, recognizing k-cycles and k-circular caterpillars may be more complex. Iteratively removing from a k-circular caterpillar cliques containing at least one vertex of degree k, results in a k-cycle. Thus, the k-circular caterpillar recognition problem and the k-cycle recognition problem are equivalent. We leave k-cycle recognition as an open question. See Appendix A for a summary of the approaches considered.

6 Conclusion

This document leaves numerous questions to be further researched. In Section 3, we state the best known approximation ratio of a polynomial time algorithm for cyclewidth, which raises the question of whether this bound can be improved. It is NP-hard to approximate the pathwidth of a graph to within an additive constant [4], but it is unknown whether the same is true for cyclewidth.

In Section 4, we investigate the cyclewidth of cacti, which leaves many other graph classes to be considered. Future research should focus on finding the other graph classes on which CWP can be solved in polynomial time, as well as those on which CWP remains NP-complete. Additionally, future research should consider the question of finding bounds on the difference between cyclewidth and pathwidth for specific classes of graphs, besides cacti.

In Section 5, we leave unanswered the question of recognizing maximal graphs of cyclewidth k.

References

[1] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. In *J. Algorithms*, pages 1–16. Springer, 1998.

[2] Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs, 1993.

[3] HansL. Bodlaender. Treewidth: Characterizations, applications, and computations. In FedorV. Fomin, editor, *Graph-Theoretic Concepts in Computer Science*, volume 4271 of *Lecture Notes in Computer Science*, pages 1–14. Springer Berlin Heidelberg, 2006.

[4] HansL. Bodlaender, JohnR. Gilbert, Hjlmtr Hafsteinsson, and Ton Kloks. Approximating treewidth, pathwidth, and minimum elimination tree height. In Gunther Schmidt and Rudolf Berghammer, editors, *Graph-Theoretic Concepts in Computer Science*, volume 570 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 1992.

[5] Uriel Feige, MohammadTaghi Hajiaghayi, and James R. Lee. Improved approximation algorithms for minimum-weight vertex separators. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 563–572, New York, NY, USA, 2005. ACM.

[6] FedorV. Fomin, Pierre Fraigniaud, and Nicolas Nisse. Nondeterministic graph searching: From pathwidth to treewidth. In Joanna Jdrzejowicz and Andrzej Szepietowski, editors, *Mathematical Foundations of Computer Science 2005*, volume 3618 of *Lecture Notes in Computer Science*, pages 364–375. Springer Berlin Heidelberg, 2005.

[7] Zvi Galil and Nimrod Megiddo. Cyclic ordering is np-complete. *Theoretical Computer Science*, 5(2):179 – 182, 1977.

[8] M. Garey, D. Johnson, G. Miller, and C. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM Journal on Algebraic Discrete Methods*, 1(2):216–227, 1980.

[9] Rudolf Halin. S-functions for graphs. *Journal of Geometry*, 8(1-2):171–186, 1976.

[10] W. Hsu. Maximum weight clique algorithms for circular-arc graphs and circle graphs. *SIAM Journal on Computing*, 14(1):224–231, 1985.

[11] Wen-Lian Hsu. $o(m.n)$ algorithms for the recognition and isomorphism problems on circular-arc graphs. *SIAM J. Comput.*, 24(3):411–439, June 1995.

[12] Wen-Lian Hsu and Kuo-Hui Tsai. Linear time algorithms on circular-arc graphs. *Information Processing Letters*, 40(3):123 – 129, 1991.

[13] Toshinobu Kashiwabara and Toshio Fujisawa. Np-completeness of the problem of finding a minimum clique number interval graph containing a given graph as a subgraph. In *Proc. Symposium of Circuits and Systems*, 1979.

[14] Dexter Kozen and Alexa Sharp. On distance coloring, 2007.

[15] Ross M. McConnell. Linear-time recognition of circular-arc graphs. *Algorithmica*, 37(2):93–147, 2003.

[16] Charis Papadopoulos and Constantinos Voglis. Drawing graphs using modular decomposition. In Patrick Healy and NikolaS. Nikolov, editors, *Graph Drawing*, volume 3843 of *Lecture Notes in Computer Science*, pages 343–354. Springer Berlin Heidelberg, 2006.

[17] Neil Robertson and P.D. Seymour. Graph minors. i. excluding a forest. *Journal of Combinatorial Theory, Series B*, 35(1):39 – 61, 1983.

[18] Neil Robertson and P.D Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309 – 322, 1986.

[19] P. Scheffler. A linear algorithm for the pathwidth of trees. In Rainer Bodendiek and Rudolf Henn, editors, *Topics in Combinatorics and Graph Theory*, pages 613–620. Physica-Verlag HD, 1990.

[20] Karol Suchan and Ioan Todinca. Pathwidth of circular-arc graphs. In *Proceedings of WG 2007, Lecture Notes in Computer Science 4769, 2007*, pages 258–269, 2007.

Harvey Mudd College
Email: nwein@hmc.edu

Appendix A: Approaches Considered for the k-cycle Recognition Problem

Consider the problem of deciding whether or not a given graph G is a k-cycle.

Decomposition Construction

Consider the problem of finding a set S of $k+1$-cliques in G such that there exists a cycle decomposition of G such that each bag contains exactly the vertices of a $k+1$-clique in S, or determining that such an S does not exist. Locating a $k+1$-clique in a circular arc graph, and thus in a k-cycle, can be done in polynomial time [10]; however, by Lemma 2.9, a k-cycle may contain a $2k$-clique, and thus $\binom{2k}{k+1}$ $k+1$-cliques. Let G' be a k-cycle, let K be a $k+1$-clique in G', and let (C, X) be a width-k cycle decomposition of G'. By Lemma 2.10, either K spans (C, X) or $V(C)$ is contained in a single bag. Since every bag of (C, X) must be a clique, $V(K)$ must be contained in a single bag if K is not adjacent to every other vertex in G'. For the same reason, K must be spanning if there exist $u, v, w \in V(K)$ and $x, y, z \in V(G')$ such that x, y, and z are each adjacent to a different combination of exactly two of $\{u, v, w\}$. If K does not satisfy either of these criteria, it may be the case that K is spanning in some width-k cycle decompositions of G' and in others, $V(C)$ is contained in a single bag. Thus, finding such an S or determining that one does not exist may not be a straightforward endeavor.

Recognition of k-paths

Consider the following manipulation of G. Find a $k+1$-clique K in G such that there exists a width-k cycle decomposition (C, X) of G with $X_i = V(K)$ for some i. Replace K with two $k+1$-cliques, K_1 and K_2, and define a one-to-one correspondence between $V(K)$ and each of $V(K_1)$ and $V(K_2)$. Bipartition the set S of edges incident to at least one vertex in K into S_1 and S_2. For each edge e in S_1 and S_2, respectively, replace the endpoint $v \in K$ of e with the vertex in K_1 and K_2, respectively, corresponding to v. Let G' be the resulting graph. G is a k-cycle if and only if is possible to bipartition S such that G' is a k-path. In polynomial time, it may not be possible to find K and determine whether there exists a bipartition of S such that G' is a k-path.

The following is a similar method without the bipartition step, but involving two $k+1$-cliques in G instead of one. Consider finding two disjoint $k+1$-cliques, K_1 and K_2, in G such that there exists a width-k cycle decomposition (C, X) of G with $X_i = V(K_1)$ and $X_j = V(K_2)$ for some i, j. If such K_1 and K_2 exist, G is a k-cycle if and only if $G \backslash (K_1 \cup K_2)$ has exactly two connected components, H_1 and H_2, and $G[H_1 \cup K_1] \cup G[H_1 \cup K_2]$ and $G[H_2 \cup K_1] \cup G[H_2 \cup K_2]$ are both k-paths. If G is a k-cycle, such K_1 and K_2 may not exist, and if they do, it may not be possible to find them in polynomial time.

Minimum-Thickness Circular Arc Representation

All k-cycles are circular arc graphs whose minimum-thickness circular arc representations contain either k or $k+1$ arcs covering every point on the circle. If G is not a circular arc graph then G is not a k-cycle, so we suppose that G is a circular arc graph. Given a minimum-thickness circular arc representation of G corresponding to a cycle decomposition (C, X) of G, G is a k-cycle if and only if for all i, X_i is a $k+1$-clique with exactly k vertices in common with each of the bags corresponding to the neighbors of i and every vertex is in at least two bags. These properties can be checked in linear time, so the k-cycle recognition problem reduces to the problem of finding a minimum-thickness circular arc representation of a circular arc graph.

The following results concerning circular arc graphs may be of interest. Circular arc graphs can be recognized in linear time [15]. The circular arc graph isomorphism problem is solvable in $O(mn)$ time [11]. A minimum-width path decomposition of a circular arc graph can be found in $O(n^2)$ time [20]. Clique cover, maximum independent set, and minimum dominating set can all be solved in linear time on circular arc graphs [12].

Reduction from NP-Complete Problem

Brief consideration was given to reducing the following NP-complete problems to the k-cycle recognition problem.
The circular arc graph coloring problem: Given a circular arc graph G, is G k-colorable? [8]
The cyclic ordering problem: Given a set S and a set of cyclically ordered triples in S, is there a cyclic ordering of S consistent with all of the cyclically ordered triples? [7]