Succinct Data Structures for Bounded Degree/Chromatic Number Interval Graphs

Meng He*, J. Ian Munro†, and Kaiyu Wu*

*Faculty of Computer Science
Dalhousie University
Halifax, NS, Canada
{mhe@cs.,kevin.wu@}dal.ca

†Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada
imunro@uwaterloo.ca

Abstract

An interval graph is the intersection graph of intervals on the real line. We consider the problem of constructing space efficient data structures for two subclasses of interval graphs: those with maximum degree σ_1 and those with chromatic number at most σ_2.

We show that both bounded degree and bounded chromatic number interval graphs have a tight lower bound of $n \lg \sigma_i - o(n \lg \sigma_i)$ bits ($i = 1, 2$). This improves the lower bound of Chakraborty and Jo from $\frac{1}{6} n \lg \sigma_i - O(n)$. For bounded chromatic number interval graphs, we give the first succinct data structure occupying $n \lg \sigma_2 + O(n)$ bits that supports navigational operations and distance queries in $O(\sigma_2 \lg n)$ time. To match Chakraborty and Jo's time complexity of $O(\lg \lg \sigma_2)$, which uses $(\sigma_2 - 1)n + O(n)$ bits, we use $2n \lg \sigma_2 + O(n)$ bits instead.

1 Introduction

We examine space efficient representations of several subclasses of interval graphs. In particular, a succinct data structure for a class of combinatorial objects (for example ordinal trees on n nodes) with N members uses $\lg N + o(\lg N)$ bits in the worst case and a compact data structure uses $\Theta(\lg N)$ bits in the worst case. An interval graph is a graph where we can assign an interval $[l_v, r_v]$ to each vertex v, and edges are implicitly defined whenever the intervals of two vertices intersect. Interval graphs are a well-known class of graphs and have applications in operations research [1] and bioinformatics [2]. For a more in-depth treatment of interval graphs and their applications, see the book of Golumbic [3]

There are several prior works focusing on succinct data structures for various subclasses of graphs with nice combinatorial properties. First is the paper we are improving upon by Chakraborty and Jo [4] which considers interval graphs with bounded degree or chromatic number. For other classes of graphs that have been considered in the literature, we have [5, 6] for interval graphs, proper interval graphs and circular arc graphs. [7] for path graphs, [8] for chordal graphs, [9] deals with graphs of bounded treewidth, [10] covers series-parallel, block-cactus and 3-leaf power graphs, and [11] focuses on permutation graphs.

The queries supported by these graph data structures are the standard navigational operations: `adjacency`: given two vertices return whether they are adjacent; `degree`: given a vertex, return its degree; `neighbourhood`: given a vertex, return all vertices adjacent to it. Some of the data structures also support distance related

Author	Lower Bound	Space	`adjacency`	`neighbourhood`	`degree`	`dist`[a]
Bounded Degree Interval Graph						
[4]	$\frac{1}{6}n \lg \sigma_1$	$n \lg \sigma_1$	$O(1)$	$O(d)$	$O(1)$	-
Thm. 5, 6	$n \lg \sigma_1$	$n \lg \sigma_1$	$O(1)$	$O(d)$	$O(1)$	$O(1)$
Bounded Chromatic Number Interval Graph						
[4]	$\frac{1}{6}n \lg \sigma_2$	$(\sigma_2 - 1)n$	$O(\lg \lg \sigma_2)$	$O(\sigma_2 \lg \lg \sigma_2 + d)$	$O(\sigma_2 \lg \lg \sigma_2)$	-
Thm. 4, 7	$n \lg \sigma_2$	$n \lg \sigma_2$	$O(\sigma_2 \lg n)$	$O(d\sigma_2 \lg n)$	$O(\sigma_2 \lg n)$	$O(\sigma_2 \lg n)$
Thm. 8	*	$2n \lg \sigma_2$	$O(\lg \lg \sigma_2)$	$O(\sigma_2 \lg \lg \sigma_2 + d)$	$O(\lg \lg \sigma_2)$	$O(\lg \lg \sigma_2)$

[a]We use - to denote that it is not supported. When it is supported, `shortest_path` is also supported using an additional $O(1)$ time per vertex returned

Table 1: Comparison of our results with previous results. Here d denotes the degree of the vertex. Lower order terms in the lower bound and the space are omitted.

operations: `shortest_path`: given two vertices, return a shortest path between them; and `dist`: given two vertices, return the length of a shortest path between them.

1.1 Our Results

Our results are summarized in Table 1.

In section 3, we improve the lower bound for bounded degree (maximum degree bounded by σ_1) and bounded chromatic number (chromatic number at most σ_2) interval graphs, from $\frac{1}{6}n \lg \sigma_i - O(n)$ $(i = 1, 2)$ to $n \lg \sigma_i - o(n \lg \sigma_i)$. Furthermore, using our techniques, we give an alternative lower bound proof for general interval graphs of $n \lg n - O(n \lg \lg n)$ bits [5, 12].

In section 4, we give succinct and compact data structures for bounded chromatic number interval graphs. We give a matching upper bound occupying $n \lg \sigma_2$ bits of space, supporting queries in $O(\sigma_2 \lg n)$ time. We then give a compact data structure occupying $2n \lg \sigma_2$ bits of space, supporting queries in $O(\lg \lg \sigma_2)$ time. By applying the theorem of He et. al. [6], we also immediately obtain the `shortest_path` and `dist` operations for bounded degree interval graphs as well.
Due to space constraints, omitted details and proofs can be found in the thesis [13].

2 Preliminaries

In this paper, we will use the standard graph theoretic notation. We will use $G = (V, E)$ to denote a graph with vertex set V and edge set E. We will use $n = |V|$ and $m = |E|$ to denote the number of vertices and edges. All of our graphs will be unweighted. We assume the word-RAM model with $\Theta(\lg n)$-size words. We use $\lg(\cdot)$ to denote $\log_2(\cdot)$.

2.1 Succinct Data Structures

A bit vector B is a length n array of bits, that supports the queries $\texttt{rank}_b(B, i)$: given an index, return the number of b $(b \in \{0, 1\})$ bits up to index i, $\texttt{select}_b(B, j)$: given a number j, return the index of the jth b bit in the array, and $\texttt{access}(B, i)$ (or just

$B[i]$): return the bit at index i. We may generalize this to larger alphabets as well: $\texttt{rank}_c(B, i)$ returns the number of occurrences of the character c up to index i and $\texttt{select}_c(B, j)$ returns the index of the jth occurrence of the character c.

Lemma 1 ([14])**.** *A bit vector of length n can be succinctly represented using $n + o(n)$ bits to support **rank**, **select** and **access** in $O(1)$ time.*

Lemma 2 ([15])**.** *A string S of length n over an alphabet of size σ, can be succinctly represented using $n \lg \sigma + o(n \lg \sigma)$ bits to support **rank**, **access** in $O(\lg \lg \sigma)$ time and **select** in $O(1)$ time.*

2.2 Interval Graphs

A graph G is an interval graph if we can find an interval $I_v = [l_v, r_v]$ for every vertex v such that two vertices u, v are adjacent if and only if the corresponding intervals intersect: $I_u \cap I_v \neq \emptyset$. We will say that a collection of intervals $\mathcal{I}$ is an *intersection model* for the interval graph G. Given an intersection model for G, we may sort the endpoints so that they lie in the range $[1, 2n]$ and that endpoints are all distinct while preserving the intersection structure. A maximal clique is a clique that is maximal under inclusions. It is known that for an interval graph G, the number of maximal cliques of G, k is at most n [16]. Furthermore, we may arrange the maximal cliques of G linearly, so that for each vertex v, the maximal cliques of G containing v are consecutive. By numbering the maximal cliques as $1, \ldots, k$, we may retrieve an interval for a vertex v, by setting l_v as the first maximal clique v appears in and r_v as the last. It is not hard to see that this is indeed an intersection model for G as two intervals intersect $\Leftrightarrow$ the vertices both belong to some maximal clique $\Leftrightarrow$ the vertices are adjacent.

Given an intersection model for an interval graph G with k maximal cliques, we may read off the maximal cliques of G by selecting a set of integers $i_1 < i_2 < \cdots < i_k$ and creating maximal cliques $C_j = \{v; i_j \in [l_v, r_v]\}$ which are the intervals containing each integer. To determine the values of i_j, we use the following lemma:

Lemma 3. *Let $\mathcal{I}$ be an intersection model of G, with the intervals sorted. For an integer i, if the endpoint at position i is a right endpoint, and the endpoint at position $i - 1$ is a left endpoint, then the clique $C = \{v; i \in [l_v, r_v]\} = \{v; i - 1 \in [l_v, r_v]\}$ is a maximal clique. Moreover, all maximal cliques are found in this way.*

The subclasses of interval graphs we are interested in are those of bounded degree (denoted by σ_1) and those of bounded chromatic number (denoted by σ_2). As interval graphs are perfect graphs, the chromatic number of the graph is equal to the size of the maximum clique. Again this is not hard to see, as interval graphs can be optimally colored using a greedy algorithm - by scanning from left to right, and whenever a new interval appears, colour it with the smallest currently unused colour of its current neighbours. To construct data structures for these subclasses, we will consider the following abstract statements of the Theorems of Acan et al. [5] and He et al. [6].

Corollary 1. *Let G be an interval graph. Let D be a data structure that can compute the right endpoint of the ith interval using $g(n)$ bits of space and $f(n)$ time. Then there is a data structure occupying $g(n) + 6n$ bits of space[1] supporting* `adjacency`, `degree`, `dist` *in $O(f(n))$ time,* `neighbourhood` *in $O(f(n))$ time per neighbour and* `shortest_path` *in $O(f(n) + d)$ time where d is the distance between the two vertices.*

3 Lower Bounds

In this section, we derive lower bounds of $n \lg \sigma_i - o(n \lg \sigma_i)$ $(i = 1, 2)$ for both bounded degree interval graphs and bounded chromatic number interval graphs. As interval graphs are defined by the existence of an intersection model, we will construct graphs by exhibiting those. To avoid overcounting, we will construct graphs that only have few intersection models.

Given an sorted intersection model of G, we may obtain an ordering of maximal cliques $C_1, \ldots, C_k$ of G using Lemma 3. This ordering of maximal cliques has the property that for any vertex v, the maximal cliques containing v are consecutive - which we will call the *consecutive clique property.* We may obtain an intersection model for G, by numbering the maximal cliques $1, \ldots, k$ and assigning the interval $[l_v, r_v]$ to each vertex v where l_v is the first maximal clique that v appears in and r_v is the last maximal clique that v appears in. [2]

Given an ordering of maximal cliques of G with the consecutive clique property, we will say that the interval for v is $I_v = [l_v, r_v]$ as described. We will define the support $\mathtt{supp}(v)$ as $[l_v, r_v]$. For a subset of the vertices $V_1 \subseteq V$, we will extend the definition of support to $\mathtt{supp}(V_1) = \cup_{v \in V_1} \mathtt{supp}(v)$.

For vertices u, v, we say that they *overlap* if $I_v \cap I_u \notin \{I_v, I_u, \emptyset\}$ (note that this is independent of the ordering of maximal cliques). That is they overlap if they intersect but one does not contain the other. For an interval graph G, denote the overlap graph $Ov(G)$ on the same vertex set, but rather than using intersection of the supports as the adjacency criterion, we use overlapping as the criterion. Thus $(u, v) \in E(Ov(G))$ if u, v overlap. As u, v overlap implies that they intersect, $Ov(G)$ is a subgraph of G. In particular, any connected component of $Ov(G)$ is also connected in G. The theorem of Köbler et al. [17] gives a condition for the number of orderings of maximal cliques of G to be at most 2.

Theorem 1. *If $Ov(G)$ has a single component, then the number of orderings of the maximal cliques of G satisfying the consecutive clique property is at most 2 - one ordering and the reverse.*

Ultimately, as stated, we will give a way to construct a sequence of maximal cliques satisfying the consecutive clique property. We will show that the graph arising from this sequence of maximal cliques will be close to having a single overlap component

[1] A bit of optimization can cut this down to $g(n) + (5 + \varepsilon)n$ bits for $\varepsilon > 0$.

[2] Conversely, it is known and easy to see that a graph G is an interval graph if the maximal cliques of G can be linearly ordered to have the consecutive clique property [16]

so that we can apply the above lemma. In doing so, we will show that we do not construct each graph more than twice. In order to show that our graph is close to having a single overlap component, we will first need to prove a few structural lemmas on the properties of overlap components of an interval graph.

Lemma 4. *Let V_1 be a connected component of the overlap graph of an interval graph G, then $\textbf{\textit{supp}}(V_1)$ is a single interval $[l, r]$.*

Lemma 5. *Let V_1 with support $[l, r]$ such that $l \neq r$ be a connected component of the overlap graph of an interval graph. Then for any value $x \in [l, r]$, there exists a vertex $v \in V_1$ such that $l_v \leq x \leq r_v$ with $l_v \neq r_v$. Furthermore, if $x \neq l$, then we may choose $l_v < x \leq r_v$.*

Proof. First we note that no vertex can have $l_v = r_v$, since it would not overlap any other vertex. In the case that $x = l$, then as the support of V_1 is $[l, r]$, there exists a vertex v with $l_v = l$ and $r_v > l$. Now suppose that $x \neq l$ and suppose that no such vertex can be found. Consider the two sets of vertices: $\{v; l_v < x\}$ and $\{v; l_v \geq x\}$ whose union is V_1. Note that for the first set, any vertex must have $r_v < x$, otherwise we may pick that vertex. But then the support of the first set is contained in $[l, x-1]$ and the second is $[x, r]$, whose union does not contain the interval $(x-1, x)$, contradicting Lemma 4. □

Corollary 2. *Let V_1 be a connected component of the overlap graph of an interval graph with support $[l, r]$, with more than 1 vertex. Then we may find a sequence of vertices $v_1, v_2, \ldots$ with the following properties: $l = l_{v_1} < l_{v_2} < l_{v_3} \ldots$, and for each vertex: $l_{v_i} < l_{v_{i+1}} \leq r_{v_i}$.*

Proof. We repeatedly apply Lemma 5. Stab $[l, r]$ with the value r to obtain the last vertex in the chain v_p. Stab $[l, r]$ with the value l_{v_p} to obtain v_{p-1} with the property that $l_{v_{i-1}} < l_{v_i} \leq r_{v_{i-1}}$, and repeat. □

These lemmas state that we may find a sequence of overlapping intervals to cover the support of overlap component. Next we show how two overlap connected components interact. This matches our intuition that two components must either be completely disjoint or one is contained in the other. We cannot have the components overlap each other (as that would mean they would merge into one component).

Lemma 6. *Let V_1, V_2 be two connected components of $Ov(G)$ where G is an interval graph. Let $\textbf{\textit{supp}}(V_1) = [l_1, r_1]$ and $\textbf{\textit{supp}}(V_2) = [l_2, r_2]$. Then either:*
- $[l_1, r_1]$ does not intersect $[l_2, r_2]$;
- $[l_1, r_1] \subseteq [l_2, r_2]$ and there exists a vertex $u \in V_2$ such that $[l_1, r_1] \subseteq [l_u, r_u]$. (or the symmetric case)

With all of our structural lemmas complete we now give our class of interval graphs by defining our maximal cliques. Fix Σ and we will label our vertices by the integers 1 to n. Let $C_1 = \{1, \ldots, \Sigma\}, C_2 = \{2, \ldots, \Sigma + 1\}$, $C_\Sigma = \{\Sigma, \ldots, 2\Sigma - 1\}$ and finally $C_{\Sigma+1} = \{\Sigma + 1, \ldots, 2\Sigma\}$. In this "start-up" region, we simply remove the smallest vertex from the maximal clique, then add the next vertex to obtain the next maximal

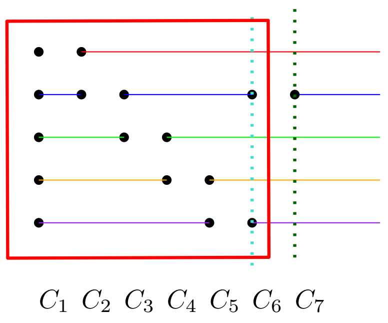

Figure 1: Construction of our class of interval graphs, with $\Sigma = 5$.

clique. For $C_{\Sigma+2}$ to $C_{n-\Sigma+1}$ (the rest of the cliques) we apply the following operation to get C_{i+1} from C_i: choose an arbitrary element of C_i that is in the smaller half of values and remove it, then add the next vertex (i.e. $i + \Sigma$). For example, to go from $C_{\Sigma+1}$ to $C_{\Sigma+2}$ we delete one of the elements in the smaller half, that is one of $\{\Sigma + 1, \dots, \lceil (3/2)\Sigma \rceil\}$, and add the next vertex, which is $2\Sigma + 1$. It is not hard to see that C_i are indeed the maximal cliques of the graph using Lemma 3.

We will now show that there are at most 2 ways to order the maximal cliques so that it satisfies the consecutive clique property.

Theorem 2. *Let G be an interval graph obtained in the algorithm above. Then there are at most two orderings of the maximal cliques which satisfy the consecutive clique property.*

Proof. First we note that there are exactly two vertices with the same left and right endpoints: $\texttt{supp}(v_1) = [1, 1]$ and $\texttt{supp}(v_n) = [n - \Sigma + 1, n - \Sigma + 1]$. Furthermore, if we remove them and try to re-add them, to maintain the same graph, v_1 must go in the first clique and v_n must go in the last clique. Thus if we show the rest of the graph is overlap connected, then by Theorem 1 there are at most two ways to order the cliques and satisfy the consecutive clique property.

Consider any component V_1 with support $[l, r]$. We note that $l \neq r$, since no vertex has the same left and right endpoints any more. We claim that $l = 1$ and $r = n - \sigma + 1$. Suppose for a contradiction that $r \neq n - \Sigma + 1$, then consider the vertex $v_{r+\Sigma-1}$ which is added in C_r. Since we do not remove it this clique, $r_{v_{r+\Sigma-1}} > r$. Furthermore, since $\texttt{supp}(V_1) = [l, r]$, there exists a vertex $v \in V_1$ with $l_v < r_v = r$. These two vertices overlap so $v_{r+\Sigma+1} \in V_1$ and $\texttt{supp}(V_1) \neq [l, r]$. To show that $l = 1$ is similar by considering the vertex that ends in C_l, which must overlap the vertex $v \in V_1$ that has $l = l_v < r_v$.

Now suppose that $G \setminus \{v_1, v_n\}$ were not overlap connected, and let V_1, V_2 be two overlap connected components. By above, we must have $\texttt{supp}(V_1) = \texttt{supp}(V_2) = [1, n - \sigma + 1]$. By Lemma 6, there exists a vertex $u \in V_2$ such that $[1, n - \sigma + 1] \subseteq \texttt{supp}(u)$. However, by construction, no vertex that starts in C_1 can end in $C_{n-\Sigma+1}$, a contradiction.

Therefore $G \setminus \{v_1, v_n\}$ is overlap connected and we are done. $\square$

With this, we are able to show lower bounds for many classes of interval graphs. We will count the number of graphs we are able to construct using the above construction, subject to certain restrictions (for instance, the size of the maximal cliques σ). The way we will do this is to notice that we make a decision at every clique: which vertex in the smaller half to remove. Every decision will generate a different graph depending on the vertex removed. By counting the number of outcomes over all the decision, we obtain the number of different graphs created.

The first lower bound we will show using the above construction is an alternative lower bound proof for interval graphs of Acan et al. and Gavoille and Paul [5, 12] by setting $\Sigma = n/\lg n$.

Theorem 3. *Representing interval graphs needs $n \lg n - O(n \lg\lg n)$ bits.*

Proof. The number of graphs that can be created using the above algorithm is the following. We make choices at $n - 2\Sigma$ cliques - to determine which vertex should end at that clique. Each of these choices is between $\Sigma/2$ elements. Thus the number of graphs we obtain is $(\Sigma/2)^{(n-2\Sigma)}$. [3] Hence the information theoretic lower bound is: $\lg{(\Sigma/2)}^{(n-2\Sigma)} = (n-2\Sigma)\lg(\Sigma/2)$. Setting $\Sigma = n/\lg n$ completes the proof. □

Next we will apply this construction to improve the lower bound results of [4] on bounded degree and bounded chromatic number interval graphs.

Theorem 4. *Representing interval graphs with bounded chromatic number σ needs $n \lg \sigma_2 - O(\sigma_2 \lg \sigma_2 + n)$ bits. (or equal to $n \lg n - o(n \lg n)$ when $\sigma_2 \geq n/\lg n$)*

Proof. As interval graphs are perfect graphs, the chromatic number is equal to the size of the maximum clique. By construction, all graphs G have maximum clique size Σ, so we set $\sigma_2 = \Sigma$. Thus the lower bound is simply $(n - 2\sigma_2)\lg(\sigma_2/2) = n \lg \sigma_2 - O(\sigma_2 \lg \sigma_2 + n)$.

Finally we note that this only applies when $\sigma_2 = O(n/\lg n)$, as larger σ_2 (especially when $\sigma = \Theta(n)$) would cause the lower order term to cancel out with the leading term. But as we have shown that the lower bound is already $n \lg n - o(n \lg n)$ for $\sigma_2 = n/\lg n$, any larger σ_2 would also inherit this lower bound. □

Theorem 5. *Representing interval graphs with bounded degree σ_1 needs $n \lg \sigma_1 - o(n \lg \sigma_1)$ bits.*

Proof. To limit the degree of our vertices, we need to make sure that the clique that any vertex ends on is not too far from the clique that the vertex begins. For a vertex v with $\texttt{supp}(v) = [l_v, r_v]$, we have $\deg(v) = (\Sigma - 1) + (r_v - l_v)$. The first term says that it is adjacent to $\Sigma - 1$ vertices of the clique that it begins at, and the second term says that for each additional clique, it gains one more neighbour.

To limit the size of $r_v - l_v$, rather than selecting an arbitrary vertex to remove, every Δ cliques, we always remove the smallest vertex. Consider the rank of any vertex v in the cliques that contain it (that is, how many vertices are smaller than

[3]We over count by at most a factor of 2, but this is a constant term after taking the lg.

it in the clique?). By construction, in the clique that it begins at, it is rank Σ - the largest numbered vertex (except those in C_1, where their ranks at at most Σ). Since we always remove a vertex in the smaller half, the next $\Sigma/2$ cliques will decrease its rank by 1. Since we remove the smallest vertex in the clique every Δ cliques, we are guaranteed to decrease the rank of v every Δ cliques and thus remove it after at most $\Delta\cdot\Sigma/2$ cliques. Thus $r_v - l_v \leq \Sigma/2 + \Delta\Sigma/2$ and $\deg(v) \leq \Sigma + \Sigma/2 + \Delta\Sigma/2 = \frac{\Sigma}{2}(3+\Delta)$. And thus we set $\sigma_1 = \frac{\Sigma}{2}(3+\Delta)$.

The number of graphs we obtain is also changed. Rather than making $(n-2\Sigma)$ choices, we make $\frac{\Delta-1}{\Delta}(n-2\Sigma)$ choices since we always remove the smallest vertex every Δ cliques. Thus the lower bound is instead: $\frac{\Delta-1}{\Delta}(n-2\Sigma)\lg(\Sigma/2)$.

By setting $\Sigma = 2\sigma_1/\lg\sigma_1$ and $\Delta = \lg(\sigma_1) - 3$, the lower bound we obtain is $n\lg\sigma_1 - O(n\lg\lg\sigma_1)$. □

4 Data Structure for Restricted Interval Graphs

In this section we will consider data structures for bounded degree and bounded chromatic number interval graphs, primarily using the abstract formulation of the theorems of Acan et al. and He et al. [5, 6] stated in Corollary 1

Chakraborty and Jo [4] gave a straightforward and immediate adaptation of Acan et al.'s [5] data structure to apply to bounded degree interval graphs, using $n\lg\sigma_1$ bits. A brief description is as follows. It can be shown that for an interval graph whose degree is bounded by σ_1, then for any interval, the length is $O(\sigma_1)$. Thus by storing the difference $r_v - l_v$, we have $g(n) = n\lg\sigma_1 + O(n)$ and $f(n) = O(1)$.

Thus they obtained the theorem:

Theorem 6. *Let G be an interval graph with bounded degree σ. Then G can be represented using $n\lg\sigma + O(n)$ bits to support the operations* ***`adjacency`***, ***`degree`***, ***`dist`*** *in $O(1)$ time, and* ***`neighbourhood`*** *and* ***`shortest_path`*** *in $O(1)$ time per vertex returned (as a neighbour or on the path)* [4].

As we have shown in the previous section by giving a tight lower bound, this is indeed succinct. We will now focus our attention to bounded chromatic number interval graphs. Chakraborty and Jo [4] gave a data structure using $(\sigma_2 - 1)n$ bits of space and $O(\lg\lg\sigma_2)$ query times. We will first show that our lower bound of $n\lg\sigma_2$ is tight by giving a succinct data structure, with somewhat bad query times. We will then give a compact data structure matching their $O(\lg\lg\sigma_2)$ query times. We will assume that σ_2 is non-constant so that the auxiliary space needed is a lower order term.

4.1 Succinct Data Structure

As in Corollary 1, we will construct the data structure D to compute the right end points, using $n\lg\sigma_2$ bits of space. The idea is given the left endpoint of some interval, we scan the indices corresponding to the right endpoints, and find the one that

[4]Chakraborty and Jo did not include the `dist` query in their result, but as He et al. stated in their paper, the augmentation is fairly universal to all subclasses of interval graphs.

corresponds with our interval. As the maximal cliques are of size at most σ_2, the number of unclosed intervals at any moment is at most σ_2 (all such unclosed intervals are adjacent to each other and must have different colours), thus we may store which unclosed interval (in sorted order by left endpoints, i.e. if we store "5-th" interval, then this end point matches unclosed interval with the 5th smallest left end point.) matches each index that closes an interval in $\lg \sigma$ bits. Thus the total space required is $n \lg \sigma_2 + O(n)$ bits.

To support the navigational operations we will need to be able to find the index of the right endpoint of the interval given the left endpoint of an interval. To do this we use the following algorithm which keeps track of the rank of l_v among the left endpoints of unclosed intervals. Let B be a bit-vector where $B[i] = 0$ if the endpoint at index i is a left endpoint, and $B[i] = 1$ if it is a right endpoint. We note that B is one of the components of the data structure of Corollary 1 so we do not need to store it again. By our definition the rank of the vertex v is the excess (the number of unclosed intervals, found by $\mathtt{rank}_0(B, l_v) - \mathtt{rank}_1(B, l_v))$ at the index of l_v. For each following index that corresponds to the end (right endpoint) of an interval there are 3 cases:
- It closes an interval with rank larger than the current rank: we do nothing.
- It closes an interval with rank equal to the current rank: this is our right endpoint.
- It closes an interval with rank less than the current rank: we decrement our rank.

In this way, we are able to compute index of the right endpoint of the interval in $O(n)$ time. To optimize this, at right endpoint r of some of the intervals, we store the set of vertices whose left endpoints are currently unclosed. We will call these sets shortcuts and we store them at every $\sigma_2 \lg n$ right endpoints. The idea is rather than start the algorithm at the left endpoint of the interval, we start it at the last shortcut set containing our left endpoint - so we are guaranteed to find the matching right endpoint before the next shortcut set - that is after at most $\sigma_2 \lg n$ iterations.

We binary search the shortcut sets to find the last shortcut set containing the left endpoint of our interval. By storing each shortcut set using perfect hashing we may find this set in $O(\sigma \lg n)$ time, while using $O(n)$ extra bits. Thus we have the following theorem:

Theorem 7. *Let G be an interval graph with chromatic number σ_2. Then G may be represented using $n \lg \sigma_2 + O(n)$ bits of space and support* ***adjacency****,* ***degree****,* ***neighbourhood****,* ***dist*** *in $O(\sigma_2 \lg n)$ time (or per neighbour) and* ***shortest_path***(u, v) *in $O(\sigma_2 \lg n +$* ***dist***$(u, v))$ *time.*

For our compact data structure, we colour the graph using σ_2 colours. We consider a string S where $S[i] = c$ is the colour of the vertex v where $i = l_v$ or r_v. Thus to compute r_v from l_v, we simply find the colour c of v using $S[l_v]$, then r_v must be the index of the next occurrence of c in S, which we compute using select in $O(\lg \lg \sigma_2)$ time. Details omitted. Thus we have:

Theorem 8. *Let G be an interval graph with chromatic number σ_2. Then G can be represented using $2n \lg \sigma_2 + o(n \lg \sigma_2)$ bits of space and support* ***adjacency****,* ***degree****,* ***dist*** *in $O(\lg \lg \sigma_2)$ time,* ***neighbourhood***(v) *in $O(\min($**degree**$(v) \lg \lg \sigma_2, \sigma_2 \lg \lg \sigma_2 +$* ***degree***$(v))$ *time and* ***shortest_path***(u, v) *in $O(\lg \lg \sigma_2 +$* ***dist***$(u, v))$ *time.*

5 References

[1] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph Naor, and Baruch Schieber, "A unified approach to approximating resource allocation and scheduling," *J. ACM*, vol. 48, no. 5, pp. 1069–1090, 2001.

[2] Peisen Zhang, Eric A. Schon, Stuart G. Fischer, Eftihia Cayanis, Janie Weiss, Susan Kistler, and Philip E. Bourne, "An algorithm based on graph theory for the assembly of contigs in physical mapping of DNA," *Comput. Appl. Biosci.*, vol. 10, no. 3, pp. 309–317, 1994.

[3] Martin Charles Golumbic, *Algorithmic graph theory and perfect graphs*, Elsevier, 2004.

[4] Sankardeep Chakraborty and Seungbum Jo, "Compact representation of interval graphs of bounded degree and chromatic number," in *Data Compression Conference, DCC 2022, Snowbird, UT, USA, March 22-25, 2022*. 2022, pp. 103–112, IEEE.

[5] Hüseyin Acan, Sankardeep Chakraborty, Seungbum Jo, and Srinivasa Rao Satti, "Succinct encodings for families of interval graphs," *Algorithmica*, vol. 83, no. 3, pp. 776–794, 2021.

[6] Meng He, J. Ian Munro, Yakov Nekrich, Sebastian Wild, and Kaiyu Wu, "Distance oracles for interval graphs via breadth-first rank/select in succinct trees," in *31st International Symposium on Algorithms and Computation, ISAAC 2020,*. 2020, vol. 181 of *LIPIcs*, pp. 25:1–25:18, Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[7] Girish Balakrishnan, N. S. Narayanaswamy, Sankardeep Chakraborty, and Kunihiko Sadakane, "Succinct data structure for path graphs," in *Data Compression Conference, DCC 2022, Snowbird, UT, USA, March 22-25, 2022*. 2022, pp. 262–271, IEEE, Full version: https://arxiv.org/abs/2111.04332.

[8] J. Ian Munro and Kaiyu Wu, "Succinct data structures for chordal graphs," in *29th International Symposium on Algorithms and Computation, ISAAC 2018*. 2018, vol. 123 of *LIPIcs*, pp. 67:1–67:12, Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[9] Arash Farzan and Shahin Kamali, "Compact navigation and distance oracles for graphs with small treewidth," *Algorithmica*, vol. 69, no. 1, pp. 92–116, 2014.

[10] Sankardeep Chakraborty, Seungbum Jo, Kunihiko Sadakane, and Srinivasa Rao Satti, "Succinct data structures for series-parallel, block-cactus and 3-leaf power graphs," in *Combinatorial Optimization and Applications, COCOA 2021*. 2021, vol. 13135 of *Lecture Notes in Computer Science*, pp. 416–430, Springer.

[11] Konstantinos Tsakalidis, Sebastian Wild, and Viktor Zamaraev, "Succinct permutation graphs," *CoRR*, vol. abs/2010.04108, 2020.

[12] Cyril Gavoille and Christophe Paul, "Optimal distance labeling for interval graphs and related graph families," *SIAM Journal on Discrete Mathematics*, vol. 22, no. 3, pp. 1239–1258, Jan. 2008.

[13] Kaiyu Wu, *Succinct and Compact Data Structures for Intersection Graphs*, Ph.D. thesis, University of Waterloo, 2023.

[14] J. Ian Munro, Venkatesh Raman, and S. Srinivasa Rao, "Space efficient suffix trees," *J. Algorithms*, vol. 39, no. 2, pp. 205–222, 2001.

[15] Alexander Golynski, J. Ian Munro, and Srinivasa Rao Satti, "Rank/select operations on large alphabets: a tool for text indexing," 01 2006, pp. 368–373.

[16] Delbert Ray Fulkerson and Oliver Alfred Gross, "Incidence matrices and interval graphs," *Pacific Journal of Mathematics*, vol. 15, pp. 835–855, 1965.

[17] Johannes Köbler, Sebastian Kuhnert, Bastian Laubner, and Oleg Verbitsky, "Interval graphs: Canonical representation in logspace," in *Automata, Languages and Programming*, Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, Eds., Berlin, Heidelberg, 2010, pp. 384–395, Springer Berlin Heidelberg.