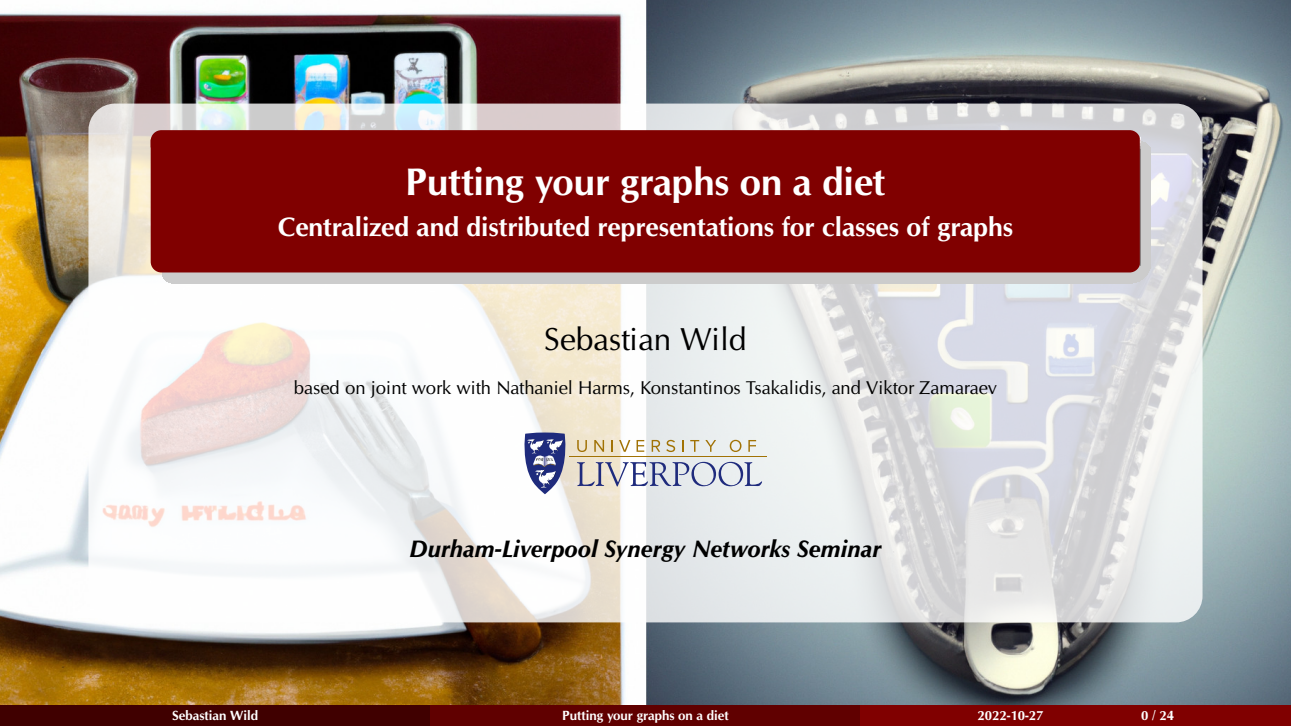


Putting your graphs on a diet

Centralized and distributed representations for classes of graphs

Sebastian Wild

based on joint work with Nathaniel Harms, Konstantinos Tsakalidis, and Viktor Zamaraev

UNIVERSITY OF LIVERPOOL

Durham-Liverpool Synergy Networks Seminar

DALL·E:
putting a social network on a diet in photorealistic style



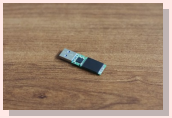
https://labs.openai.com/



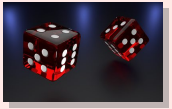
DALL·E:
a photorealistic image of a social network inside a large zip icon

https://labs.openai.com/

Outline



1 Introduction



2 Randomization in Labeling Schemes



3 Succinct Permutation Graphs



4 Conclusion

1 Introduction



Motivation

- ubiquitous data model: entities and relations $\rightsquigarrow$ graphs/networks
- large data $\rightsquigarrow$ desire to
  1. compress / use less space
  2. use distributed storage of data

Is efficient computation & distributed storage intrinsically more costly?

- largely missing:
  - an information theory of (graph-)structured data
    How much space is needed to store a graph?
  - When and how can we achieve such space with
    - efficient queries?
    - distributed storage?
- today: some feeble steps in well-defined parts

Besta, Hoefler: *Survey and Taxonomy of Lossless Graph Compression and Space-Efficient Graph Representations*, arXiv **2018**

Spinrad: *Efficient graph representations*, Fields monographs **2003**

Motivation

- ubiquitous data model: entities and relations $\rightsquigarrow$ graphs/networks
- large data $\rightsquigarrow$ desire to
  1. compress / use less space
  2. use distributed storage of data

Is efficient computation & distributed storage intrinsically more costly?

- largely missing:
  - an information theory of (graph-)structured data
    How much space is needed to store a graph?
  - When and how can we achieve such space with
    - efficient queries?
    - distributed storage?
- today: some feeble steps in well-defined parts

Besta, Hoefler: *Survey and Taxonomy of Lossless Graph Compression and Space-Efficient Graph Representations*, arXiv **2018**

Spinrad: *Efficient graph representations*, Fields monographs **2003**

Motivation

- ubiquitous data model: entities and relations $\rightsquigarrow$ graphs/networks
- large data $\rightsquigarrow$ desire to
  1. compress / use less space
  2. use distributed storage of data

Is efficient computation & distributed storage intrinsically more costly?

- largely missing:
  - an information theory of (graph-)structured data
    How much space is needed to store a graph?
  - When and how can we achieve such space with
    - efficient queries?
    - distributed storage?
- today: some feeble steps in well-defined parts



Besta, Hoefler: *Survey and Taxonomy of Lossless Graph Compression and Space-Efficient Graph Representations,* arXiv **2018**



Spinrad: *Efficient graph representations,* Fields monographs **2003**

Today's scope

All graphs in this talk are

- undirected
- unlabeled (just topology / adjacency relation)
- static (they don't change)
- uniformly taken from a ground set ($\neq$ no typical / atypical graphs)

The ground sets are **hereditary graph families** $\mathcal{F}$: (Some questions become degenerate otherwise)

1. $\mathcal{F}$ closed under isomorphism
2. $\mathcal{F}$ closed under taking **induced subgraphs**

$\rightsquigarrow$ $\mathcal{F}_n$: graphs $G \in \mathcal{F}$ with vertex set $V(G) = [n]$

most natural families:

- planar graphs, excluded-minor graphs
- bounded-degeneracy graphs
- geometric-intersection graphs
- bounded-twinwidth graphs
- ...

We will ask questions like: What (hereditary) graph families can ...?

Today's scope

All graphs in this talk are

- undirected
- unlabeled (just topology / adjacency relation)
- static (they don't change)
- uniformly taken from a ground set ($\neq$ no typical / atypical graphs)

The ground sets are **hereditary graph families** $\mathcal{F}$: (Some questions become degenerate otherwise)

1. $\mathcal{F}$ closed under isomorphism
2. $\mathcal{F}$ closed under taking **induced subgraphs**

$\rightsquigarrow$ $\mathcal{F}_n$: graphs $G \in \mathcal{F}$ with vertex set $V(G) = [n]$

most natural families:

- planar graphs, excluded-minor graphs
- bounded-degeneracy graphs
- geometric-intersection graphs
- bounded-twinwidth graphs
- ...

We will ask questions like: What (hereditary) graph families can ...?

Space-efficient graph representations

Given a (hereditary) graph family $\mathcal{F}$, we define

1. A **succinct encoding** of $\mathcal{F}$:

   $encode : \mathcal{F} \to \{0,1\}^*$, $decode : \{0,1\}^* \to \mathcal{F}$

   - lossless: $decode(encode(G)) = G$ for $G \in \mathcal{F}$.
   - **succinct**: $G \in \mathcal{F}^n \rightsquigarrow |encode(G)| = \log_2(|\mathcal{F}^n|) \cdot (1 + o(1))$
   - **efficient**: *encode*, *decode* efficiently computable (say polytime)

2. A **succinct data structure** for $\mathcal{F}$ (for adjacency):

   succinct encoding plus adjacency-list queries

   - $adjacent(v,u)$: 1 if $vu \in E(G)$ else 0
   - $nextNeigbor(v,u)$: successor of u in v's adj list
   - computable efficiently on word-RAM
     say $o(\log(|\mathcal{F}^n|))$ time; often $O(1)$
   - (potentially more queries)

3. A **succinct labeling scheme** for $\mathcal{F}$ (for adjacency):

   $\ell : V(G) \to \{0,1\}^*$,
   $labelAdj : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$

   - $labelAdj(\ell(v), \ell(u)) = adjacent(v,u)$ for $v, u \in V(G)$.
   - **succinct**: $G \in \mathcal{F}^n \rightsquigarrow |\ell(v)| \leqslant \frac{1}{n} \log_2(|\mathcal{F}^n|)(1 + o(1))$
   - weaker version: **compact**: $|\ell(v)| = O(\frac{1}{n} \log_2(|\mathcal{F}^n|))$
   - (labels and decoder can differ for each $G \in \mathcal{F}$, but heredity puts limits on that)

Space-efficient graph representations

Given a (hereditary) graph family $\mathcal{F}$, we define

1. A **succinct encoding** of $\mathcal{F}$:

   $encode : \mathcal{F} \to \{0,1\}^*$, $decode : \{0,1\}^* \to \mathcal{F}$

   - lossless: $decode(encode(G)) = G$ for $G \in \mathcal{F}$.
   - **succinct**: $G \in \mathcal{F}^n \rightsquigarrow |encode(G)| = \log_2(|\mathcal{F}^n|) \cdot (1 + o(1))$
   - **efficient**: *encode*, *decode* efficiently computable (say polytime)

2. A **succinct data structure** for $\mathcal{F}$ (for adjacency):

   succinct encoding plus adjacency-list queries

   - $adjacent(v, u)$: 1 if $vu \in E(G)$ else 0
   - $nextNeigbor(v, u)$: successor of u in v's adj list
   - computable efficiently on word-RAM
     say $o(\log(|\mathcal{F}^n|))$ time; often $O(1)$
   - (potentially more queries)

3. A **succinct labeling scheme** for $\mathcal{F}$ (for adjacency):

   $\ell : V(G) \to \{0,1\}^*$,
   $labelAdj : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$

   - $labelAdj(\ell(v), \ell(u)) = adjacent(v, u)$ for $v, u \in V(G)$.
   - **succinct**: $G \in \mathcal{F}^n \rightsquigarrow |\ell(v)| \leqslant \frac{1}{n} \log_2(|\mathcal{F}^n|)(1 + o(1))$
   - weaker version: **compact**: $|\ell(v)| = O(\frac{1}{n} \log_2(|\mathcal{F}^n|))$
   - (labels and decoder can differ for each $G \in \mathcal{F}$, but heredity puts limits on that)

Space-efficient graph representations

Given a (hereditary) graph family $\mathcal{F}$, we define

1. A **succinct encoding** of $\mathcal{F}$:

   $encode : \mathcal{F} \to \{0,1\}^*$, $decode : \{0,1\}^* \to \mathcal{F}$

   - lossless: $decode(encode(G)) = G$ for $G \in \mathcal{F}$.
   - **succinct**: $G \in \mathcal{F}^n \rightsquigarrow |encode(G)| = \log_2(|\mathcal{F}^n|) \cdot (1 + o(1))$
   - **efficient**: *encode*, *decode* efficiently computable (say polytime)

2. A **succinct data structure** for $\mathcal{F}$ (for adjacency):

   succinct encoding plus adjacency-list queries

   - $adjacent(v, u)$: 1 if $vu \in E(G)$ else 0
   - $nextNeigbor(v, u)$: successor of u in v's adj list
   - computable efficiently on word-RAM
     say $o(\log(|\mathcal{F}^n|))$ time; often $O(1)$
   - (potentially more queries)

3. A **succinct labeling scheme** for $\mathcal{F}$ (for adjacency):

   $\ell : V(G) \to \{0,1\}^*$,
   $labelAdj : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$

   - $labelAdj(\ell(v), \ell(u)) = adjacent(v, u)$ for $v, u \in V(G)$.
   - **succinct**: $G \in \mathcal{F}^n \rightsquigarrow |\ell(v)| \leqslant \frac{1}{n} \log_2(|\mathcal{F}^n|)(1 + o(1))$
   - weaker version: **compact**: $|\ell(v)| = O(\frac{1}{n} \log_2(|\mathcal{F}^n|))$
   - (labels and decoder can differ for each $G \in \mathcal{F}$, but heredity puts limits on that)

Questions

1. Do all graph families have a succinct encoding?
2. Do all graph families have a succinct data structure (with adjacency-list queries)?
3. Do all graph families have a succinct (adjacency) labeling scheme?

A caveat

How many graphs of size n are there in a family $\mathcal{F}$?

$\mathcal{F}^n$: set of **labeled graphs** of size n

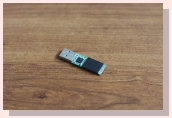
$\mathcal{F}_n$: set equivalence classes (under graph isomorphisms) of $\mathcal{F}^n$;
set of **unlabeled graphs** of size n

- $\mathcal{F}$ = all complete graphs
  $|\mathcal{F}^n| = |\mathcal{F}_n| = 1$ (the single complete graph over $[n]$)
- $\mathcal{F}$ = at most one edge in total
  $|\mathcal{F}^n| = 1 + \binom{n}{2}$ (1 empty graph plus $\binom{n}{2}$ ways to pick a pair for the single edge)
  $|\mathcal{F}_n| = 2$

Expert note: Concatenating all labels of a labeling scheme allows to reconstruct the *equivalence class* (under graph isomorphisms), but not the *labeled graph* – unless the original labels are made part of $\ell(v)$.

$\rightsquigarrow$ need to pay attention

Outline



1 Introduction



2 Randomization in Labeling Schemes



3 Succinct Permutation Graphs



4 Conclusion



2 Randomization in Labeling Schemes

Harms, Wild, Zamaraev: *Randomized Communication and Implicit Graph Representations,* STOC **2022**

Implicit Graph Conjecture

3. **Do all graph families have a compact (adjacency) labeling scheme?**

- A graph family $\mathcal{F}$ is called **factorial**
  if $\log_2 |\mathcal{F}^n| = \Theta(n \log n)$ (so $|\mathcal{F}^n|$ grows *very roughly* like $n!$)

$\rightsquigarrow$ any adjacency labeling scheme for $\mathcal{F}$ must use labels of $|\ell(v)| = \Omega(\log n)$ bits per vertex v.

Implicit Graph Conjecture
Any hereditary, factorial graph family admits a $O(\log n)$-bit adjacency labeling scheme.

Kannan, Naro, Rudich: *Implicit representation of graphs,* SIDMA **1992**

Spinrad: *Efficient graph representations,* Fields monographs **2003**

$\rightsquigarrow$ long-standing open question . . . let's narrow it down further

Implicit Graph Conjecture

3 Do all graph families have a compact (adjacency) labeling scheme?

- A graph family $\mathcal{F}$ is called **factorial**
  if $\log_2 |\mathcal{F}^n| = \Theta(n \log n)$ (so $|\mathcal{F}^n|$ grows *very roughly* like $n!$)

$\rightsquigarrow$ any adjacency labeling scheme for $\mathcal{F}$ must use labels of $|\ell(v)| = \Omega(\log n)$ bits per vertex v.

Implicit Graph Conjecture
Any hereditary, factorial graph family admits a $O(\log n)$-bit adjacency labeling scheme.

Kannan, Naro, Rudich: *Implicit representation of graphs,* SIDMA **1992**

Spinrad: *Efficient graph representations,* Fields monographs **2003**

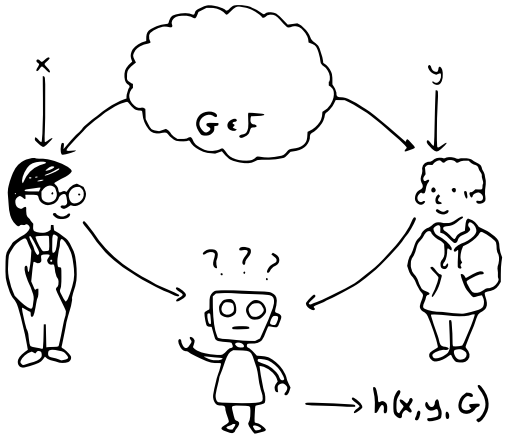
$\rightsquigarrow$ long-standing open question . . . let's narrow it down further

Labeling schemes as communication problem

Graph labeling for $\mathcal{F}$ has similarity to a **communication complexity** setting:

1. Alice & Bob each get a vertex (x resp. y)
2. They each send one message to a referee
3. The referee has to output whether x and y are adjacent in G.

   The referee only knows G and the messages from Alice and Bob



Picture: Nathaniel Harms

Labeling schemes as communication problem

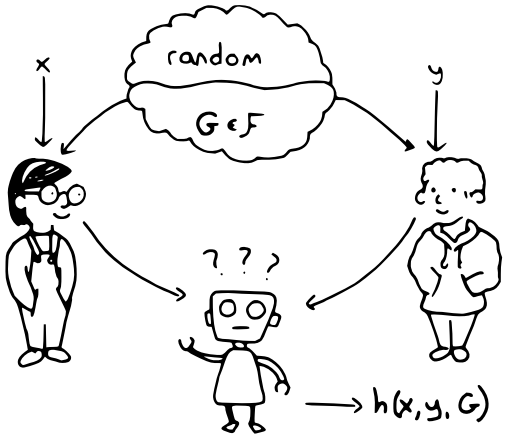
Graph labeling for $\mathcal{F}$ has similarity to a **communication complexity** setting:

1. Alice & Bob each get a vertex (x resp. y)
2. They each send one message to a referee
3. The referee has to output whether x and y are adjacent in G.

   The referee only knows G and the messages from Alice and Bob

Standard addition in comm. complexity:

- Alice & Bob have access to a **shared random bitstring**. (Referee does not see it).
- Output correct with probability $\frac{2}{3}$



Picture: Nathaniel Harms

Labeling schemes as communication problem

Graph labeling for $\mathcal{F}$ has similarity to a **communication complexity** setting:

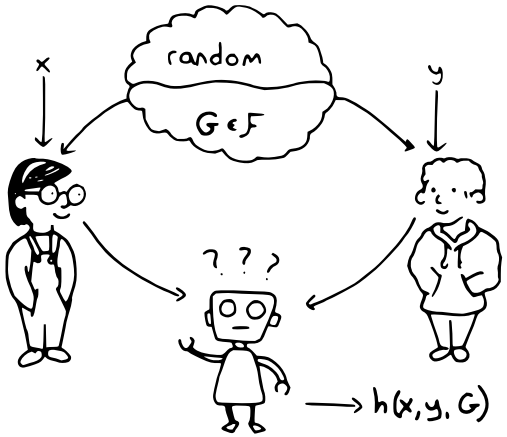
1. Alice & Bob each get a vertex (x resp. y)
2. They each send one message to a referee
3. The referee has to output whether x and y are adjacent in G.

   The referee only knows G and the messages from Alice and Bob

Standard addition in comm. complexity:

- Alice & Bob have access to a **shared random bitstring**. (Referee does not see it).
- Output correct with probability $\frac{2}{3}$

← What would this mean for labeling schemes?



Picture: Nathaniel Harms

Adjacency sketches

Randomized version of labeling scheme a.k.a. **adjacency sketch**

- **randomized** encoder computes labels $sk : V(G) \to \{0,1\}^*$
- deterministic decoder $labelAdj : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ (as before)
- $\forall v, u \in V(G): \quad \mathbb{P}\Big[labelAdj(sk(v), sk(u)) \;=\; adjacent(v,u)\Big] \;\geqslant\; \frac{2}{3}$



Harms: *Universal Communication, Universal Graphs, and Graph Labeling*, ITCS **2020**

Observations:

- any standard adjacency labeling scheme is also a valid sketch sketch
- with $O(\log n)$ independent copies and using majority vote
  we can amplify success prob to $\ll \frac{1}{n^2}$ $\rightsquigarrow$ deterministic!

$\rightsquigarrow$ any c-bit adjacency sketch implies a $O(c \log n)$ labeling scheme!

Adjacency sketches

Randomized version of labeling scheme a.k.a. **adjacency sketch**

- **randomized** encoder computes labels $sk : V(G) \to \{0,1\}^*$
- deterministic decoder $labelAdj : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ (as before)
- $\forall v, u \in V(G): \quad \mathbb{P}\Big[labelAdj(sk(v), sk(u)) \;=\; adjacent(v,u)\Big] \;\geqslant\; \frac{2}{3}$



Harms: *Universal Communication, Universal Graphs, and Graph Labeling*, ITCS **2020**

Observations:

- any standard adjacency labeling scheme is also a valid sketch sketch
- with $O(\log n)$ independent copies and using majority vote we can amplify success prob to $\ll \frac{1}{n^2}$ $\rightsquigarrow$ deterministic!

$\rightsquigarrow$ any c-bit adjacency sketch implies a $O(c \log n)$ labeling scheme!

Greater-Than and Chain graphs

So: $O(1)$-bit sketch $\implies$ $O(\log n)$ labeling; *is the converse also true?*

No!

- The **Greater-Than problem**:
  - Alice & Bob get numbers $x, y \in [n]$
  - Referee must decide whether $x \geqslant y$ from messages
- known to require $\Theta(\log n)$ bits of communication (one-way public coin)
  - $\rightsquigarrow$ randomization doesn't help; still need to send x and y in full.

Miltersen, Nisan, Safra, Wigderson: *On data structures and asymmetric communication complexity*, JCSS **1998**

- $H_n^{\circ\circ}$ does **not** have $o(\log n)$ **adj sketch**
- but it has $O(\log n)$ **labeling scheme**

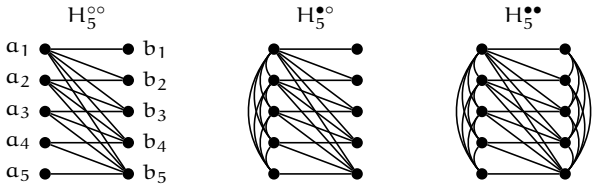
Greater-Than and Chain graphs

So: $O(1)$-bit sketch $\implies$ $O(\log n)$ labeling; *is the converse also true?*

No!

- The **Greater-Than problem**:
  - Alice & Bob get numbers $x, y \in [n]$
  - Referee must decide whether $x \geqslant y$ from messages
- known to require $\Theta(\log n)$ bits of communication (one-way public coin)
  - $\rightsquigarrow$ randomization doesn't help; still need to send x and y in full.

Miltersen, Nisan, Safra, Wigderson: *On data structures and asymmetric communication complexity*, JCSS **1998**



- $H_n^{\circ\circ}$ does **not** have $o(\log n)$ **adj sketch**
- but it has $O(\log n)$ **labeling scheme**

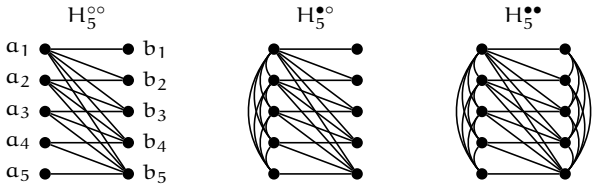
Greater-Than and Chain graphs

So: $O(1)$-bit sketch $\implies$ $O(\log n)$ labeling; *is the converse also true?*

No!

- The **Greater-Than problem**:
  - Alice & Bob get numbers $x, y \in [n]$
  - Referee must decide whether $x \geqslant y$ from messages
- known to require $\Theta(\log n)$ bits of communication (one-way public coin)
  - $\rightsquigarrow$ randomization doesn't help; still need to send x and y in full.

Miltersen, Nisan, Safra, Wigderson: *On data structures and asymmetric communication complexity*, JCSS **1998**



- $H_n^{\circ\circ}$ does **not** have $o(\log n)$ **adj sketch**
- but it has $O(\log n)$ **labeling scheme**

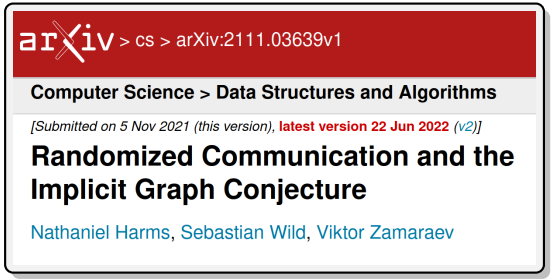
Probabilistic Implicit Graph Conjecture

⇝ Our conjecture (Nov 2021)

Probabilistic Implicit Graph Conjecture

Any hereditary, factorial, and **stable** graph family admits a $O(1)$-bit adjacency sketch.

- Here stable means:
  $\mathcal{F}$ does **not** contain all $H_n^{\circ\circ}$, $H_n^{\bullet\circ}$, $H_n^{\bullet\bullet}$
- This is clearly necessary
  ($O(1)$ sketch $\implies$ stable);
  we conjectured it is also sufficient



Probabilistic Implicit Graph Conjecture

2 weeks later:

arXiv > cs > arXiv:2111.10436

Computer Science > Computational Complexity

[Submitted on 19 Nov 2021 (v1), last revised 8 Dec 2021 (this version, v2)]

A counter-example to the probabilistic universal graph conjecture via randomized communication complexity

Lianna Hambardzumyan, Hamed Hatami, Pooya Hatami

- This is clearly necessary
  ($O(1)$ sketch $\implies$ stable);
  we conjectured it is also sufficient

arXiv > cs > arXiv:2111.03639v1

Computer Science > Data Structures and Algorithms

[Submitted on 5 Nov 2021 (this version), latest version 22 Jun 2022 (v2)]

Randomized Communication and the Implicit Graph Conjecture

Nathaniel Harms, Sebastian Wild, Viktor Zamaraev

Probabilistic Implicit Graph Conjecture

2 weeks later:

arXiv > cs > arXiv:2111.10436

Computer Science > Computational Complexity

[Submitted on 1

A count
univers
commu

Lianna Ham

arXiv > cs > arXiv:2111.03639v1

Computer Science > Data Structures and Algorithms

[Submitted on 5 … 2022 (v2)]

…he

Another week later:

arXiv > math > arXiv:2111.13198

Mathematics > Combinatorics

[Submitted on 25 Nov 2021 (v1), last revised 13 Dec 2021 (this version, v2)]

The Implicit Graph Conjecture is False

Hamed Hatami, Pooya Hatami

- Th
(O
we

Aftermath

With both conjectures down, the search for characterizations is wide open!

1. Which factorial graph families admit $O(\log n)$**-bit adjacency labeling schemes**?
2. Maybe more humble:
   Which ***stable*** factorial graph families admit $O(\log n)$**-bit adjacency labeling schemes**?
3. Which stable factorial graph classes admit $O(1)$**-bit adjacency sketch**?

   This is equivalent to characterizing which functions can be computed with constant communication, a natural and open question in communication complexity.

   We give many positive examples
   - stable families of bounded twin-width
   - stable monogenic bipartite graphs
   - all stable subfamilies of interval graphs
   - all stable subfamilies of permutation graphs
   - Cartesian products of stable families (e. g., hypercubes)
   - minimal factorial families



Harms, Wild, Zamaraev: *Randomized Communication and Implicit Graph Representations*, STOC **2022**

(Secondary) Goal of our work: new labeling schemes

Typical **techniques for adjacency labeling schemes**

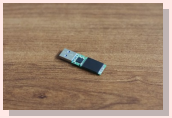
1. bounded arboricity / degeneracy: list neighbors explicitly
2. exploit realization of geometric intersection graphs, e.g., interval graphs
3. Boolean combinations of subschemes:
   - cover graph ($E(G)$) with (finitely many) graphs: Boolean combinations of sublabeling schemes
   - recursive decomposition of graphs, e.g., for graphs of bounded cliquewidth



(de)randomization as design principle for labeling schemes

- derandomized Cartesian product sketch
- can solve cases where other techniques provably don't work

Outline



1 Introduction



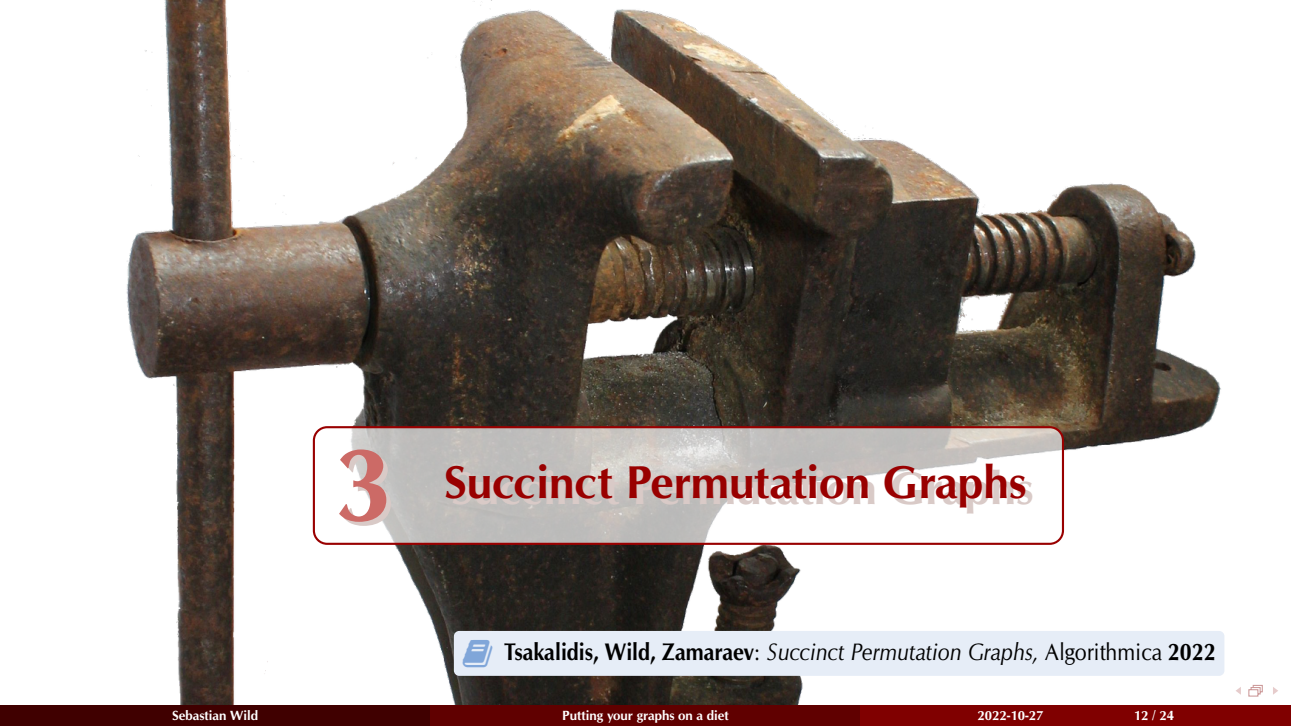
2 Randomization in Labeling Schemes



3 Succinct Permutation Graphs



4 Conclusion



3 Succinct Permutation Graphs

Tsakalidis, Wild, Zamaraev: *Succinct Permutation Graphs,* Algorithmica **2022**

Succinct Graph Data Structures

2 Do all graph families have a succinct data structure (with adjacency-list queries)?

- Not known to my knowledge …
- No, for graphs with m edges and $n^\delta < m < n^{2-\delta}$, ($\delta > 0$ constant) when we want *adjacent* and *nextNeighbor* in $O(1)$ time

  (But this is not a hereditary property.)
  (This is only for $O(1)$ time queries.)

Farzan, Munro: *Succinct encoding of arbitrary graphs,* TCS **2013**

Succinct Graph Conjecture
Every hereditary graph family admits a succinct data structure with *adjacent* and *nextNeighbor* queries in $O(\log^c(n))$ time.

Let's build some evidence for this.

Succinct Graph Data Structures

2 Do all graph families have a succinct data structure (with adjacency-list queries)?

- Not known to my knowledge …
- No, for graphs with m edges and $n^\delta < m < n^{2-\delta}$, ($\delta > 0$ constant) when we want *adjacent* and *nextNeighbor* in $O(1)$ time

  (But this is not a hereditary property.)
  (This is only for $O(1)$ time queries.)

Farzan, Munro: *Succinct encoding of arbitrary graphs,* TCS **2013**



Succinct Graph Conjecture
Every hereditary graph family admits a succinct data structure with *adjacent* and *nextNeighbor* queries in $O(\log^c(n))$ time.

Let's build some evidence for this.

Known succinct graph data structures

For several graph families, succinct data structure are known

- interval graphs
- circle graphs
- path graphs
- series-parallel graphs
- bounded clique-width (compact ds)

*Usually *adjacent* & *neighborhood* queries $\rightsquigarrow$ *nextNeighbor* only supported using extra space ✓

Goals:

1. Add **permutation graphs** to this list!
2. Can we do *nextNeighbor* without extra space?

Known succinct graph data structures

For several graph families, succinct data structure are known

- interval graphs
- circle graphs
- path graphs
- series-parallel graphs
- bounded clique-width (compact ds)

* Usually *adjacent* & *neighborhood* queries $\rightsquigarrow$ *nextNeighbor* only supported using extra space ✓

Goals:

1. Add **permutation graphs** to this list!
2. Can we do *nextNeighbor* without extra space?

Next-neighbor query

- in algorithms: assume adjacency lists representation (mostly)
  - key feature: iterate over neighbors
  - if we are space-conscious, don't want a **copy** of list of all neighbors
  - instead, keep pointer into adjacency list

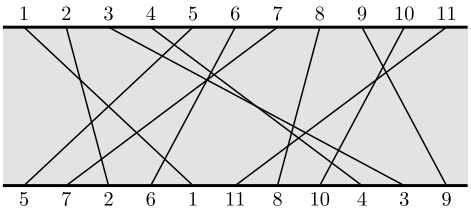
$\rightsquigarrow$ *nextNeighbor* query:

- *nextNeigbor*$(v, \bot)$: first neighbor in v's adj list
- *nextNeigbor*(v, u): successor of u in v's adj list
- as in adj lists, order of iteration is arbitrary / not specified

Permutation Graphs

- $G = (V, E)$ is a **permutation graph (PG)** iff
  - V = chords between 2 parallel lines
  - adjacent iff chords intersect

  (A type of geometric intersection graph)



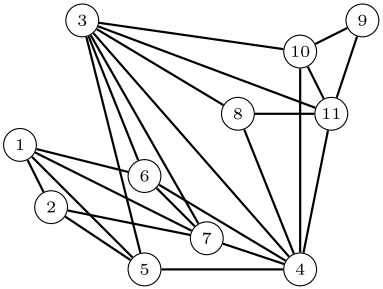
- PGs can be dense $\rightsquigarrow$ can't list edges

- labeled PGs on n vertices
  $|\mathcal{P}^n| \leqslant n! \cdot n!$
  - storing rank of start and endpoint of chord
    $\rightsquigarrow$ encoding for labeled graph
    $\rightsquigarrow$ $2n \log_2 n$ bits

- unlabeled PGs
  can prove $\log_2(|\mathcal{P}_n|) = n \log_2(n)(1 + o(1))$

$\rightsquigarrow$ aim for $n \log_2 n$ bit data structure
which is allowed to *choose* vertex labeling

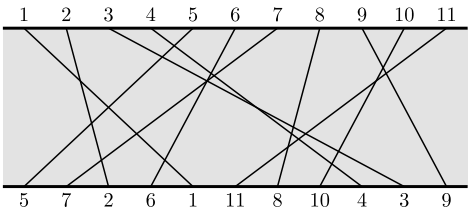
$\rightsquigarrow$ store permutation of endpoints!



Permutation Graphs

- $G = (V, E)$ is a **permutation graph (PG)** iff
  - V = chords between 2 parallel lines
  - adjacent iff chords intersect

  (A type of geometric intersection graph)



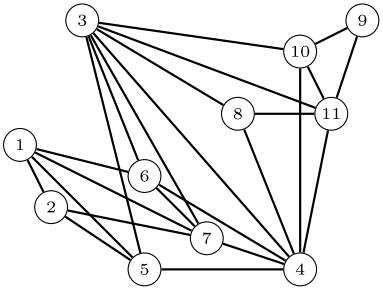
- PGs can be dense $\rightsquigarrow$ can't list edges

- $|\mathcal{P}_n| \leqslant n! \cdot n!$ (labeled PGs on n vertices)
  - storing rank of start and endpoint of chord
    - $\rightsquigarrow$ encoding for labeled graph
    - $\rightsquigarrow$ $2n \log_2 n$ bits

- can prove $\log_2(|\mathcal{P}_n|) = n \log_2(n)(1 + o(1))$ (unlabeled PGs)

$\rightsquigarrow$ aim for $n \log_2 n$ bit data structure
which is allowed to *choose* vertex labeling

$\rightsquigarrow$ store permutation of endpoints!



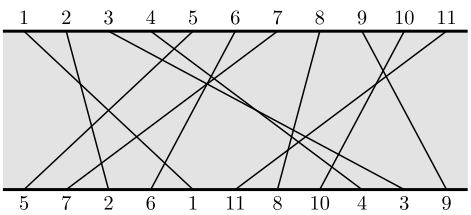
Permutation Graphs

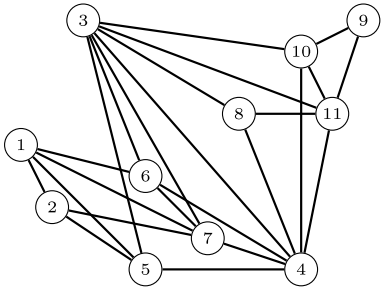
- $G = (V, E)$ is a **permutation graph (PG)** iff
  - V = chords between 2 parallel lines
  - adjacent iff chords intersect

  (A type of geometric intersection graph)
- PGs can be dense $\rightsquigarrow$ can't list edges
- $|\mathcal{P}^n| \leqslant n! \cdot n!$ (labeled PGs on n vertices)
  - storing rank of start and endpoint of chord
    - $\rightsquigarrow$ encoding for labeled graph
    - $\rightsquigarrow$ $2n \log_2 n$ bits
- can prove $\log_2(|\mathcal{P}_n|) = n \log_2(n)(1 + o(1))$ (unlabeled PGs)

$\rightsquigarrow$ aim for $n \log_2 n$ bit data structure which is allowed to *choose* vertex labeling

$\rightsquigarrow$ store permutation of endpoints!





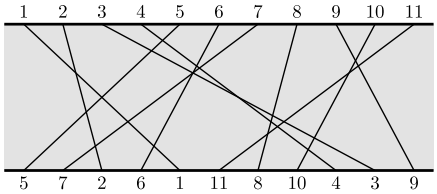
PG – Succinct encoding

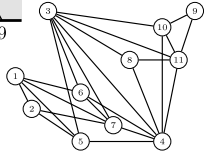
- List of lower endpoints π
  $5, 7, 2, 6, 1, 11, 8, 10, 4, 3, 9$

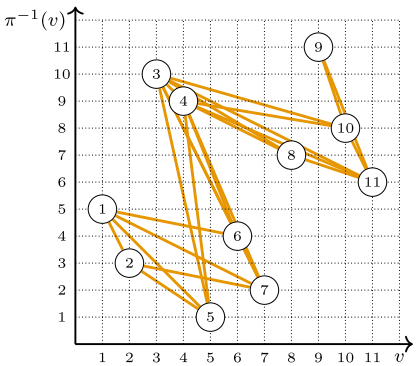
$\rightsquigarrow$ $n\lceil \log_2 n \rceil \sim n \log_2 n$ bits

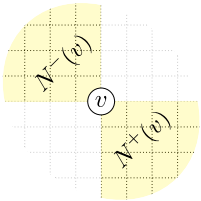
- Decoding algorithm trivial
- Encoding *given an intersection model*: trivial
- **Recognition of PG**
  - G is PG iff G and $\overline{G}$ (complement graph) are **comparability graphs**
  - G **comparability graph** iff
    $E(G)$ can be oriented **transitively**, i.e.,
    so that $u \to v \to w$ implies $u \to w$
  - can be found in $O(n + m)$ time

  McConnell, Spinrad: *Modular decomposition and transitive orientation*, Dis. Math. **1999**









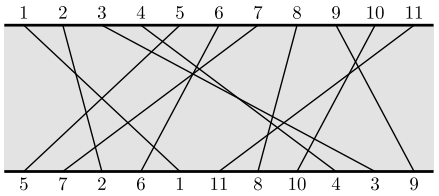
PG – Succinct encoding

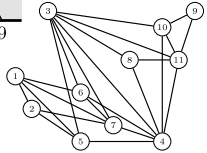
- List of lower endpoints π
  $5, 7, 2, 6, 1, 11, 8, 10, 4, 3, 9$

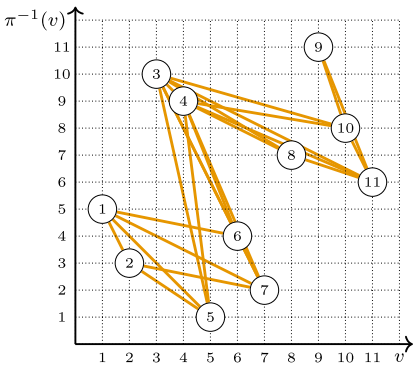
$\rightsquigarrow$ $n\lceil \log_2 n \rceil \sim n \log_2 n$ bits

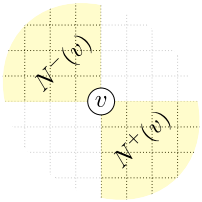
- Decoding algorithm trivial
- Encoding *given an intersection model*: trivial
- **Recognition of PG**
  - G is PG iff G and $\overline{G}$ (complement graph) are **comparability graphs**
  - G **comparability graph** iff
    $E(G)$ can be oriented **transitively**, i.e.,
    so that $u \to v \to w$ implies $u \to w$
  - can be found in $O(n + m)$ time

  McConnell, Spinrad: *Modular decomposition and transitive orientation*, Dis. Math. **1999**







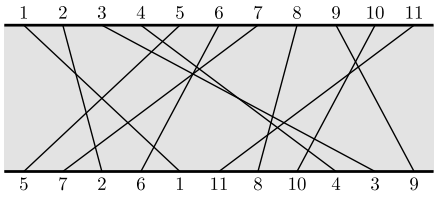


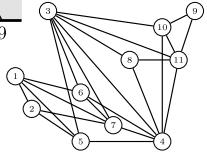
PG – Succinct encoding

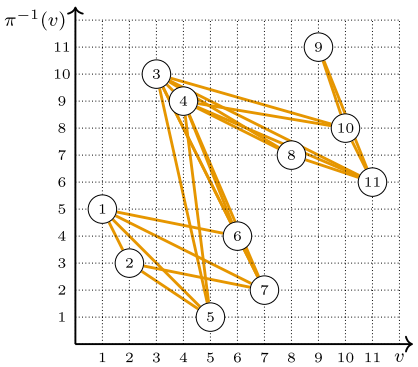
- List of lower endpoints π
  $5, 7, 2, 6, 1, 11, 8, 10, 4, 3, 9$

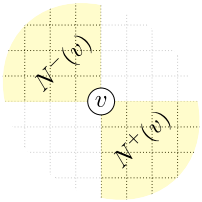
$\rightsquigarrow$ $n\lceil \log_2 n \rceil \sim n \log_2 n$ bits

- Decoding algorithm trivial
- Encoding *given an intersection model*: trivial
- **Recognition of PG**
  - G is PG iff G and $\overline{G}$ (complement graph) are **comparability graphs**
  - G **comparability graph** iff
    $E(G)$ can be oriented **transitively**, i.e.,
    so that $u \to v \to w$ implies $u \to w$
  - can be found in $O(n + m)$ time



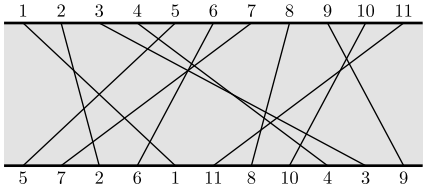


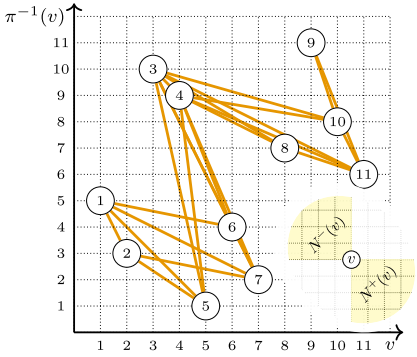




McConnell, Spinrad: *Modular decomposition and transitive orientation*, Dis. Math. **1999**

PG – Succinct data structure



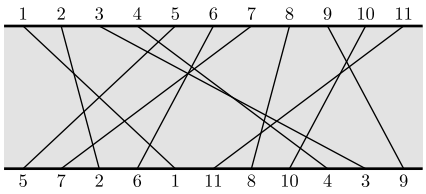


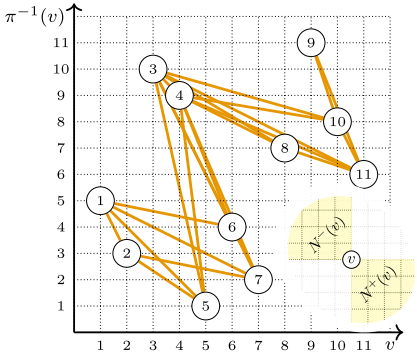
Can we support queries efficiently in $n \log_2(n)(1 + o(1))$ bits?

- **adjacency**
  $v < u$: *adjacent*(v, u) iff $\pi^{-1}(v) > \pi^{-1}(u)$ ✓
- **neighborhood / next-neighbor**
  - exploit point grid representation!
  - neighborhood = upper left and lower right quadrant
  - ⇝ given by 2-sided orthogonal range queries
  - (with a bit more effort)
    nextNeighbor via **range-min queries** over π^{-1}
  - ⇝ can be supported in $O(1)$ amortized time with $O(n)$ additional bits ✓
- **distance, shortest path**
  - shortest paths in PG have structure
  - ⇝ reduces to shortest paths in proper interval graphs

He, Munro, Nekrich, Wild, Wu: *Distance Oracles for Interval Graphs via Breadth-First Rank/Select in Succinct Trees*, ISAAC **2020**

PG – Succinct data structure



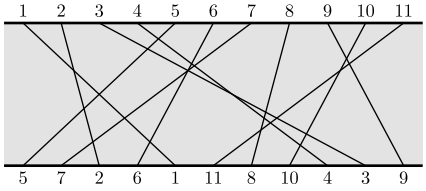


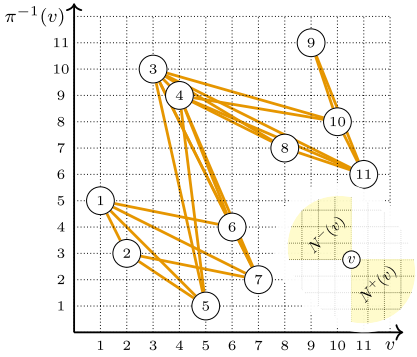
Can we support queries efficiently in $n \log_2(n)(1 + o(1))$ bits?

- **adjacency**
  $v < u$: *adjacent*(v, u) iff $\pi^{-1}(v) > \pi^{-1}(u)$ ✓
- **neighborhood / next-neighbor**
  - exploit point grid representation!
  - neighborhood = upper left and lower right quadrant
  - ⇝ given by 2-sided orthogonal range queries
  - (with a bit more effort)
    nextNeighbor via **range-min queries** over π^{-1}
  - ⇝ can be supported in $O(1)$ amortized time with $O(n)$ additional bits ✓
- **distance, shortest path**
  - shortest paths in PG have structure
  - ⇝ reduces to shortest paths in proper interval graphs

He, Munro, Nekrich, Wild, Wu: *Distance Oracles for Interval Graphs via Breadth-First Rank/Select in Succinct Trees*, ISAAC **2020**

PG – Succinct data structure





Can we support queries efficiently in $n \log_2(n)(1 + o(1))$ bits?

- **adjacency**
  $v < u$: *adjacent*(v, u) iff $\pi^{-1}(v) > \pi^{-1}(u)$ ✓
- **neighborhood / next-neighbor**
  - exploit point grid representation!
  - neighborhood = upper left and lower right quadrant
  - ⇝ given by 2-sided orthogonal range queries
  - (with a bit more effort)
    nextNeighbor via **range-min queries** over π^{-1}
  - ⇝ can be supported in $O(1)$ amortized time with $O(n)$ additional bits ✓
- **distance, shortest path**
  - shortest paths in PG have structure
  - ⇝ reduces to shortest paths in proper interval graphs

He, Munro, Nekrich, Wild, Wu: *Distance Oracles for Interval Graphs via Breadth-First Rank/Select in Succinct Trees*, ISAAC **2020**

PG – Result

	permutation graphs (a) by grid	permutation graphs (b) by array	bipartite permutation	circular permutation
Space Usage	$n \lg n + o(n \log n)$	$n \lg n + O(n)$	$2n + o(n)$	$n \lg n + O(n)$
Lower Bound	$\sim n \lg n$ [4]	$\sim n \lg n$ [4]	$\sim 2n$ [32]	$\sim n \lg n$ [4]
`adj`	$\log n / \log \log n$	1	1	1
`deg`	$\log n / \log \log n$	$deg + 1$	1	$deg + 1$
`nbrhood`	$(deg + 1) \log n / \log \log n$	$deg + 1$	$deg + 1$	$deg + 1$
`next_nbr`	$(deg + 1) \log n / \log \log n$	1 (amortized)	1	1 (amortized)
`spath`	$(dist + 1) \log n / \log \log n$	$dist + 1$	$dist + 1$	$dist + 1$
`spath_succ`	$\log n / \log \log n$	1	1	1
`dist`	$\log n / \log \log n$	1	$dist + 1$	1
Theorem	Thm. 3.1–(a)	Thm. 3.1–(b)	Thm. 5.1	Thm. 6.4



Tsakalidis, Wild, Zamaraev: *Succinct Permutation Graphs*, Algorithmica **2022**

Intrinsic cost of decentralization

- our succinct PGs in particular support (unweighted shortest-path) distance queries:
  - "exact distance oracle"
  - optimal $n \log_2(n)(1 + o(1))$ bit space
  - optimal $O(1)$ query time
- known lower bound for **distance *labeling schemes***:

> **Bazzaro & Gavoille (Thm 5.3)**
> Any distance labeling scheme for PGs requires $3 \log_2(n) - O(\log \log n)$ bits per vertex label.

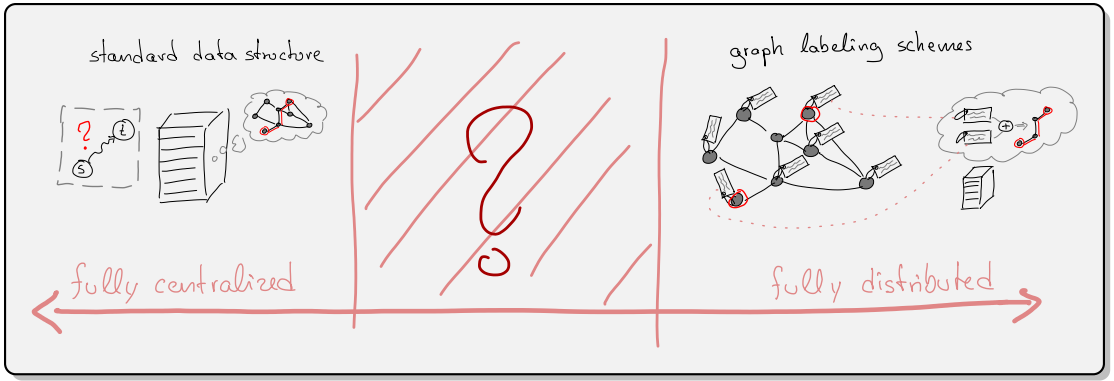


Bazzaro, Gavoille: *Localized and compact data-structure for comparability graphs,* Dis. Math. **2009**

$\rightsquigarrow$ Total space of a distributed exact distance oracle for PGs ($\sum$ labels) $\geqslant 3n \log_2(n) - O(n \log \log n)$

$\rightsquigarrow$ Example of a intrinsic space overhead of distributed computation

Between extremes





How much centralization is needed for succinctness (of PG distance oracles)?

Semi-distributed graph representations

(One possible) Model: $\langle L(n), D(n) \rangle$ **Semi-distributed graph representations**

- each vertex stores $L(n)$ bit label $\ell(v)$
- additionally $D(n)$ bit centralized data structure $d(G)$
- *distance*(v, u) computed from $\ell(v), \ell(u), d(G)$

$\rightsquigarrow$ total space $nL(n) + D(n)$



For **distances in PGs**, we know

- $\langle 0 \quad , n \log_2(n) \rangle$ (our succinct data structure)
- $\langle 9 \log_2(n) \quad , 0 \quad \rangle$ (labeling scheme of Bazzaro & Gavoille)
- $\langle (3-\varepsilon) \log_2(n), 0 \quad \rangle$ is **impossible** (labeling lower bound)

NEW

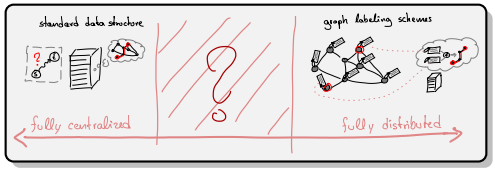
- $\langle 2 \log_2(n) \quad , O(n) \quad \rangle$ modifying our data structure
- $\rightsquigarrow$ $O(n)$ bit centralized space enough to circumvent lower bound
- $\rightsquigarrow$ Is $o(n)$ also enough? Other trade-offs?

Semi-distributed graph representations

(One possible) Model: $\langle \mathbf{L}(n), \mathbf{D}(n) \rangle$ **Semi-distributed graph representations**

- each vertex stores $L(n)$ bit label $\ell(v)$
- additionally $D(n)$ bit centralized data structure $d(G)$
- *distance*(v, u) computed from $\ell(v), \ell(u), d(G)$

$\rightsquigarrow$ total space $nL(n) + D(n)$



For **distances in PGs**, we know

- $\langle 0 \quad , n \log_2(n) \rangle$ (our succinct data structure)
- $\langle 9 \log_2(n) \quad , 0 \quad \rangle$ (labeling scheme of Bazzaro & Gavoille)
- $\langle (3-\varepsilon) \log_2(n), 0 \quad \rangle$ is **impossible** (labeling lower bound)
- NEW $\langle 2 \log_2(n) \quad , O(n) \quad \rangle$ modifying our data structure

$\rightsquigarrow$ $O(n)$ bit centralized space enough to circumvent lower bound

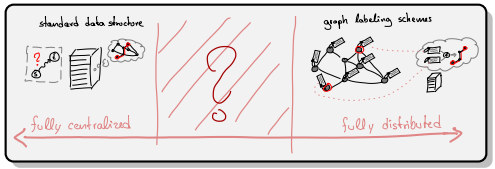
$\rightsquigarrow$ Is $o(n)$ also enough? Other trade-offs?

Semi-distributed graph representations

(One possible) Model: $\langle \mathbf{L}(n), \mathbf{D}(n) \rangle$ **Semi-distributed graph representations**

- each vertex stores $L(n)$ bit label $\ell(v)$
- additionally $D(n)$ bit centralized data structure $d(G)$
- *distance*(v, u) computed from $\ell(v), \ell(u), d(G)$

$\rightsquigarrow$ total space $nL(n) + D(n)$



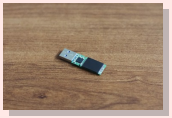
For **distances in PGs**, we know

- $\langle 0 \quad , n \log_2(n) \rangle$ (our succinct data structure)
- $\langle 9 \log_2(n) \quad , 0 \quad \rangle$ (labeling scheme of Bazzaro & Gavoille)
- $\langle (3-\varepsilon) \log_2(n), 0 \quad \rangle$ is **impossible** (labeling lower bound)
- NEW $\langle 2 \log_2(n) \quad , O(n) \quad \rangle$ modifying our data structure

$\rightsquigarrow$ $O(n)$ bit centralized space enough to circumvent lower bound

$\rightsquigarrow$ Is $o(n)$ also enough? Other trade-offs?

Outline



1 Introduction



2 Randomization in Labeling Schemes



3 Succinct Permutation Graphs



4 Conclusion





4 Conclusion

Questions

1. **Do all graph families have a succinct encoding?**
   - No, if we have to recognize graphs in the family.
   - If *encode* is not required to work correctly on $G \notin \mathcal{F}$??

   Spinrad: *Efficient graph representations,* Fields monographs **2003**

2. **Do all graph families have a succinct data structure (with adjacency queries)?**
   - No, for graphs with m edges and $n^\delta < m < n^{2-\delta}$, ($\delta > 0$ constant) when we want *adjacent* and *nextNeighbor* in $O(1)$ time

     But this is not a hereditary property.

     Farzan, Munro: *Succinct encoding of arbitrary graphs,* TCS **2013**
   - For hereditary classes?? Only faster-than-decompress queries??

3. **Do all graph families have a compact (adjacency) labeling scheme?**
   - **Recent news**: Resounding No!
   - Which families do??

   Hatami, Hatami: *The Implicit Graph Conjecture is False,* FOCS **2022**



DALL·E:
`a photorealistic image of a social network inside a large zip icon`

based on 

https://labs.openai.com/

Icons made by *Freepik*, *Gregor Cresnar*, *Those Icons*, *Smashicons*, *Good Ware*, *Pause08*, and *Madebyoliver* from `www.flaticon.com`.

Vector graphics from Pressfoto, brgfx, macrovector and Jannoon028 on freepik.com
Other photos from `www.pixabay.com`.