SIAM J. COMPUT. © 1999 Society for Industrial and Applied Mathematics
Vol. 28, No. 5, pp. 1906–1922

TRACTABILITY OF PARAMETERIZED COMPLETION PROBLEMS ON CHORDAL, STRONGLY CHORDAL, AND PROPER INTERVAL GRAPHS[*]

HAIM KAPLAN[†], RON SHAMIR[‡], AND ROBERT E. TARJAN[§]

Abstract. We study the parameterized complexity of three NP-hard graph completion problems. The *minimum fill-in* problem asks if a graph can be triangulated by adding at most k edges. We develop $O(c^k m)$ and $O(k^2 mn + f(k))$ algorithms for this problem on a graph with n vertices and m edges. Here $f(k)$ is exponential in k and the constants hidden by the big-O notation are small and do not depend on k. In particular, this implies that the problem is fixed-parameter tractable (FPT).

The *proper interval graph completion* problem, motivated by molecular biology, asks if a graph can be made proper interval by adding no more than k edges. We show that the problem is FPT by providing a simple search-tree-based algorithm that solves it in $O(c^k m)$-time. Similarly, we show that the parameterized version of the *strongly chordal graph completion* problem is FPT by giving an $O(c^k m \log n)$-time algorithm for it.

All of our algorithms can actually enumerate all possible k-completions within the same time bounds.

Key words. design and analysis of algorithms, parameterized complexity, chordal graphs, proper interval graphs, strongly chordal graphs, minimum fill-in, physical mapping of DNA

AMS subject classifications. 68Q20, 68R15, 05C85

PII. S0097539796303044

1. Introduction.

The focus of this paper is the parameterized complexity of several graph completion problems. Many well-known NP-hard problems can be stated with a parameter k so that they have polynomial-time algorithms when k is fixed. (For example, given a graph, decide if it has a vertex cover of size at most k, an independent set of size at least k, or pathwidth at most k.) The way the complexity depends on k varies dramatically, however. Some problems (e.g., *vertex cover* and *pathwidth*) can be solved in linear time when k is fixed, but for others (like *independent set*) the best known algorithms require $\Omega(n^k)$ steps. How the complexity depends on k can be crucial for applications in which small, fixed parameter values are important, as in the problems we study here.

Downey and Fellows initiated a systematic complexity analysis of such problems [1, 9, 10]. They called those parameterized problems that have algorithms of complexity $O(f(k)n^\alpha)$ (with α a constant) *fixed-parameter tractable* (FPT) and defined

[*]Received by the editors May 6, 1996; accepted for publication (in revised form) November 4, 1997; published electronically May 26, 1999. Portions of this paper were presented at the 34th Annual IEEE Symposium on the Foundations of Computer Science, Santa Fe, NM, 1994, and were published as *Tractability of parameterized completion problems on chordal and interval graphs: Minimum fill-in and physical mapping* in Proceedings of the 35th Symposium on Foundations of Computer Science, IEEE Computer Science Press, Los Alamitos, CA, 1994, pp. 780–791.
http://www.siam.org/journals/sicomp/28-5/30304.html

[†]AT&T Labs Research, 180 Park Ave, Florham Park, NJ 07932 (hkl@research.att.com).

[‡]Department of Computer Science, Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel (shamir@math.tau.ac.il). The work of this author was supported in part by a grant from the Ministry of Science and the Arts, Israel.

[§]Department of Computer Science, Princeton University, Princeton, NJ 08544, and InterTrust Technologies Corporation, Sunnyvale, CA 94086 (ret@cs.princeton.edu). The research of this author at Princeton University was partially supported by NSF grant CCR-8920505 and Office of Naval Research contract N00014-91-J-1463.

a hierarchy of parameterized decision problem classes, $FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots$, with appropriate reducibility and completeness notions. They also conjectured that each of the containments in this hierarchy is proper. (See [1, 9, 10] for definitions and details.) Thus, for example, *vertex cover* and *pathwidth* are in FPT [3, 11, 24], but *independent set* is $W[1]$-complete [1] and *bandwidth* is W[t]-hard for all t [4].

Let Π be a family of graphs such that $K_n \in \Pi$ for every n. The Π-*completion* problem is defined as follows. Given a graph $G = (V, E)$, find a smallest set of edges A such that $G = (V, E \cup A) \in \Pi$. The parameterized version of the Π-completion problem, denoted by Π-*completion(k)*, asks whether there exists an edge set A such that $|A| \leq k$ and $G = (V, E \cup A) \in \Pi$.

In this paper we study the parameterized complexity of Π-*completion(k)* for three graph families Π; namely, chordal, proper interval, and strongly chordal graphs.

A graph is *chordal* (or *triangulated*) if every cycle of length four or more contains a chord (an edge between nonadjacent vertices on the cycle). The *chordal completion* problem is also known as the *minimum fill-in* problem and has received a lot of attention in the past due to its importance in sparse matrix computation (cf. [17]). Rose [33] has shown that for a sparse, symmetric matrix, finding an order of Gaussian elimination steps on diagonal elements that minimizes the number of nonzeros generated in the elimination process (assuming no lucky cancellation of nonzeros) is equivalent to solving the minimum fill-in problem on a corresponding graph.

Yannakakis [41] has shown that minimum fill-in is NP-complete. We focus here on *chordal completion(k)* or *fill-in(k)*, the parametrized version of the problem as defined above. Here k is fixed (to be thought of as a small constant) and is not part of the input. For a graph with n vertices and m edges, the problem can be solved by enumeration in $O(n^{2k}m)$-time, but we seek an algorithm with better dependence on k. In section 2 we describe two such algorithms. We first present a fairly simple, $O(c^k m)$-time search-tree-based algorithm, which already implies that the problem is in FPT. The same technique was previously used by Downey and Fellows [11] to prove parameterized tractability of *vertex cover*, *dominating set in planar graph, feedback vertex set,* and *face cover number of planar graph*. We then develop a more involved algorithm that gives a stronger complexity result: its multiplicative factor depending on k is *polynomial*, and the exponential in k appears only as an *additive* factor. Specifically, this algorithm has complexity $O(k^2 nm + f(k))$, where $f(k)$ is exponential in k. For appropriate values of k (growing with the graph size), this algorithm will beat the simple algorithm. Here and throughout the paper we specify the dependence of the complexity on k explicitly, and the constants hidden by the big-O notation do not depend on k.

The second part of the paper deals with the parameterized complexity of the *proper interval completion* (PIC) problem. An *interval graph* is a graph for which one can assign an interval on the real line to each vertex so that two vertices are adjacent iff their intervals intersect. It is a *unit interval graph* if all intervals assigned have equal length. It is *proper interval* if it has an assignment in which no interval properly contains another. The last two notions are equivalent for finite graphs [30]. Interval completion problems arise in molecular biology and in the human genome project. In *physical mapping* of DNA, a set of long contiguous intervals of the DNA chain (called *clones*) is given, together with experimental information on their pairwise overlaps. The goal is to build a map describing the relative position of the clones [7, 27, 22, 4]. We concentrate here on the biologically important case in which all clones have equal length. In the presence of "false negative" errors (unidentified overlaps) the problem of building a map with fewest errors is equivalent to PIC. This problem is NP-hard [18]. But what about its complexity for a small fixed number of errors? Let PIC(k) be the parameterized version of the problem, in which one asks for an augmenting set with no more than k edges if one exists. We prove parameterized tractability of PIC(k) by providing an $O(c^k m)$-time algorithm.

The third part of the paper considers the parameterized version of the *strongly chordal completion* problem (SCC(k)). The class of strongly chordal graphs was defined and characterized by Farber [12]. Denote by $N(v)$ the set of neighbors of a vertex v, including v itself. A *perfect elimination ordering* of a graph $G = (V, E)$ is an ordering $v_1, v_2, \ldots, v_n$ of V with the property that for each i, j, and l, if $i < j, i < l$, and $v_l, v_j \in N(v_i)$, then $v_l \in N(v_j)$. Rose [31] has shown that a graph is chordal iff it admits a perfect elimination ordering. A *strong elimination ordering* of a graph $G = (V, E)$ is an ordering $v_1, v_2, \ldots, v_n$ of V with the property that for each i, j, k, and l, if $i < j$, $k < l$, $v_k, v_l \in N(v_i)$, and $v_k \in N(v_j)$, then $v_l \in N(v_j)$. A graph is *strongly chordal* if it admits a strong elimination ordering. It is easy to see that every strong elimination ordering is also a perfect elimination ordering, and thus every strongly chordal graph is also a chordal graph. In addition every interval graph is strongly chordal. One can obtain a strong elimination order for an interval graph G by fixing a representation R of G and ordering the vertices in increasing right-endpoint order of their intervals in R. Interest in strongly chordal graphs arises in several ways. First, the problems of locating minimum weight dominating sets and minimum weight independent dominating sets in strongly chordal graphs with real vertex weights can be solved in polynomial time, whereas each of these problems is NP-hard for chordal graphs [13]. Second, these graphs have surprisingly nice structural properties and are intimately related to the class of totally balanced matrices [2]. We show that SCC(k) is FPT by describing an $O(c^k m \log n)$-time algorithm for it.

Section 2 contains the algorithms for chordal completion. Section 2.1 describes the simple search-tree–based algorithm, and section 2.2 gives the details of the more involved $O(k^2 nm + f(k))$-time algorithm. Section 3 extends the search tree algorithm of section 2.1 to solve PIC(k), and section 4 extends it to solve SCC(k). These extensions require additional ideas in order to handle the obstructions characterizing each particular graph family. Section 5 contains a summary and suggestions for some further research.

2. Minimum fill-in.

In this section we present two algorithms for *fill-in(k)*. In section 2.1 we begin by describing an $O(c^k m)$-time algorithm for the problem. Then in section 2.2 we use additional new ideas to develop an $O(k^2 nm + f(k))$-time algorithm. Which of these algorithms is faster depends on the size of k compared to n and m. Both algorithms can actually enumerate all minimal k-triangulations of the input graph within the same time bounds.

We will use the following notation. Let $G = (V, E)$ be an undirected graph. For $X \subseteq V$, we denote by G_X the subgraph of G induced by the vertex set X. We define the *length* of a path (cycle) as the number of edges on the path (cycle). A *triangulation* of a graph $G = (V, E)$ is a set of edges F where $E \cap F = \emptyset$ and $\tilde{G} = (V, E \cup F)$ is a chordal graph. We will also say that the set of edges F *triangulates* G. If $|F| \leq k$, then F is a *k-triangulation.* We shall also refer to $\tilde{G}$ as a triangulation of G when there is no confusion. We assume without loss of generality that G is connected and $n \geq 2$; thus $n = O(m)$. A triangulation F is *minimal* if no proper subset of F triangulates G.

2.1. A linear algorithm for fixed k.

A *triangulation of a chordless cycle* C is a set T of chords of C such that there is no induced chordless cycle in $C \cup T$. We shall characterize and count the number of minimal triangulations of a cycle C. We call a cycle an *l-cycle* if it contains l vertices. A *triangle* is a 3-cycle. The proof of the following lemma is straightforward by induction.

LEMMA 2.1. *A minimal triangulation T of an n-cycle C consists of $n-3$ chords. It partitions C into $n-2$ triangles. Any two of these triangles are either disjoint or share a chord. Every chord in T is shared by exactly two triangles.*

The following lemma is well known (cf. [34] and the proof of Lemma 4.3, which is similar).

LEMMA 2.2. *There is a* 1-1 *correspondence between the minimal triangulations of a cycle with l vertices and the binary trees with $l-2$ internal nodes.*

Denote by c_l the lth Catalan number, i.e., $c_l = \binom{2l}{l}\frac{1}{l+1}$. Note that $c_l < 4^l$. Denote the number of binary trees with n internal nodes by b_n. The value b_n satisfies the recurrence $b_0 = 1$, $b_n = \sum_{i+j=n-1} b_i b_j$ for $n \geq 1$. The solution to this recurrence is $b_n = c_n$ (cf. [19]). Thus the following lemma is implied by Lemma 2.2.

LEMMA 2.3. *The number of minimal triangulations of an l-cycle is $c_{l-2} \leq 4^{l-2}$.*

The algorithm will traverse part of a search tree in which each node corresponds to a supergraph of G. This search tree is defined as follows. The graph G itself corresponds to the root of the tree. In order to generate the children of an internal node x that corresponds to a graph G', one needs to find a chordless cycle C in G'. Node x will have a child for each minimal triangulation of C. The graph corresponding to a child is obtained by adding the corresponding minimal triangulation to G'. If $|C| = l$, by Lemma 2.3 node x will have c_{l-2} children. Each leaf of the tree corresponds to a chordal supergraph of G. Note that every minimal triangulation of G is represented by at least one leaf.

One can find a chordless cycle C in a nonchordal graph with m edges in $O(m+n)$-time by the maximum cardinality search (MCS) algorithm described in [37, 38]. Using the algorithm described in [35] and the mapping described in Lemma 2.2, one can generate all minimal triangulations in $O(|C|)$-time per triangulation.

The algorithm actually visits only the nodes of the search tree that correspond to supergraphs of G with no more than k additional edges. If one such node is a leaf, then we have found a k-triangulation. Otherwise, no such triangulation exists.

THEOREM 2.4. *All minimal k-triangulations of a graph G can be found in $O(2^{4k}m)$-time.*

Proof. Let T be the subtree of the search tree traversed by the algorithm. For a node $x \in T$ let $G_x = (V, E_x)$ be the corresponding supergraph of G, d_x the maximum length of a path from x to a leaf of T, and $a_x = \max(\{|E_l| - |E_x| \mid l \text{ is a leaf descendant of } x\})$. Denote by l_x the total number of leaves among the descendants of x. By induction we prove that $l_x \leq 4^{d_x + a_x}$. Thus the total number of nodes in T is bounded by $2 \cdot 4^{2k}$. For each such node a linear amount of time is spent, consisting of the time to generate it and the time to find a chordless cycle in the graph corresponding to it.

Here is the induction argument. Assume the claim is true for all the children of a node x. Let l be the length of the cycle detected at x. Let $d_{\max} = \max\{d_y \mid y \text{ is a child of x}\}$, and let $a_{\max} = \max\{a_y \mid y \text{ is a child of x}\}$. Using the induction hypothesis, the number of leaf descendants of any of the c_{l-2} children of x is bounded by $4^{d_{\max}+a_{\max}}$. Thus the total number of leaf descendants of x is bounded by $4^{l-2}4^{d_{\max}+a_{\max}} = 4^{d_{\max}+1+a_{\max}+l-3} = 4^{d_x+a_x}$. The last equality follows from the fact that the size of a minimal triangulation of a chordless l-cycle is $l-3$, as stated in Lemma 2.1. ∎

The algorithm for enumerating minimal k-triangulations can actually list the same triangulation several times. We can eliminate this redundancy by storing solutions in a table and checking each new solution to see if it has been found already. If we use a k-dimensional search tree to store solutions, the extra time per search tree node to test for redundancy is $O(k \log k)$. Using universal hashing [39] or dynamic perfect hashing [16], the extra time per search tree node is $O(k)$, but the algorithm becomes randomized. These ideas apply equally well to the other enumeration algorithms proposed in this paper.

2.2. An algorithm with a polynomial multiplicative factor. To achieve an $O(k^2nm + f(k))$-time bound for minimal k-triangulation, we first describe an algorithm such that if G can be triangulated with no more than k edges, the algorithm partitions the vertex set of G into two subsets A, B such that the size of A is $O(k^3)$ and there are no chordless cycles in G that contain vertices in B. Then we prove that obtaining a k-triangulation of G is equivalent to obtaining a $(k-a)$-triangulation of A for some $a \geq 0$.

2.2.1. Partitioning the graph. The algorithm uses three main procedures, denoted P_1, P_2, P_3, executed in sequence. These procedures are described below.

(P_1) *Extracting independent chordless cycles*. This procedure starts with $B = V, A = \emptyset$ and repeatedly finds a chordless cycle in G_B using the MCS algorithm and moves its vertices to A. Note that when P_1 is finished, the induced subgraph on B is chordal.

Let $C_1, \ldots, C_j$ be the cycles extracted. The minimum number of chords needed to triangulate each C_i is $|C_i| - 3$. The algorithm maintains a dynamic lower bound cc on the minimum number of chords needed to triangulate G. After detecting the chordless cycle C_i, it increases cc by $|C_i| - 3$. Thus, if at some point $cc > k$, the algorithm can stop with a negative answer. Otherwise procedure P_1 ends when there are no more chordless cycles in B and $cc = \sum_{i=1}^{j}(|C_i| - 3) \leq k$.

The complexity of this part is $O(km)$. The MCS algorithm runs in linear time and the number of cycles detected is not greater than k since each cycle adds at least one to the dynamic lower bound cc. The size of the set A after performing this procedure is $O(k)$.

(P_2) *Extracting related chordless cycles with independent paths*. This procedure looks for chordless cycles that intersect both parts of the current partition, A and B, and contain at least two consecutive vertices in B, as long as such cycles exist. Let C be such a cycle, $|C| = l$. If $l > k + 3$, the algorithm stops with a negative answer. Otherwise every maximal subpath of C containing only vertices from B is moved into A if its length is at least one. The increase to cc depends on the structure of C. We need the following lemma in order to specify this increase precisely.

LEMMA 2.5. *Let C be a chordless cycle, and let p be a path in C of length l with $1 \leq l \leq |C| - 2$. If $l = |C| - 2$, then in every minimal triangulation of C there are at least $l - 1$ chords incident with at least one vertex of p. If $l < |C| - 2$, then in every minimal triangulation of C there are at least l chords incident with at least one vertex on p.*

Proof. If $l = |C| - 2$, then every chord in a minimal triangulation of C is incident with some vertex of p; thus the first part of the lemma holds. We prove the second part by induction on the path length. Obviously there must be a chord incident with at least one of the vertices on p; thus the lemma holds for paths of length one. Assume the result is true for every path with length less than l. Let p be a path with length l. Let (a, b) be a chord incident with p dividing the cycle C into two cycles C_1, C_2.

Case 1. $a, b \in p$. Let l_1 be the length of the subpath of p that connects a and b. Without loss of generality we can assume that $l_1 = |C_1| - 1$. Let p' be the path between the endpoints of p passing through (a, b) in C_2, and let $l_2 = |p'|$. There must be at least $l_1 - 2$ chords incident with p in C_1 and according to the induction hypothesis l_2 chords incident with p' in C_2. Thus the total number of chords incident with p will be at least $(l_1 - 2) + l_2 + 1 = l$.

Case 2. $a \in p$, $b \notin p$. Let $p_1 = p \cap C_1$, $p_2 = p \cap C_2$. For at least one $i = 1, 2$, $|p_i| < |C_i| - 2$. Without loss of generality assume that $|p_1| < |C_1| - 2$. By applying the induction hypothesis and using the previous part of the lemma, we find that the total number of chords incident with p is at least $l_1 + (l_2 - 1) + 1 = l$. ☐

Suppose that C is a chordless l-cycle that contains $j \geq 1$ disjoint maximal subpaths $p_1, \dots, p_j$, each of length at least one, that are in B. Let $l_i = |p_i|$, $i = 1, \dots, j$. Obviously if $l_1 = l - 2$, then $j = 1$, i.e., there is only one such subpath. Otherwise $l_i < l - 2$ for every $1 \leq i \leq j$. Using the previous lemma, we can increase our lower bound cc as follows. If there is only one such subpath, cc is increased by either $(l_1 - 1)$ if $l_1 = l - 2$ or l_1 if $l_1 < l - 2$. Otherwise cc is increased by the larger of $\frac{1}{2}\sum_{i=1}^{j} l_i$ (the factor $\frac{1}{2}$ is needed because a chord can be counted twice in the sum) and $\max\{l_i \mid 1 \leq i \leq j\}$. P_2 terminates whenever either cc is greater than k, in which case it stops with a negative answer, or when there are no more cycles of the appropriate kind.

In order to complete the description of P_2 we need to specify how to detect a chordless cycle C with consecutive vertices in B if such a cycle exists. The following observation is useful.

OBSERVATION 2.6. *There exists a chordless cycle C with at least two consecutive vertices in B iff there exists an edge (x, y), $x \in A$, $y \in B$ and a path between a vertex in $(N(y) - N(x)) \cap B$ and a vertex in $N(x) - N(y)$ that avoids any other vertices in $N(x) \cup N(y)$.*

One can detect whether such a path exists as follows: Delete $N(x) \cap N(y)$ and $(N(y) - N(x)) \cap A$ from G. Find the connected components of G induced on the other vertices. Check whether there is a vertex in $(N(y) - N(x)) \cap B$ and a vertex in $N(x) - N(y)$ in the same connected component. This process requires $O(m)$-time per edge (x, y) and can be implemented so that if the path exists, then the process will output a chordless cycle through (x, y) for which the other neighbor of y is also in B.

Recall that the size of A after the execution of P_1 is $O(k)$. The number of vertices added to A after the detection of each cycle by P_2 is at most twice the increase to cc. Since cc is never greater than k, the total number of vertices in A when P_2 ends remains $O(k)$.

(P_3) *Adding essential edges in G_A*. For every nonadjacent pair of vertices $y, z \in V$ define $A_{y,z}$ to be the set of all vertices x such that y, x, z appear consecutively on some chordless cycle in G.

LEMMA 2.7. *If for some pair $y, z \in A$, $(y, z) \notin E$, $|A_{y,z}| > 2k$, then the edge (y, z) is in every k-triangulation of G.*

Proof. Assume that (y, z) is not in a k-triangulation $\overline{G} = (V, \overline{E})$ of G. Then there must be a chord in $\overline{E} - E$ incident with each vertex in $A_{y,z}$. Since no more than two such vertices can share a chord, $|\overline{E} - E| > k$, which is a contradiction. ☐

Edges (y, z) satisfying the lemma are called *essential*. For a triple y, x, z such that $(y, x) \in E$, $(x, z) \in E$, $(y, z) \notin E$ one can determine whether y, x, z appear consecutively on some chordless cycle in linear time: They appear consecutively on a chordless cycle iff y and z are in the same connected component after deleting $N(x) - \{y, z\}$ from G.

P_3 first calculates the sets $A_{y,z}$ for every pair $y, z \in A$, $(y, z) \notin E$. Then for each pair $y, z \in A$ such that $|A_{y,z}| > 2k$, we add (y, z) to G'. Finally, we add to A all vertices in each computed set $A_{y,z}$ such that $|A_{y,z}| \leq 2k$.

We now analyze the overall complexity of the partitioning scheme.

LEMMA 2.8. (1) *The execution of P_2 takes $O(knm)$-time.* (2) *The execution of P_3 takes $O(k^2nm)$-time.*

Proof. (1) For each edge (x, y), $x \in A$, $y \in B$, it takes linear time to find a chordless cycle through (x, y) with consecutive vertices in B. The size of A is always $O(k)$; thus the total number of edges incident with vertices of A is always $O(kn)$. For each such edge we may have to run the test mentioned above once, giving a total time complexity $O(knm)$.

(2) Since the size of A when P_3 begins its execution is $O(k)$, the number of triples y, x, z such that $(y, x), (z, x) \in E$, $(y, z) \notin E$, $y, z \in A$, is $O(k^2n)$. For each triple we need to check whether there exists a path between y and z after deleting $N(x) - \{y, z\}$ from G. As mentioned above, this can be done by identifying connected components of G on the remaining vertices and then checking whether y and z are in the same connected component. ☐

Thus the overall complexity of the partitioning procedure is dominated by the complexity of P_3, which is $O(k^2nm)$. Before the call to P_3, the size of the set A is $O(k)$. Procedure P_3 may add $O(k)$ additional vertices to A for each pair of vertices in A prior to its execution so that we end up with $O(k^3)$ vertices in A.

Let E_s be the set of essential edges detected by P_3, and let $G' = (V, E \cup E_s)$. Denote by A_2, B_2 the partition of the vertex set before the execution of P_3 and by A, B the final partition.

The following lemma will be useful in establishing the correctness of the partitioning scheme and the completion algorithm.

LEMMA 2.9. *Let $G = (V, E)$ be a graph and $v \in V$. Let F be a set of edges between vertices of G such that*

(1) *each $e \in F$ is a chord in a chordless cycle C_e of G,*

(2) *$F \cap E = \emptyset$,*

(3) *v is not an endpoint of any $e \in F$.*

Denote by G^+ the graph obtained from G by adding the edges in F. If there exists a chordless cycle C in G^+ with v_1, v, v_2 occurring consecutively on C, then either there exists a chordless cycle in G on which v_1, v, v_2 occur consecutively or there exists a chordless cycle $D_e = v, x_1, \ldots, x_t, v$ in G such that the path $p = x_1, \ldots, x_t$ is part of a cycle C_e for some $e \in F$, and p contains one of the endpoints of e.

Proof. For an $e = (x, y) \in F$ let P_e^1 and P_e^2 denote the two paths on C_e between x and y with x and y removed from each path. Since C_e is chordless, for every e such that $v \in C_e$, v is not adjacent to any vertex either on P_e^1 or on P_e^2.

Case 1. For every $e \in F$ such that $v \notin C_e$, there exists a path $P_e \in \{P_e^1, P_e^2\}$ such that v is not adjacent in G to any vertex on P_e. Consider the cycle C. Replacing every edge $e \in F$ along C by P_e, one gets a cycle C' in G (not necessarily chordless or simple) with the property that v is not adjacent to any vertex in $C' - \{v_1, v_2\}$. Since edges in F are not incident with v, the edges (v, v_1) and (v, v_2) exist in G. Since C is chordless, $(v_1, v_2) \notin E$. Thus C' contains a chordless cycle in G on which v_1, v, v_2 occur consecutively.

Case 2. For some $e = (x, y) \in F$, v is adjacent to a vertex $u_1 \in P_e^1$ and a vertex

$u_2 \in P_e^2$. Since C is chordless in G^+, v must be nonadjacent either to x or to y in G. Without loss of generality assume v is not adjacent to x and that u_1 and u_2 are the closest to x among all vertices on P_e^1 and P_e^2, respectively, that are adjacent to v. D_e is the chordless cycle in G consisting of the path between u_1 and u_2 through x on C_e and v. $\square$

The correctness of the partitioning scheme is captured by the following theorem.

THEOREM 2.10. *When the partitioning procedure ends, the graph G' has no chordless cycles with vertices in B.*

Proof. The proof is by contradiction. Suppose that there is a chordless cycle $C \in G'$ such that $C \cap B \neq \emptyset$, and let v be a vertex in $C \cap B$. Denote by v_1 and v_2 the two neighbors of v on C. Cycle C must contain at least one essential edge since otherwise C exists in G and either v would have been moved to A or (v_1, v_2) would have been added as an essential edge. Let F be the set of essential edges on C. By the definition of an essential edge, for each $e = (x, y) \in F$ there is a chordless cycle C_e in G in which e is a chord. Moreover, if P_e^1 and P_e^2 are the two paths connecting x and y on C_e, then either P_e^1 or P_e^2 consists of a single vertex $z_e \in B_2$. Since v is in B, it is not incident with any essential edge. Applying Lemma 2.9 we find that one of the following things must happen.

(1) There exists in G a chordless cycle on which v_1, v, v_2 occur consecutively. Thus either v should have been in A or (v_1, v_2) should have been added as an essential edge, and we obtain a contradiction.

(2) There exists a chordless cycle D_e in G on which v and z_e occur consecutively for some $e \in F$. Since both v and z_e are in B_2, they both should have been moved to A by P_2, and again we obtain a contradiction. $\square$

2.2.2. Triangulating the graph.

All that remains is to show that once we have partitioned the graph, it suffices to look for $(k-a)$-triangulations of the smaller graph with vertex set A, where a is the number of essential edges added during the partitioning algorithm. This is the content of Theorem 2.13 below. In order to prove the theorem we need some background and preliminary results.

We define the *elimination* of a vertex v from G as the operation that deletes v from G and adds an edge between every nonadjacent pair among v's neighbors. Let $\alpha = v_1, \ldots, v_n$ be an ordering of the vertices of a graph $G = (V, E)$. We denote by G_i, $0 \leq i \leq n$ the graph obtained from G after eliminating the first i vertices in α $(G_0 = G)$. Let x and y be two vertices in $G = (V, E)$. An *x,y separator* of G is a set $S \subseteq V - \{x, y\}$ such that when S is deleted from G, x and y occur in different connected components.

The following characterization of minimal triangulations was proved by Ohtsuki, Cheung, and Fujisawa.

THEOREM 2.11 (see [28]). *A triangulation F of $G = (V, E)$ is minimal iff for each $(x, y) \in F$, there exists no x, y separator S of G such that S is a clique of the triangulated graph $\tilde{G} = (V, E \cup F)$.*

Using this theorem we prove the following lemma that is needed for the proof of Theorem 2.13.

LEMMA 2.12. *Let F be a minimal triangulation of a graph $G = (V, E)$. Any edge in F is a chord in a chordless cycle of G.*

Proof. Let $v_1, \ldots, v_n$ be a perfect elimination ordering of $\tilde{G} = (V, E \cup F)$. We use this ordering to eliminate vertices from G. Since F is minimal, for each edge $e = (u, w) \in F$ there exists an index k, $k \geq 1$, such that $e \in G_k$ but $e \notin G_{k-1}$.

We claim that u and w are connected in G_{k-1} by a path such that none of its vertices is adjacent to v_k. Here is the proof of the claim. Assume that no such path exists. Then the set $N_{G_{k-1}}(v_k) - \{u, w\}$ separates u and w in G_{k-1}. But it follows from the definition of a perfect elimination ordering that this set is a clique in $\tilde{G}_{k-1}$. Since F is a minimal triangulation of G, we must also have that $\tilde{G}_{k-1}$ is a minimal triangulation of G_{k-1}. This contradicts Theorem 2.11 and the claim follows.

We obtain that e is a chord of a chordless cycle in G_{k-1}. This cycle consists of u, v_k, and w occurring consecutively and a shortest path between u and w that avoids the neighborhood of v_k in G_{k-1}. We finish the proof by showing that e is also a chord of a chordless cycle of G. This is done by arguing that if C is a chordless cycle in G_j for some j, $1 \leq j \leq n$, then there exists a chordless cycle C' in G_{j-1} that is either identical to C or contains one additional vertex. If all the edges in C are in G_{j-1}, then $C' = C$ is a chordless cycle in G_{j-1}. Otherwise there is an edge (x, y) in C that is not in G_{j-1}. For each such edge both its endpoints must be adjacent to v_j. Of the vertices on C only x and y can be adjacent to v_j since otherwise C is not chordless in G_j. Take C' to be C with the vertex v_j added between x and y. ☐

THEOREM 2.13. *Let A, B be a partition of the vertex set V of a graph $G = (V, E)$ such that the vertices of every chordless cycle in G are contained in A. A set of edges F is a minimal triangulation of G iff F is a minimal triangulation of G_A.*

Proof. Let F be a minimal triangulation of G_A. We need to prove that $\tilde{G} = (V, E \cup F)$ is chordal. Assume that $\tilde{G}$ is not chordal. Let C be a chordless cycle in $\tilde{G}$. Since $\tilde{G}$ induced on A is chordal, $C \cap B \neq \emptyset$. By assumption, G does not contain chordless cycles with vertices in B; hence C must not exist in G and thus it contains at least one edge from F and $|C \cap A| \geq 2$. Let v be a vertex in $C \cap B$. According to Lemma 2.12, each edge $e \in F$ is a chord in a chordless cycle C_e of G whose vertices are in A. Since F is a minimal triangulation of G_A, v is not an endpoint of any edge in F. Using Lemma 2.9 we conclude that there must be a chordless cycle with v on it in G, contradicting the assumptions of the theorem.

To prove the other direction, let F be a minimal triangulation of G. There exists $F' \subseteq F$ that is a minimal triangulation of G_A. According to the first part of the proof, F' also triangulates G. Since F is minimal, we conclude that $F' = F$. ☐

2.2.3. Overall running time. The final step of the algorithm is to look for $(k - a)$-triangulations in vertex set A, as justified by Theorem 2.13. One can find one or all such triangulations by the algorithm described in section 2.1. Since the size of A is $O(k^3)$, the running time for this step is $O(k^6 2^{4k})$. The total time for the three-step partitioning process is $O(k^2 nm)$, giving a time bound for the entire algorithm of $O(k^2 nm + k^6 2^{4k})$. This algorithm has a better bound than the simple algorithm only for $k = \Omega(\log n)$. It would be a nice result to obtain an algorithm with a running time of $O(km + f(k))$. We leave this as an open problem.

3. Proper interval completion.

A proper interval supergraph $G = (V, E \cup F)$ of a graph $G = (V, E)$ with $|F| \leq k$ is called a *k-proper interval supergraph* of G.

The algorithm presented in section 2.1 can be easily modified to produce all possible k-proper interval supergraphs of a graph, using the following observations. Proper interval graphs are exactly the chordal graphs that do not contain any of the three obstructions in Figure 3.1 as an induced subgraph [40]. Deng, Hell, and Huang [8] have recently described an algorithm that checks whether a graph G is a proper interval graph. In case G is indeed a proper interval graph the algorithm can provide a proper interval representation for G. The running time of the algorithm is $O(m)$, and it does not use complicated data structures such as PQ-trees [5]. It is

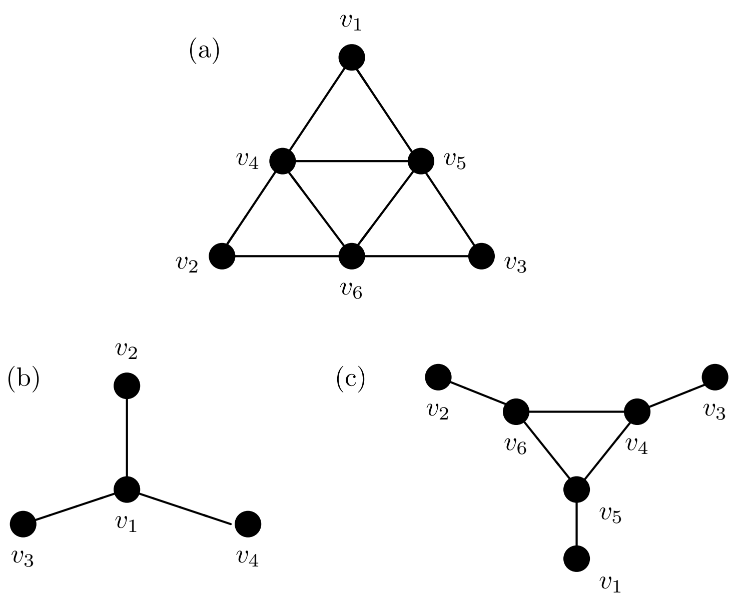

FIG. 3.1. *Obstructions for chordal graphs that are not proper interval.* (a) *Tent.* (b) *Claw.* (c) *Net.*

straightforward to check that in case the input graph is not a proper interval graph one can use the information maintained by the algorithm to extract either a chordless cycle or one of the obstructions in Figure 3.1 in linear time.

The k-completion algorithm will traverse part of a search tree defined as follows. The graph G itself corresponds to the root of the tree. Let x be a node of the search tree corresponding to a supergraph G_x of G that is not a proper interval graph. The children of x are obtained as follows. The algorithm by Deng, Hell, and Huang is applied to G_x to find either a chordless cycle or one of the obstructions in Figure 3.1. If a chordless cycle C is found in G_x, then every minimal triangulation of C gives rise to a child of x as in section 2.1. In case an obstruction is found, x has a child for every edge e between vertices of the obstruction that is not part of the obstruction. The supergraph corresponding to such a child is $G_x \cup \{e\}$. Thus if the obstruction found is a tent the node has six children, if it is a claw it has three, and if it is a net it has nine.

Each leaf in the search tree thus defined corresponds to a proper interval supergraph of G. Note that every minimal proper interval supergraph of G is represented by at least one leaf. As in section 2.1, the nodes of the search tree that are actually traversed correspond to supergraphs with no more than k additional edges. If one such node is a leaf, then we have found a k-proper interval supergraph. Otherwise, no such supergraph exists.

We summarize the result presented in this section in the following theorem. Its proof is analogous to the proof of Theorem 2.4 and is hence omitted.

THEOREM 3.1. *All k-proper interval supergraphs of a graph can be found in $O(2^{4k}m)$-time.*

Remark. Rose, Tarjan, and Lueker proved that if $G = (V, E)$ is triangulated and $G = (V, E \cup F)$ with $F \neq \emptyset$, $F \cap E = \emptyset$ is triangulated, then there exists an edge $e \in F$

such that $G = (V, E \cup \{e\})$ is also triangulated [32, Lemma 2]. Using this lemma, while traversing the search tree as described above one can avoid generating non-triangulated children of nodes that correspond to triangulations of G. Each minimal proper interval completion of G is still guaranteed to be represented by at least one leaf. In this version of the algorithm one uses the MCS algorithm to detect chordality and find a chordless cycle as long as a chordal supergraph has not been reached. When reaching a chordal supergraph, the algorithm by Deng, Hell, and Huang is applied to get one of the obstructions in Figure 3.1. The children of the node are then generated as described above. Finally, the MCS algorithm is applied to each of the children in order to avoid traversing those that are not chordal. Those that are chordal are further expanded. Such an implementation would use the algorithm of Deng, Hell, and Huang only on chordal graphs and hence a somewhat simpler version of it would suffice.

4. Strongly chordal completion.

A chord (v, w) in an even cycle C is *odd* if the paths connecting v and w on C contain an odd number of edges.

The following characterization of strongly chordal graphs is due to Farber [12].

THEOREM 4.1. *A graph G is strongly chordal iff G is chordal and every even cycle of length at least six in G has an odd chord.*

An odd chord in an even cycle C partitions C into two smaller even cycles C_1 and C_2. Any odd chord in C_1 or C_2 is an odd chord in C as well. A 4-*cycle decomposition* of an even chordless cycle C is a minimal set T of odd chords in C such that there is no induced even chordless cycle of length at least six in $C + T$.

Next we characterize and count the number of minimal 4-cycle decompositions of an even cycle C. Let $|C| = n$.

The proof of the following lemma is straightforward by induction.

LEMMA 4.2. *A minimal 4-cycle decomposition T of an even n-cycle C consists of $(\frac{n}{2} - 2)$ chords. It partitions C into $(\frac{n}{2} - 1)$ 4-cycles. Every two of these 4-cycles are either disjoint or share a chord. Every chord is shared by exactly two 4-cycles.*

A *ternary tree* is a tree in which each internal node has three children. The following theorem establishes a correspondence between the set of 4-cycle decompositions of an even n-cycle and the set of ternary trees with $n - 1$ leaves and $\frac{n}{2} - 1$ internal nodes. This correspondence is similar to the one stated in Lemma 2.2 between minimal triangulations of a chordless n-cycle and binary trees with $n - 1$ leaves.

LEMMA 4.3. *The number of 4-cycle decompositions of an even n-cycle C is equal to the number of ternary trees with $\frac{n}{2} - 1$ internal nodes.*

Proof. For every even n-cycle C, construct an invertible mapping from the set of 4-cycle decompositions of C to the set of ternary trees with $\frac{n}{2} - 1$ internal nodes, as follows. The construction is by induction on the length of the cycle. Assume that one has constructed an invertible mapping for every cycle C', where $|C'| \leq n - 2$. Let C be an n-cycle, and let e be a fixed edge on C. Let T be a 4-cycle decomposition of C, and let $C_e = \{e, e_1, e_2, e_3\}$ be the 4-cycle in $C + T$ which includes e. If e_i, $i \in \{1, 2, 3\}$ is a chord, let C_i be the cycle $C - C_e + \{e_i\}$. The 4-cycle decomposition T induces a 4-cycle decomposition T_i of C_i. The tree which corresponds to T has a root (associated with the edge e); the ith child of the root is a leaf if $e_i \in C$ or the root of the ternary tree which corresponds to T_i under the mapping associated with C_i if e_i is a chord. It is straightforward to verify that the mapping defined above is indeed invertible. $\square$

Denote the number of ternary trees with n internal nodes by t_n. The value t_n satisfies the following recurrence: $t_0 = 1$ and $t_n = \Sigma_{\{i+j+k=n-1\}} t_i t_j t_k$ if $n \geq$

1. According to Graham, Knuth, and Patashnik [19, p. 349], the solution to this recurrence is

$$t_n = \binom{3n+1}{n} \frac{1}{3n+1},$$

which is no greater than $2^{3n} = 8^n$. Together with Lemma 4.3 we obtain the following.

LEMMA 4.4. *The number of* 4*-cycle decompositions of an even n-cycle C is no greater than* $8^{\frac{n}{2}-1}$.

4.1. Finding an even cycle without odd chords. The *neighborhood matrix* of a graph is a symmetric 0-1 matrix with rows and columns indexed by the set of vertices of the graph and with an entry of 1 iff the corresponding two vertices are equal or adjacent in the graph. A *doubly lexical ordering* of a matrix is an ordering of the rows and of the columns so that the rows, as vectors, are lexically increasing and the columns, as vectors, are lexically increasing. *Lexical ordering* of vectors is the standard dictionary ordering, except that vectors will be read from highest to lowest coordinate. Thus row vectors will be compared from right to left and column vectors from bottom to top. A matrix M is *symmetric* if its rows and columns are indexed by the same set S and $M(s,t) = M(t,s)$ for all $s,t \in S$. A *symmetric ordering* of such an M is an ordering of S. It is not true that every symmetric matrix has a symmetric doubly lexical ordering. But it was proved by Lubiw [26] that a symmetric matrix that has a *dominant diagonal*, meaning that $M(s,s) \geq M(s,t)$ for all $s,t \in S$, has a symmetric doubly lexical ordering. In particular, the neighborhood matrix of any graph has a symmetric doubly lexical ordering.

A *cycle* matrix is a 0-1 $n \times n$ matrix, $n \geq 3$, with exactly two 1's in each row and in each column and such that no proper submatrix has this property. A *totally balanced matrix* is a 0-1 matrix with no cycle submatrices.

Farber [12] proved the following characterization of strongly chordal graphs.

THEOREM 4.5. *A graph is strongly chordal iff its neighborhood matrix is totally balanced.*

A Γ is an ordered 0-1-valued 2×2 matrix with exactly one 0, in the bottom right corner:

$$\Gamma = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

Lubiw proved the following property [26, Theorem 5.2] of a doubly lexical 0-1 matrix M with rows R and columns C.

THEOREM 4.6. *Let M be an ordered doubly lexical* 0-1 *matrix with rows R and columns C. Any 2×2 submatrix of M formed by $r_1 < r_2 \in R$ and $c_1 < c_2 \in C$ with $M(r_1,c_2) = M(r_2,c_1) = 1$, $M(r_2,c_2) = 0$ is, for some $k \geq 3$, embedded in a $k \times k$ submatrix of M formed by $r_1 < r_2 < \cdots < r_k \in R$ and $c_1 < c_2 < \cdots < c_k \in C$ with $M(r_i,c_{i+1}) = M(r_{i+1},c_i) = 1$ for $i = 1,\ldots,k-1$, $M(r_k,c_k) = 1$, and $M(r_i,c_j) = 0$ for other i,j except possibly $i = j = 1$. In particular, any Γ submatrix is embedded in a cycle submatrix. See Figure* 4.1.

Together with the observation that in any ordering of a cycle submatrix there is a Γ submatrix, Theorem 4.6 reestablishes the following result.

THEOREM 4.7 (see [20, 2]). *A* 0-1 *matrix has a Γ-free ordering iff it is totally balanced. Moreover, a doubly lexical ordering of a totally balanced matrix is Γ-free.*

The following theorem makes a link between cycle submatrices in a neighborhood matrix of a graph G and chordless cycles or even cycles without odd chords in G.

$$\begin{array}{c|cccccccccc|}
 & c_1 & c_2 & c_3 & c_4 & & & c_i & & & c_k \\
\hline
r_1 & ? & 1 & 0 & 0 & & & & & & \\
r_2 & 1 & 0 & 1 & 0 & & & & & & \\
r_3 & 0 & 1 & 0 & 1 & & & 0 & & & \\
r_4 & 0 & 0 & 1 & 0 & . & & & & & \\
 & & & & . & . & . & & & & \\
 & & & & & . & 0 & 1 & & & \\
r_i & & & & & & 1 & 0 & . & & \\
 & & & 0 & & & & . & . & . & \\
 & & & & & & & & . & 0 & 1 \\
r_k & & & & & & & & & 1 & 1 \\
\hline
\end{array}$$

FIG. 4.1. *Every Γ submatrix can be embedded in a cycle submatrix.*

THEOREM 4.8. *Let M be a neighborhood matrix of a graph G and let N be a $k \times k$ cycle submatrix of M with rows $r_1 < r_2 < \cdots < r_k$ and columns $c_1 < c_2 \cdots < c_k$. Let $V_N = \{v_l \mid l = r_i$ or $l = c_j, 1 \leq i, j \leq k\}$. Then either the vertices of V_N form an even cycle without odd chords or there exists a subset $C \subseteq V_N$ that induces a chordless cycle.*

Proof. If $r_i \neq c_j$ for every $1 \leq i, j \leq k$, V_N clearly forms an even cycle without odd chords. Assume $r_i = c_j$ for some i and j. This implies that $N(i,j) = M(r_i, c_j) = 1$. Let i' be the other row in which column j has a 1 and let j' be the other column in which row i has a 1. $N(i', j') = 0$ since otherwise we get a contradiction to the fact that N is a cycle submatrix. Thus $r_{i'} \neq c_{j'}$. Among the vertices in V_N, v_{r_i} is adjacent only to $v_{r_{i'}}$ and $v_{c_{j'}}$. These two are not adjacent, but there is a path connecting them in $V - \{v_{r_i}\}$. Thus there exists a chordless cycle $C \subseteq V_N$ through v_{r_i}. ∎

Let M be a symmetric $n \times n$ neighborhood matrix of a connected graph G with m edges and n vertices. Using Paige and Tarjan's implementation [29] of the algorithm described by Lubiw [26], one can obtain a doubly lexical ordering of M in $O(m \log n)$-time. Lubiw [26] also shows how to search for a Γ submatrix in a doubly lexically ordered M in $O(m)$-time. Given a Γ submatrix in a doubly lexically ordered M, a cycle submatrix that contains it can also be found in $O(m)$-time [26]. According to Theorem 4.8, either the rows and columns of this cycle submatrix induce an even cycle without odd chords or a subset of them induce a chordless cycle in G. As suggested by the proof of Theorem 4.8, this cycle can be extracted from the cycle submatrix in $O(m)$-time.

4.2. The k-completion algorithm. As in sections 2.1 and 3, the k-completion algorithm will traverse part of a search tree in which each node corresponds to a supergraph of G. This search tree is defined as follows. The graph G itself corresponds to the root of the tree. In order to generate the children of an internal node x that corresponds to a graph G', one needs to find either a chordless cycle or an even cycle without odd chords in G'. In case a chordless cycle C is found, node x will have a child for each minimal triangulation T of C. If an even cycle without odd chords, C, is found, x will have a child for each 4-cycle decomposition of C. The graph corresponding to a child is obtained by adding the corresponding minimal triangulation or 4-cycle decomposition to G'. If C is a chordless l-cycle, by Lemma 2.3 node x will have at most c_{l-2} children. If C is an even l-cycle without odd chords, then x will have $t_{\frac{l}{2}-1}$ children. Each leaf of the tree corresponds to a strongly chordal supergraph of G. Note that every such supergraph of G that is minimal is represented by at least one leaf.

Remark. In the case that a chordless cycle C is found in the graph corresponding to a node x, it will be more efficient to generate a child only for each triangulation T of C such that $C + T$ has no even cycles without odd chords.

One can find a chordless cycle C in a nonchordal graph with m edges and n vertices in $O(m)$-time by using the MCS algorithm described in [37, 38]. An even cycle without odd chords can be found in a chordal graph that is not strongly chordal in $O(m \log n)$-time using Paige and Tarjan's implementation [29] of Lubiw's algorithm [26], as described in section 4.1. Obviously, one can use Paige and Tarjan's algorithm for both tasks in order to simplify the implementation while getting some penalty in the performance. The algorithm described in [35] can be easily extended to enumerate all ternary trees with n internal nodes, spending $O(n)$-time for each. Applying Lemma 4.3, one obtains an algorithm that enumerates all 4-cycle decompositions of an even cycle C in $O(|C|)$-time for each. It is straightforward to check that a more involved enumeration procedure that enumerates all minimal strongly chordal triangulations of a chordless even cycle C in $O(|C|)$-time for each could be designed as well, based on the ideas in [35].

The nodes of this search tree that are actually traversed correspond to supergraphs of G with no more than k additional edges. If one such node is a leaf, then we have found a strongly chordal supergraph with no more than k additional edges. Otherwise, no such supergraph exists. The proof of the following theorem is analogous to the proof of Theorem 2.4.

THEOREM 4.9. *All minimal strongly chordal supergraphs of a graph G with no more than k additional edges can be found in $O(8^{2k} m \log n)$-time.*

Remark. An alternative implementation that avoids traversing nonchordal children of chordal supergraphs can be designed as described in the remark at the end of section 3.

Remark. For dense matrices, Spinrad describes a faster algorithm which can obtain a doubly lexical ordering in $O(n^2)$-time [36]. Hence the complexity of the algorithm described above can be improved for dense graphs to $O(8^{2k} \min(n^2, m \log n))$-time.

5. Concluding remarks.

We have presented polynomial algorithms for the fixed-parameter version of three graph completion problems: *chordal completion(k)*, *strongly chordal completion(k)*, and *proper interval completion(k)*. Note that the class of proper interval graphs is a subset of the strongly chordal graphs, which are a subset of the chordal graphs. Our results immediately imply that *chordal completion*, *strongly chordal completion*, and *proper interval completion* have a polynomial-time algorithm when k is part of the input but restricted to be at most logarithmic in the size of the graph.

Another important graph family that we have not discussed in this paper is interval graphs. The *interval completion(k)* problem has an important application in molecular biology, as discussed in section 1. Its NP-completeness was proved in [23]. NP-completeness is also implied by the proof of Yannakakis [41] for *chordal graph completion*, as the graphs generated in that proof are chordal iff they are interval. To date the complexity status of the parametric version of the problem is open. It is not known whether the problem is in FPT or hard for some level of the W-hierarchy. The obstructions that have to exist in a chordal graph that is not interval are described in [25]. An arbitrarily large obstruction X could exist in a graph that is not interval but could be made interval with the addition of any one out of $O(|X|)$ edges. This causes difficulties when one tries to apply the techniques of this paper to this graph class.

When the input is restricted to bounded-degree interval graphs for some fixed bound d, the obstruction size is bounded by $O(d)$ and the search tree technique applies to get a quadratic FPT result using the characterization of [25]. It is an open problem whether this obvious bound can be improved.

For the molecular biology application in physical mapping, one can assume that the ratio of sizes of the largest and the smallest clones is at most a small constant c (in practice, $c = 10$ suffices). Fishburn and Graham [15] (see also [14, Chapter 8]) provided characterizations for interval graphs which have such length restrictions. Their results, together with the characterizations of [25], imply that the obstruction size is $O(c)$ and thus for this case, too, the search tree technique applies and the k-completion problem is FPT.

After a preliminary version of this paper appeared in [21], Cai published a paper [6] that rediscovers our simple search-tree-based algorithm for *chordal completion(k)* (see section 2.1). Using a better-known bound on the lth Catalan number, namely, $c_l = O(4^l/l^{3/2})$, and a lemma showing that $c_{i+1}c_{j+1} \leq c_{i+j+1}$, Cai proves that our algorithm in fact runs in $O((4^k/(k+1)^{3/2})(m+n))$-time. Cai also proves a straightforward generalization of our Theorem 3.1. This generalization says that the parameterized version of the graph modification problem with respect to any graph property that can be characterized by a finite set of forbidden induced subgraphs is FPT. The proof of this generalization is similar to the proof of Theorem 3.1.

REFERENCES

[1] K. Abrahamson, R. Downey, and M. Fellows, *Fixed-parameter intractability* II, in Proc. 10th Symposium on Theoretical Aspects of Computer Science (STACS '93), Lecture Notes in Comput. Sci. 665, Springer–Verlag, Berlin, 1993, pp. 374–385.

[2] R. P. Anstee and M. Farber, *Characterizations of totally balanced matrices*, J. Algorithms, 5 (1984), pp. 215–230.

[3] H. L. Bodlaender, *A linear time algorithm for finding tree-decompositions of small treewidth*, in Proc. 25th ACM Symposium on the Theory of Computing, ACM Press, New York, 1993, pp. 226–234.

[4] H. L. Bodlaender, M. R. Fellows, and M. T. Hallet, *Beyond NP-Completeness for problems of bounded width: Hardness for the W hierarchy (extended abstract)*, in Proc. 26th ACM Symposium on the Theory of Computing, ACM Press, New York, 1994, pp. 449–458.

[5] K. S. Booth and G. S. Lueker, *Testing for the consecutive ones property, interval graphs, and planarity using PQ-tree algorithms*, J. Comput. System Sci., 13 (1976), pp. 335–379.

[6] L. Cai, *Fixed-parameter tractability of graph modification problems for hereditary properties*, Inform. Process. Lett., 58 (1996), pp. 171–176.

[7] A. V. Carrano, *Establishing the order of human chromosome-specific DNA fragments*, in Biotechnology and the Human Genome, A. D. Woodhead and B. J. Barnhart, eds., Plenum Press, New York, 1988, pp. 37–50.

[8] X. Deng, P. Hell, and J. Huang, *Linear-time representation algorithms for proper circular-arc graphs and proper interval graphs*, SIAM J. Comput., 25 (1996), pp. 390–403.

[9] R. G. Downey and M. R. Fellows, *Fixed-parameter intractability*, in Proc. 7th Structure in Complexity Theory Conference (Structures '92), Boston, MA, 1992, IEEE Computer Society Press, Los Alamitos, CA, pp. 36–49.

[10] R. G. Downey and M. R. Fellows, *Fixed-parameter tractability and completeness* III: *Some structural aspects of the W hierarchy*, in Complexity Theory: Current Research (Proc. 1992 Dagstuhl Workshop on Structural Complexity), Cambridge University Press, Cambridge, UK, 1993, pp. 191–226.

[11] R. G. Downey and M. R. Fellows, *Parameterized computational feasibility*, in Complexity Theory: Current Research, K. Ambos-Spies, S. Homer, and U. Schoning, eds., Cambridge University Press, New York, 1993, pp. 166–191.

[12] M. Farber, *Characterizations of strongly chordal graphs*, Discrete Math., 43 (1983), pp. 173–189.

[13] M. Farber, *Domination, independent domination, and duality in strongly chordal graphs*, Discrete Appl. Math., 7 (1984), pp. 115–130.

[14] P. Fishburn, *Interval Orders and Interval Graphs*, John Wiley, New York, 1985.

[15] P. Fishburn and R. L. Graham, *Classes of interval graphs under expanding length restrictions*, J. Graph Theory, 9 (1985), pp. 459–472.

[16] M. L. Fredman, J. Komlós, and E. Szemerédi, *Storing a sparse table with $o(1)$ worst case access time*, J. ACM, 31 (1984), pp. 538–544.

[17] A. George and J. W. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice–Hall, Englewood Cliffs, NJ, 1981.

[18] M. C. Golumbic, H. Kaplan, and R. Shamir, *On the complexity of DNA physical mapping*, Adv. Appl. Math., 15 (1994), pp. 251–261.

[19] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, Addison–Wesley, Reading, MA, 1989.

[20] A. J. Hoffman, A. W. J. Kolen, and M. Sakarovitch, *Totally balanced and greedy matrices*, SIAM J. Alg. Discrete Methods, 6 (1985), pp. 721–730.

[21] H. Kaplan, R. Shamir, and R. E. Tarjan, *Tractability of parameterized completion problems on chordal and interval graphs: Minimum fill-in and physical mapping*, in Proc. 35th Symposium on Foundations of Computer Science, IEEE Computer Science Press, Los Alamitos, CA, 1994, pp. 780–791.

[22] R. M. Karp, *Mapping the genome: Some combinatorial problems arising in molecular biology*, in Proc. 25th Annual ACM Symposium on the Theory of Computing, ACM Press, New York, 1993, pp. 278–285.

[23] T. Kashiwabara and T. Fujisawa, *An NP-complete problem on interval graphs*, in IEEE International Symposium on Circuits and Systems (12th), Institute of Electrical and Electronics Engineers, Piscataway, NJ, 1979, pp. 82–83.

[24] T. Kloks, *Treewidth*, Ph.D. thesis, Dept. of Computer Science, Utrecht University, Utrecht, The Netherlands, 1993.

[25] C. G. Lekkerkerker and J. C. Boland, *Representation of a finite graph by a set of intervals on the real line*, Fund. Math., 51 (1962), pp. 45–64.

[26] A. Lubiw, *Doubly lexical orderings of matrices*, SIAM J. Comput., 16 (1987), pp. 854–879.

[27] R. Nagaraja, *Current approaches to long-range physical mapping of the human genome*, in Techniques for the Analysis of Complex Genomes, R. Anand, ed., Academic Press, London, 1992, pp. 1–18.

[28] T. Ohtsuki, L. K. Cheung, and T. Fujisawa, *Minimal triangulation of a graph and optimal pivoting order in a sparse matrix*, J. Math. Anal. Appl., 54 (1976), pp. 622–633.

[29] R. Paige and R. E. Tarjan, *Three partition refinement algorithms*, SIAM J. Comput., 16 (1987), pp. 973–989.

[30] F. S. Roberts, *Indifference graphs*, in Proof Techniques in Graph Theory, F. Harary, ed., Academic Press, New York, 1969, pp. 139–146.

[31] D. J. Rose, *Triangulated graphs and the elimination process*, J. Math. Anal. Appl., 32 (1970), pp. 597–609.

[32] D. J. Rose, R. E. Tarjan, and G. S. Lueker, *Algorithmic aspects of vertex elimination on graphs*, SIAM J. Comput., 5 (1976), pp. 266–283.

[33] J. D. Rose, *A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations*, in Graph Theory and Computing, R. C. Reed, ed., Academic Press, New York, 1972, pp. 183–217.

[34] D. D. Sleator, R. E. Tarjan, and W. P. Thurston, *Rotation distance, triangulations, and hyperbolic geometry*, J. AMS, 1 (1988), pp. 647–681.

[35] M. Solomon and R. A. Finkel, *A note on enumerating binary trees*, J. ACM, 27 (1980), pp. 3–5.

[36] J. Spinrad, *Doubly lexical ordering of dense* 0-1 *matrices*, Inform. Process. Lett., 45 (1993), pp. 229–235.

[37] R. E. Tarjan and M. Yannakakis, *Simple linear-time algorithms to test chordality of graphs, text acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs*, SIAM J. Comput., 13 (1984), pp. 566–579.

[38] R. E. Tarjan and M. Yannakakis, *Addendum: Simple linear-time algorithms to test chordality of graphs, text acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs*,

SIAM J. Comput., 14 (1985), pp. 254–255.

[39] M. N. Wegman and J. L. Carter, *New classes and applications of hash functions*, in Proc. 20th IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 1979, pp. 175–182.

[40] G. Wegner, *Eigenschaften der Nerven Homologische Eihfacher Familien in* R^n, Ph.D. thesis, Götingen University, Götingen, Germany, 1967.

[41] M. Yannakakis, *Computing the minimum fill-in is NP-complete*, SIAM J. Alg. Discrete Methods, 2 (1981), pp. 77–79.