# Succinct Representations of Intersection Graphs on a Circle

Hüseyin Acan *Sankardeep Chakraborty* Seungbum Jo
Kei Nakashima Kunihiko Sadakane Srinivasa Rao Satti

DATA COMPRESSION CONFERENCE (DCC)

February 28, 2021

# Motivation

# Motivation

- Large data sets
  - Biological sequences
  - Web graph
  - Geographical images etc.

# Motivation

- Large data sets
  - Biological sequences
  - Web graph
  - Geographical images etc.
- Goal:
  - *Representation* of the data;
  - *Retrieval* of the queries;
  - *Minimise* resources: Time and Space.

# Motivation

- Large data sets
  - Biological sequences
  - Web graph
  - Geographical images etc.
- Goal:
  - *Representation* of the data;
  - *Retrieval* of the queries;
  - *Minimise* resources: Time and Space.
- Compression – a potential solution (?)

# Motivation

- Large data sets
  - Biological sequences
  - Web graph
  - Geographical images etc.
- Goal:
  - *Representation* of the data;
  - *Retrieval* of the queries;
  - *Minimise* resources: Time and Space.
- Compression – a potential solution (?)
- Decompressing before the operations could:
  - be time inefficient, specifically when a tiny part is to be read;
  - not be feasible as not enough disk space.

# Motivation

- Large data sets
  - Biological sequences
  - Web graph
  - Geographical images etc.
- Goal:
  - *Representation* of the data;
  - *Retrieval* of the queries;
  - *Minimise* resources: Time and Space.
- Compression – a potential solution (?)
- Decompressing before the operations could:
  - be time inefficient, specifically when a tiny part is to be read;
  - not be feasible as not enough disk space.
- Can we operate directly on the compressed data?

# Compressed Data Structures

# Compressed Data Structures

- Set-up:

# Compressed Data Structures

- Set-up:
  - Given input data (i.e., some combinatorial object) $S$, can we represent $S$ *efficiently* so as to perform queries in (close to) constant time?

# Compressed Data Structures

- Set-up:
  - Given input data (i.e., some combinatorial object) $S$, can we represent $S$ *efficiently* so as to perform queries in (close to) constant time?
  - *Efficiently* could be:
    - Compact: $O(ITLB)$
    - Succinct: $ITLB + o(ITLB)$
    - Implicit/in-place: $ITLB + O(1)$

    where ITLB= Information-theoretic lower bound.
  - Model: Word-RAM with logarithmic word size and uniform cost; space is counted in bits.

# Examples of Succinct Data Structures

# Examples of Succinct Data Structures

- Sequences:
  - Data: $x \in \Sigma^n$ for some alphabet $\Sigma = \{0, 1, \cdots, \sigma - 1\}$
  - Naive encoding: $n \lceil \lg \sigma \rceil$ bits.
  - ITLB: $\lg \sigma^n = \lceil n \lg \sigma \rceil$ bits (DPT'10)

# Examples of Succinct Data Structures

- Sequences:
  - Data: $x \in \Sigma^n$ for some alphabet $\Sigma = \{0, 1, \cdots, \sigma - 1\}$
  - Naive encoding: $n \lceil \lg \sigma \rceil$ bits.
  - ITLB: $\lg \sigma^n = \lceil n \lg \sigma \rceil$ bits (DPT'10)
- Ordinal Trees:
  - Data: $x$ is an *ordinal tree* with $n$ nodes.
  - Ordinal: rooted tree, arbitrary # children, order matters.
  - Naive encoding: $\geq n$ pointers; $\Omega(n \lg n)$ bits.
  - ITLB: $\lg(\frac{1}{n+1}\binom{2n}{n}) = 2n - O(\lg n)$ bits (FM'08)

# Examples of Succinct Data Structures

- Sequences:
  - Data: $x \in \Sigma^n$ for some alphabet $\Sigma = \{0, 1, \cdots, \sigma - 1\}$
  - Naive encoding: $n \lceil \lg \sigma \rceil$ bits.
  - ITLB: $\lg \sigma^n = \lceil n \lg \sigma \rceil$ bits (DPT'10)
- Ordinal Trees:
  - Data: $x$ is an *ordinal tree* with $n$ nodes.
  - Ordinal: rooted tree, arbitrary # children, order matters.
  - Naive encoding: $\geq n$ pointers; $\Omega(n \lg n)$ bits.
  - ITLB: $\lg(\frac{1}{n+1}\binom{2n}{n}) = 2n - O(\lg n)$ bits (FM'08)
- (Arbitrary and various special classes of) Graphs, Permutations, Functions, Equivalence classes etc.

# Examples of Succinct Data Structures

- Sequences:
  - Data: $x \in \Sigma^n$ for some alphabet $\Sigma = \{0, 1, \cdots, \sigma - 1\}$
  - Naive encoding: $n \lceil \lg \sigma \rceil$ bits.
  - ITLB: $\lg \sigma^n = \lceil n \lg \sigma \rceil$ bits (DPT'10)
- Ordinal Trees:
  - Data: $x$ is an *ordinal tree* with $n$ nodes.
  - Ordinal: rooted tree, arbitrary # children, order matters.
  - Naive encoding: $\geq n$ pointers; $\Omega(n \lg n)$ bits.
  - ITLB: $\lg(\frac{1}{n+1}\binom{2n}{n}) = 2n - O(\lg n)$ bits (FM'08)
- (Arbitrary and various special classes of) Graphs, Permutations, Functions, Equivalence classes etc.
- "Application-Oriented Succinct Data Structures for Big Data" by Tetsuo Shibuya 2019.

# Our Work

# Our Work

- We consider the problem of designing succinct data structures supporting basic navigational queries such as degree, adjacency and neighborhood efficiently for various intersection graphs on a circle.

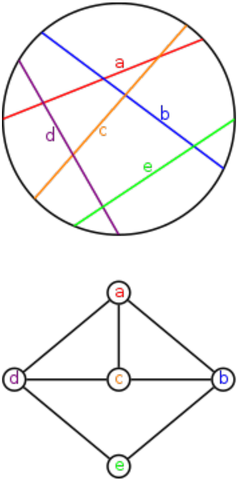
# Our Work

- We consider the problem of designing succinct data structures supporting basic navigational queries such as degree, adjacency and neighborhood efficiently for various intersection graphs on a circle.
- These include graph classes such as *circle graphs*, *k-polygon-circle graphs*, *circle-trapezoid graphs* etc.

# Definitions

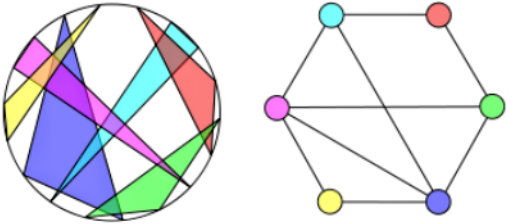
# Definitions

- A circle graph is defined as the intersection graph of chords in a circle.



# Definitions

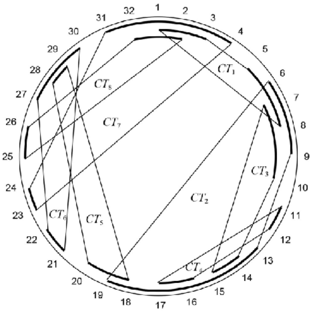
- *Polygon-circle* graphs are the intersection graphs of convex polygons inscribed into a circle, and the special case, when all the convex polygons have exactly $k$ corners, we call the intersection graph $k$-polygon-circle.



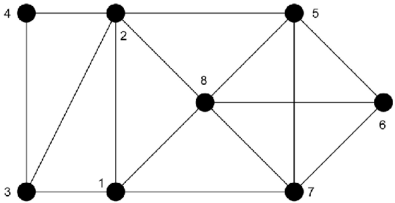
# Definitions

- Circle-trapezoid graphs are the intersection graphs of circle trapezoids on a common circle, where a circle trapezoid is defined as the convex hull of two disjoint arcs on the circle.

# Definitions




(a)




(b)

# Main Result: Lower Bounds

Table: Lower bounds of families of intersection graphs.

| Graph class | Space lower bound (in bits) |
|---|---|
| circle | $n \log n - \mathrm{O}(n)$ |
| $k$-polygon-circle | $(k-1)n \log n - \mathrm{O}(kn \log \log n)$ |
| circle-trapezoid | $3n \log n - 4 \log \log n - \mathrm{O}(n)$ |

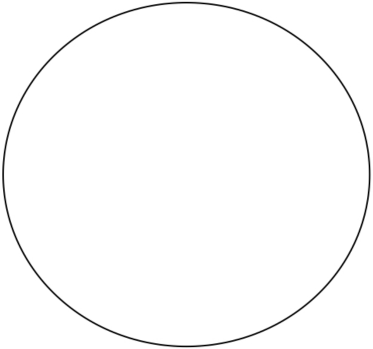
# Main Result: Upper Bounds

**Theorem**

*There exist succinct encodings for previously mentioned graph classes such that adjacent$(u, v)$ query can be reported in $\mathrm{O}(k \log\log n)$ time, and neighborhood$(v)$ and degree$(v)$ queries can be answered in $\mathrm{O}(k|\textit{degree}(v)| \cdot \log\log n)$ time.*

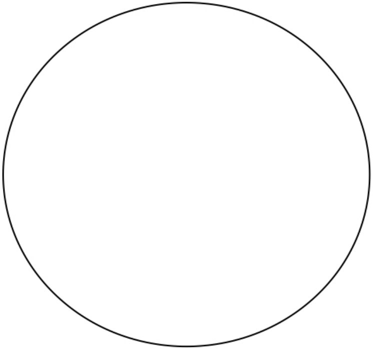
# Proof Sketch for Circle Graph Lower Bound

# Proof Sketch for Circle Graph Lower Bound



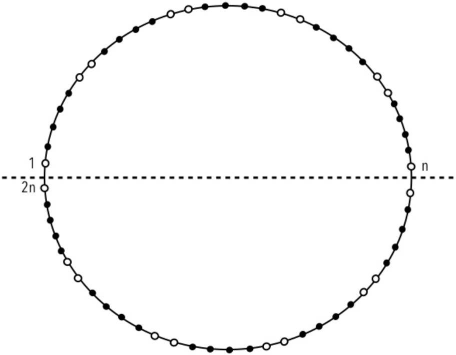
# Proof Sketch for Circle Graph Lower Bound

Place 2n points, label these clockwise such that first n points lie on the upper semi circle and the rest lie on lower semicircle



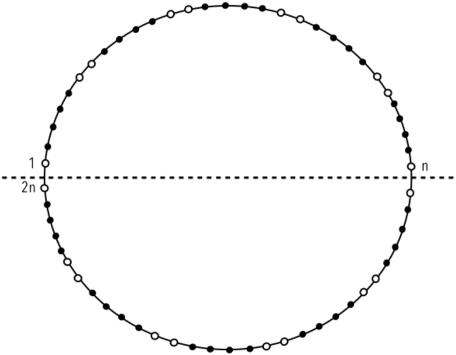
# Proof Sketch for Circle Graph Lower Bound

Place 2n points, label these clockwise such that first n points lie on the upper semi circle and the rest lie on lower semicircle



# Proof Sketch for Circle Graph Lower Bound

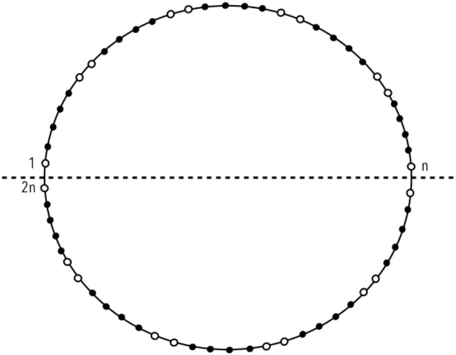
Place 2n points, label these clockwise such that first n points lie on the upper semi circle and the rest lie on lower semicircle




These 2n points will be the endpoints of n disjoint chords

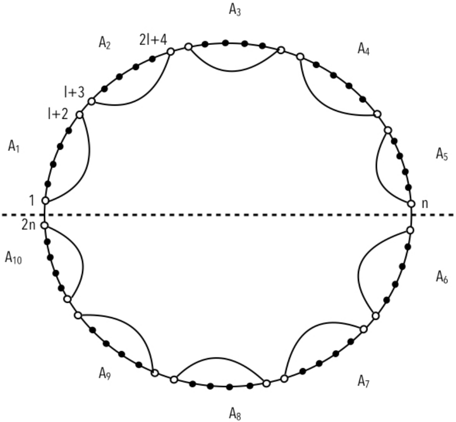
# Proof Sketch for Circle Graph Lower Bound

On each semicircle, take k chords, each of which determines an arc with l points on it, excluding the endpoints of the chord



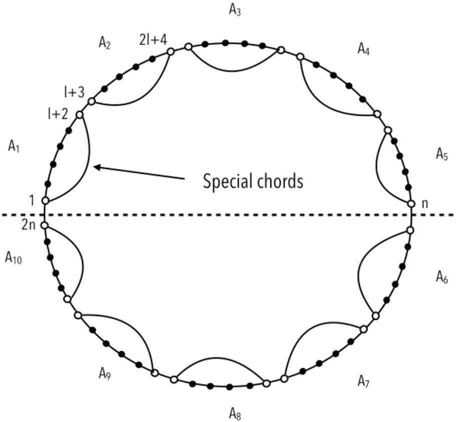
# Proof Sketch for Circle Graph Lower Bound

On each semicircle, take k chords, each of which determines an arc with l points on it, excluding the endpoints of the chord



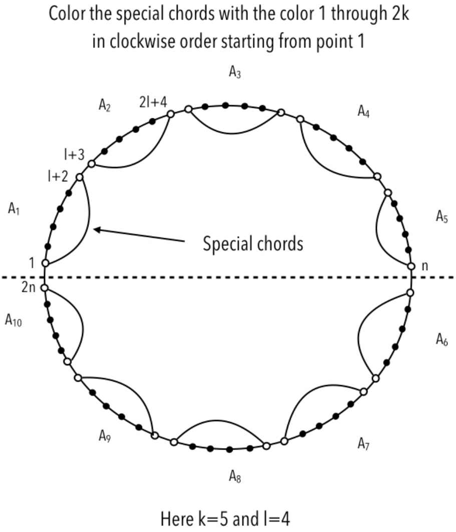

Here k=5 and l=4

# Proof Sketch for Circle Graph Lower Bound

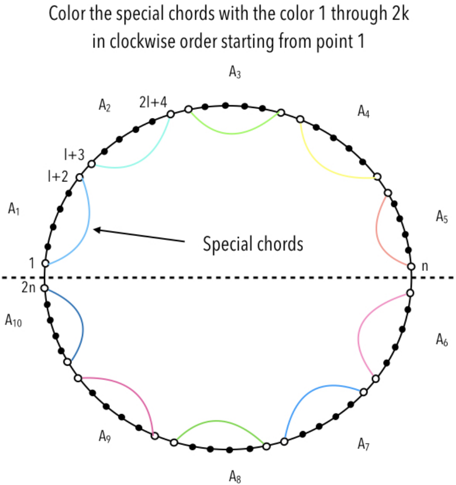



Here k=5 and l=4

# Proof Sketch for Circle Graph Lower Bound



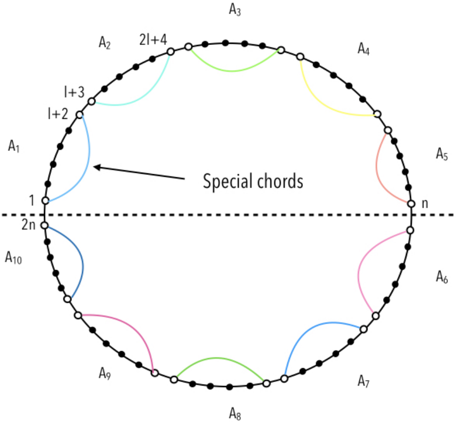
# Proof Sketch for Circle Graph Lower Bound




Here k=5 and l=4

# Proof Sketch for Circle Graph Lower Bound

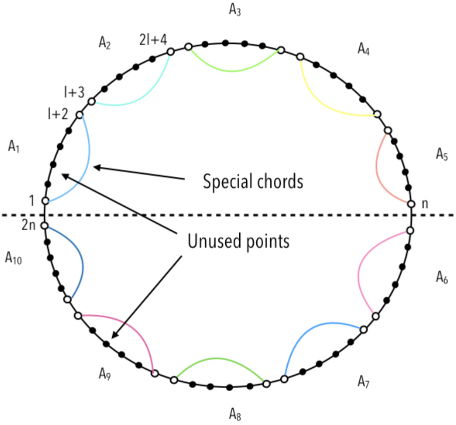
So far 4k out of 2n points have been used. Remaining (2n-4k) points lie on 2k arcs determined by the 2k special chords. We want, 2kl+4k=2n, thus, l=(n-2k)/k




Here k=5 and l=4

# Proof Sketch for Circle Graph Lower Bound

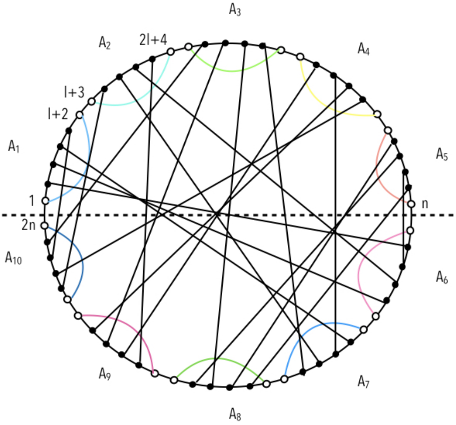
We want to match unused kl points on the upper semicircle with kl unused points on the lower semicircle. #matchings= (kl)!




Here k=5 and l=4

# Proof Sketch for Circle Graph Lower Bound

We want to match unused kl points on the upper semicircle with kl unused points on the lower semicircle. #matchings= (kl)!




Here k=5 and l=4

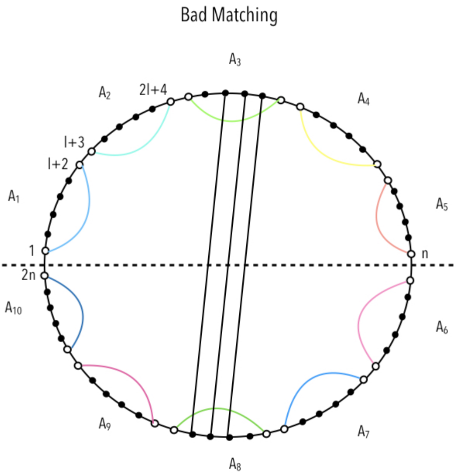
# Proof Sketch for Circle Graph Lower Bound

- For each pair in the matching, if we draw the chord connecting the points in the pair, we get $n$ chords which gives a colored circle graph. (Each chord corresponds to a vertex.)
- The $2k$ vertices corresponding to the special $2k$ chords are colored 1 through $2k$ (in the same canonical order), and the other $n - 2k$ vertices are uncolored.

# Proof Sketch for Circle Graph Lower Bound

- For each pair in the matching, if we draw the chord connecting the points in the pair, we get $n$ chords which gives a colored circle graph. (Each chord corresponds to a vertex.)
- The $2k$ vertices corresponding to the special $2k$ chords are colored 1 through $2k$ (in the same canonical order), and the other $n - 2k$ vertices are uncolored.
- Let $M$ be a matching from $\cup_{i=1}^{k} A_i$ to $\cup_{j=k+1}^{2k} A_j$. We call $M$ a *bad matching* if it contains a triple of pairs $((x_1, y_1), (x_2, y_2), (x_3, y_3))$ such that $x_1, x_2, x_3$ lie on $A_i$ for some $i \leq k$ and $y_1, y_2, y_3$ lie on $A_{k+j}$ for some $j \leq k$. Otherwise we call it a *good matching* (denoted by $\mathcal{M}$).

# Proof Sketch for Circle Graph Lower Bound



# Almost all matchings are good

**Theorem**

*Let $k = n^{3/4+\epsilon}$ for some fixed small $\epsilon > 0$. For a random matching $M$, the expected number of triples of pairs $((x_1, y_1), (x_2, y_2), (x_3, y_3))$ in bad matching tends to 0 as $n \to \infty$. i.e, $\frac{|\mathcal{M}|}{(k\ell)!} = 1 - o(1)$ as $n \to \infty$. Consequently, almost all matchings are good.*

# Almost all matchings are good

**Theorem**

*Let $k = n^{3/4+\epsilon}$ for some fixed small $\epsilon > 0$. For a random matching $M$, the expected number of triples of pairs $((x_1, y_1), (x_2, y_2), (x_3, y_3))$ in bad matching tends to 0 as $n \to \infty$. i.e, $\frac{|\mathcal{M}|}{(k\ell)!} = 1 - o(1)$ as $n \to \infty$. Consequently, almost all matchings are good.*

- A good matching can be recovered from its (colored) circle graph. In other words, there is a one to one matching between the set of good matchings and their corresponding graphs.

# Proof Sketch for Circle Graph Lower Bound

- Let $C_n$ be the number of unlabeled circle graphs with $n$ vertices. Then,

$$
\begin{aligned}
C_n \binom{n}{2k} (2k)! &\geq \text{number of circle graphs with } 2k \text{ colored vertices} \\
&\geq \text{number of circle graphs obtained from the construction} \\
&\geq |\mathcal{M}|
\end{aligned}
$$

# Proof Sketch for Circle Graph Lower Bound

- Let $C_n$ be the number of unlabeled circle graphs with $n$ vertices. Then,

$$
\begin{aligned}
C_n \binom{n}{2k} (2k)! &\geq \text{number of circle graphs with } 2k \text{ colored vertices} \\
&\geq \text{number of circle graphs obtained from the construction} \\
&\geq |\mathcal{M}|
\end{aligned}
$$

- As $|\mathcal{M}| = (1 - o(1))(k\ell)! = (1 - o(1))(n - 2k)!$, we get after simplifying, $\log C_n = n \log n - O(n)$. Hence, we need at least $n \log n - O(n)$ bits to represent a circle graph with $n$ nodes.

# Future Directions

- Faster query times?
- Other graphs?

Thank You.