Sparsity in Practice

a bit of introspection

Felix Reidl

Dagstuhl 2021

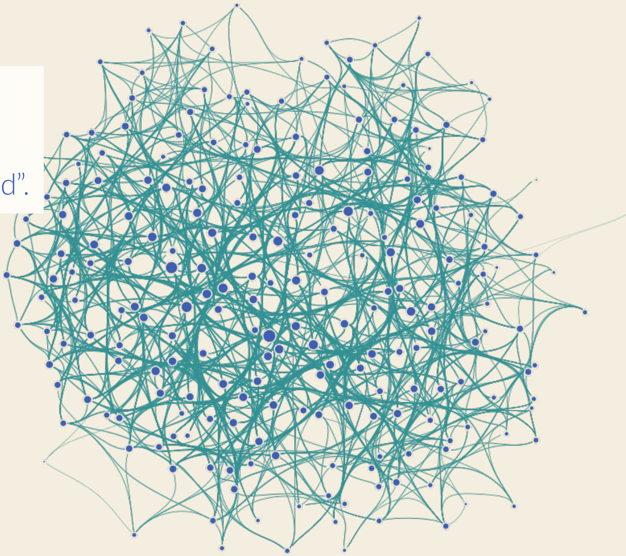
Part I

The work so far

Residence hall

- Student in ANU Hall
- Friendship

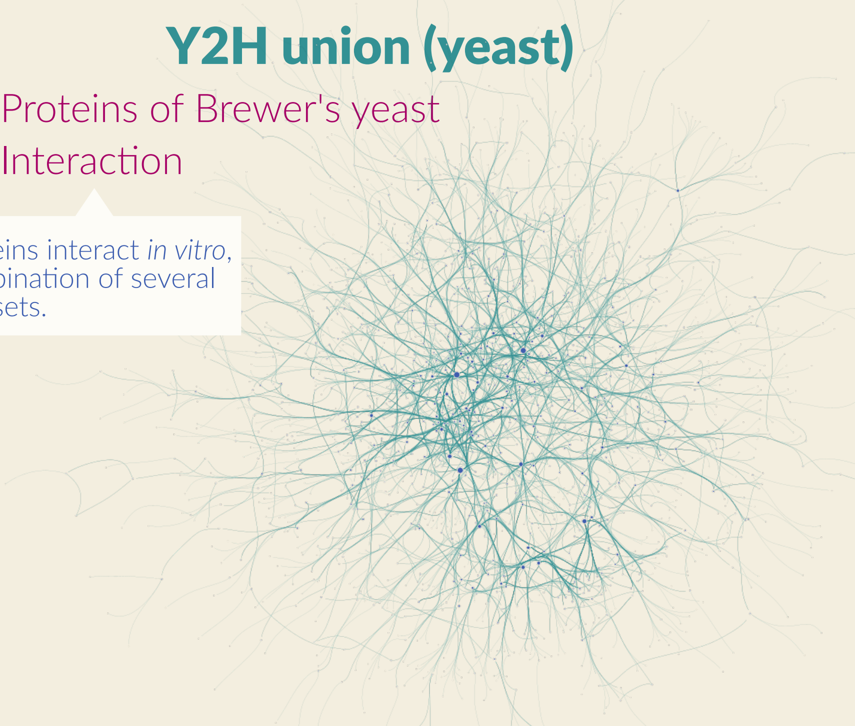
Collected via interviews by Cynthia Webster, ranked as "best friend", "close friend", and "friend".



Y2H union (yeast)

- Proteins of Brewer's yeast
- Interaction

Proteins interact *in vitro*, combination of several datasets.



Diseasome

- Gene / Genetic disorder
- Correlation: expression/disease

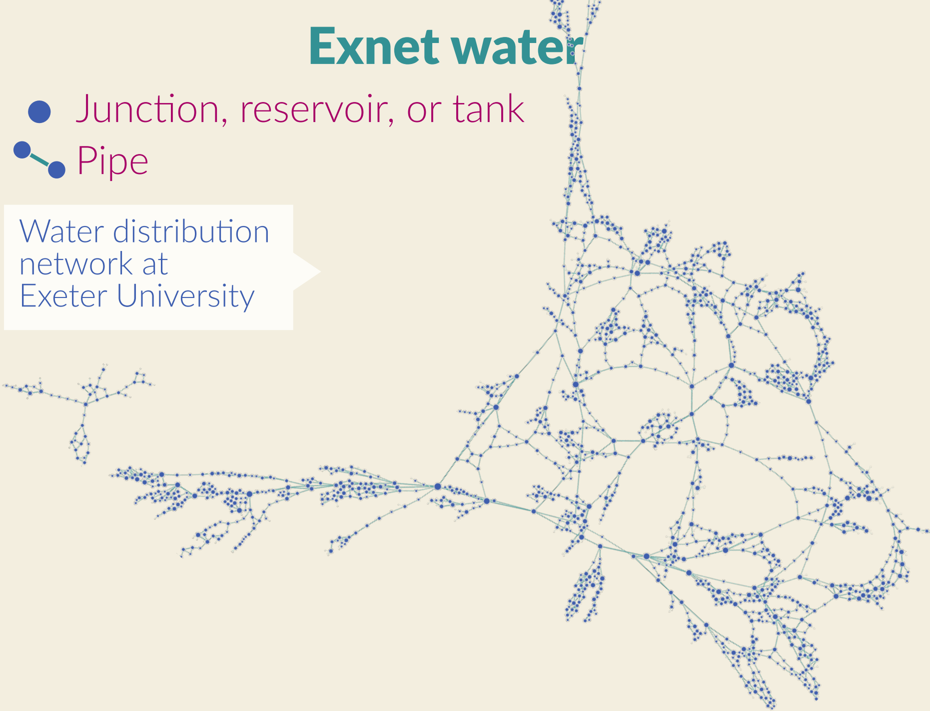
Extracted from OMIM database.

Exnet water

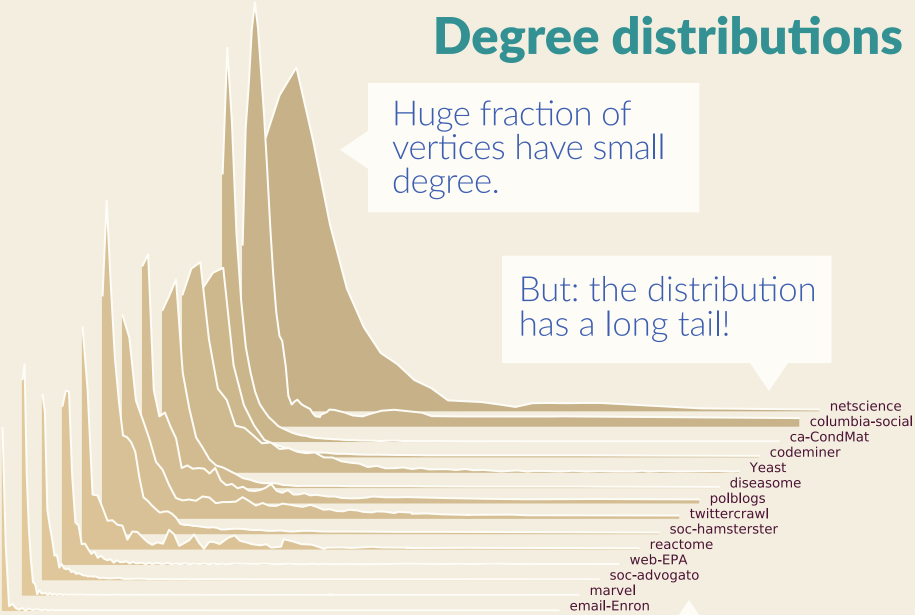
● Junction, reservoir, or tank

●—● Pipe

Water distribution network at Exeter University



Degree distributions

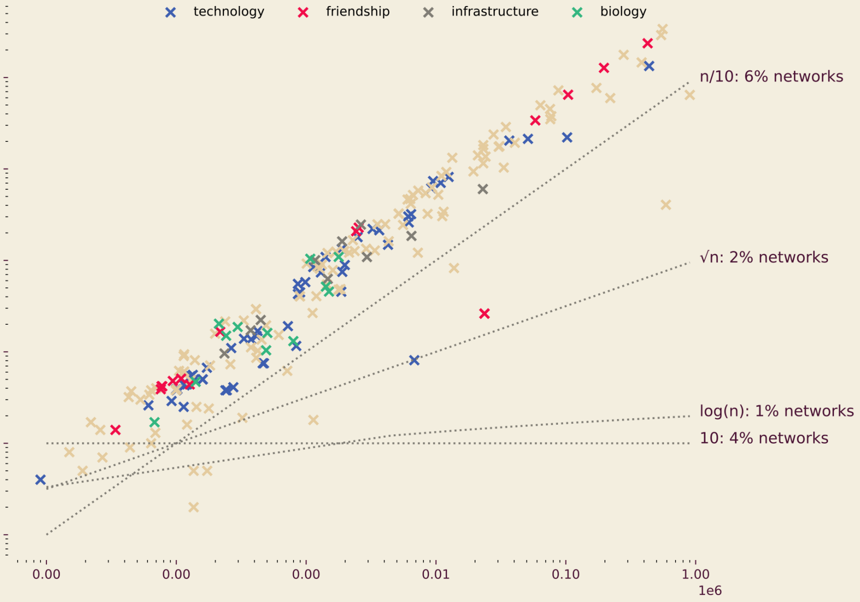


Huge fraction of vertices have small degree.

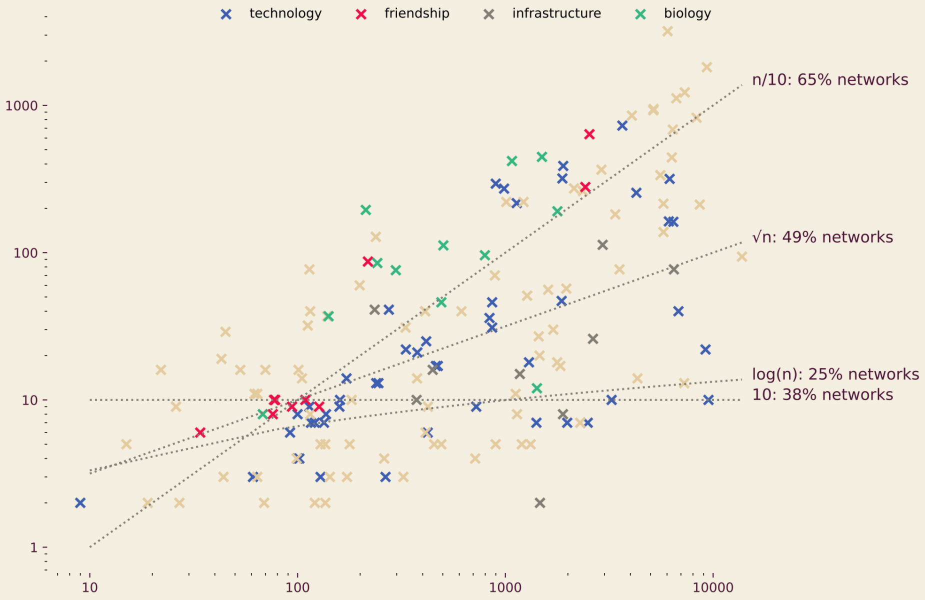
But: the distribution has a long tail!

More extreme in large networks!

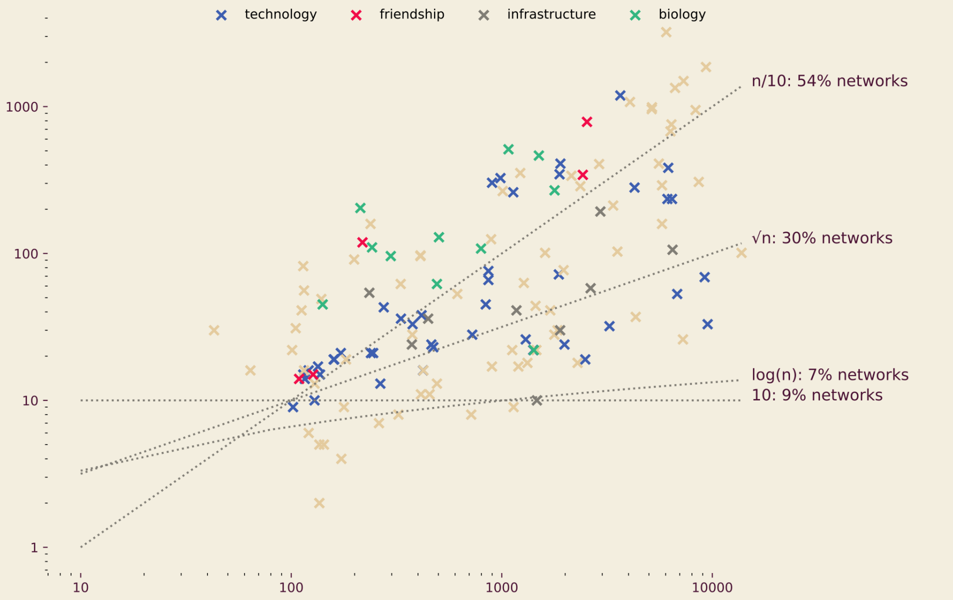
Real vertex cover



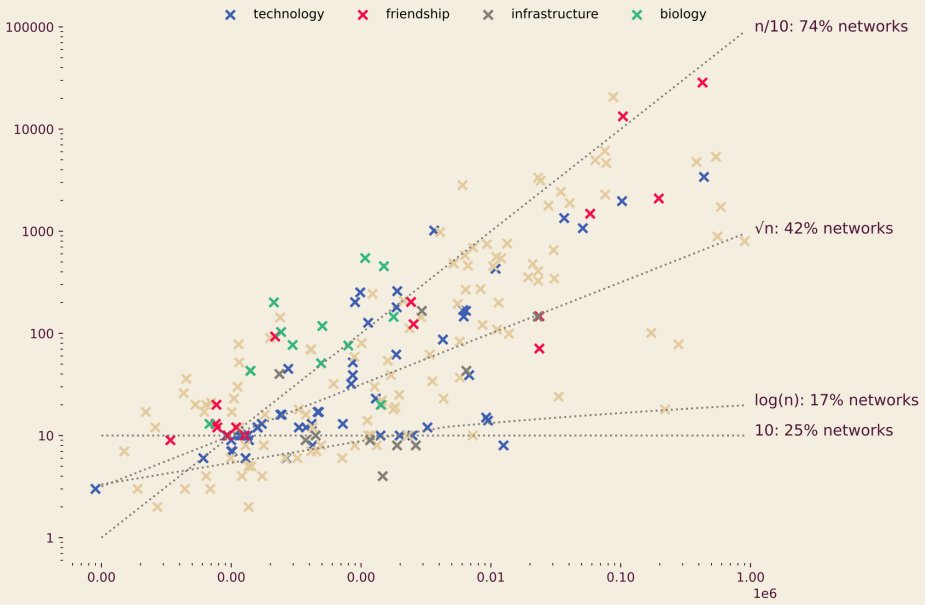
Real treewidth



Real treedepth

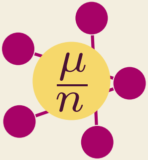


Real wcol_3

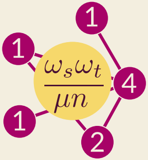


Random model sparsity

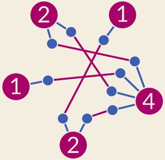
$G(n, \frac{\mu}{n})$



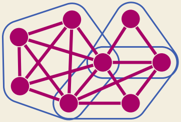
$G^{\mathrm{CL}}(D_n)$



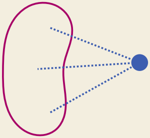
$G^{\mathrm{CF}}(D_n)$



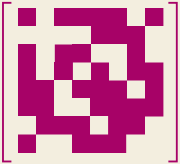
$G^{\mathrm{RIG}}(n, \alpha, \beta, \gamma)$



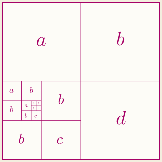
$G^{\mathrm{BA}}(n, n_0, k)$



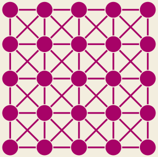
$G^{\mathrm{SGK}}(k, \alpha, \ldots, \gamma)$



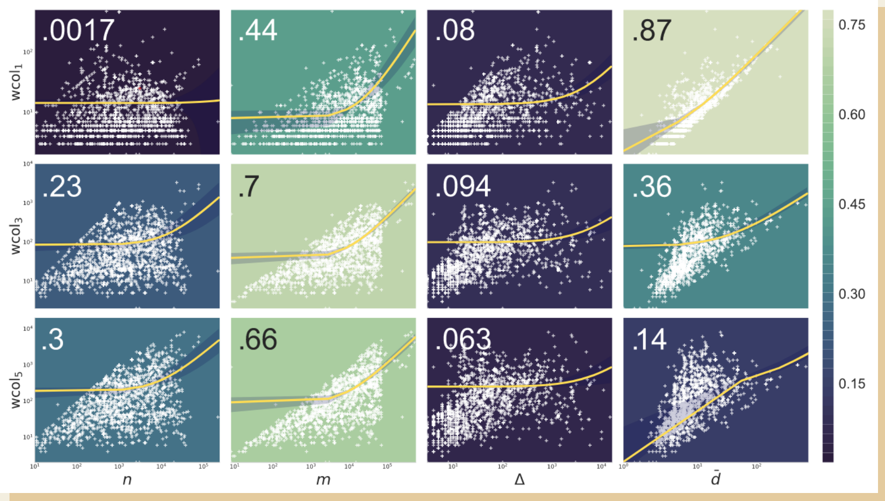
$G^{\mathrm{RMAT}}(k, m, a, b, c)$



$G^{\mathrm{KL}}(n, p, q, \gamma)$

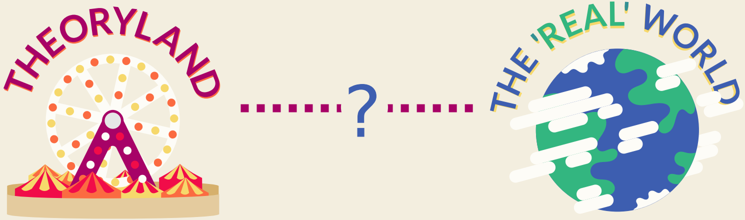


Real structural sparseness



W. Nadara, M. Pilipczuk, R. Rabinovich, FR, S. Siebertz:
Empirical Evaluation of Approximation Algorithms for Generalized Graph Coloring and Uniform Quasi-Wideness.
SEA 2018: 14:1-14:16

A hard-learnt lesson



Nešetřil J, de Mendez PO, Wood DR.
Characterisations and examples of graph classes with bounded expansion.
European Journal of Combinatorics. 2012 Apr 30;33(3):350-73.

Demaine ED, FR, Rossmanith P, Sánchez Villaamil F, Sikdar S, Sullivan BD.
Structural sparsity of complex networks: Bounded expansion in random models and real-world graphs.
Journal of Computer and System Sciences. 2019 May 24.

Farrell M, Goodrich TD, Lemons N, FR, Sánchez Villaamil F, Sullivan BD.
Hyperbolicity, degeneracy, and expansion of random intersection graphs.
InInternational Workshop on Algorithms and Models for the Web-Graph 2015 Dec 10 (pp. 29-41). Springer, Cham.

W. Nadara, M. Pilipczuk, R. Rabinovich, FR, S. Siebertz:
Empirical Evaluation of Approximation Algorithms for Generalized Graph Coloring and Uniform Quasi-Wideness.
SEA 2018: 14:1-14:16

A hard-learnt lesson

?

Nešetřil J, de Mendez PO, Wood DR.
Characterisations and examples of graph classes with bounded expansion.
European Journal of Combinatorics. 2012 Apr 30;33(3):350-73.

Demaine ED, FR, Rossmanith P, Sánchez Villaamil F, Sikdar S, Sullivan BD.
Structural sparsity of complex networks: Bounded expansion in random models and real-world graphs.
Journal of Computer and System Sciences. 2019 May 24.

Farrell M, Goodrich TD, Lemons N, FR, Sánchez Villaamil F, Sullivan BD.
Hyperbolicity, degeneracy, and expansion of random intersection graphs.
InInternational Workshop on Algorithms and Models for the Web-Graph 2015 Dec 10 (pp. 29-41). Springer, Cham.

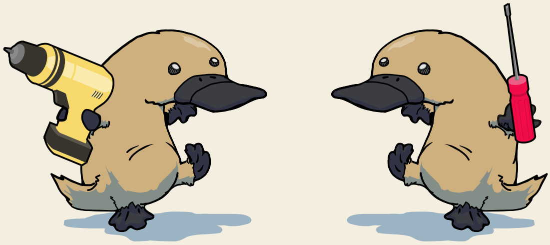
W. Nadara, M. Pilipczuk, R. Rabinovich, FR, S. Siebertz:
Empirical Evaluation of Approximation Algorithms for Generalized Graph Coloring and Uniform Quasi-Wideness.
SEA 2018: 14:1-14:16

No one* cares.

*not enough people

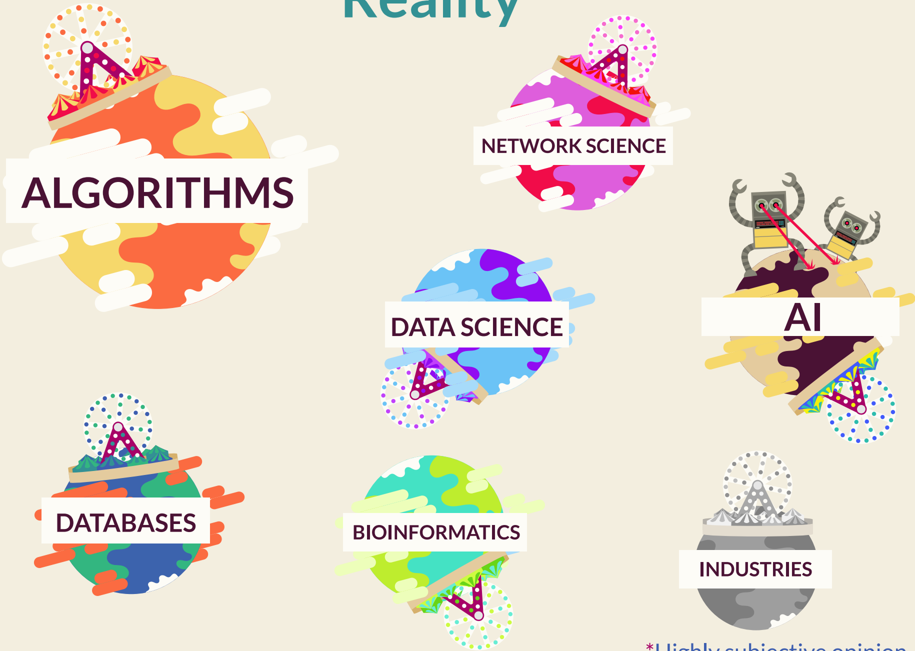
Part II

Reflection



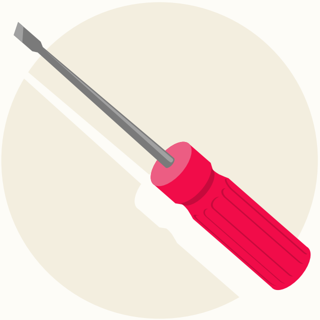
A false dichotomy

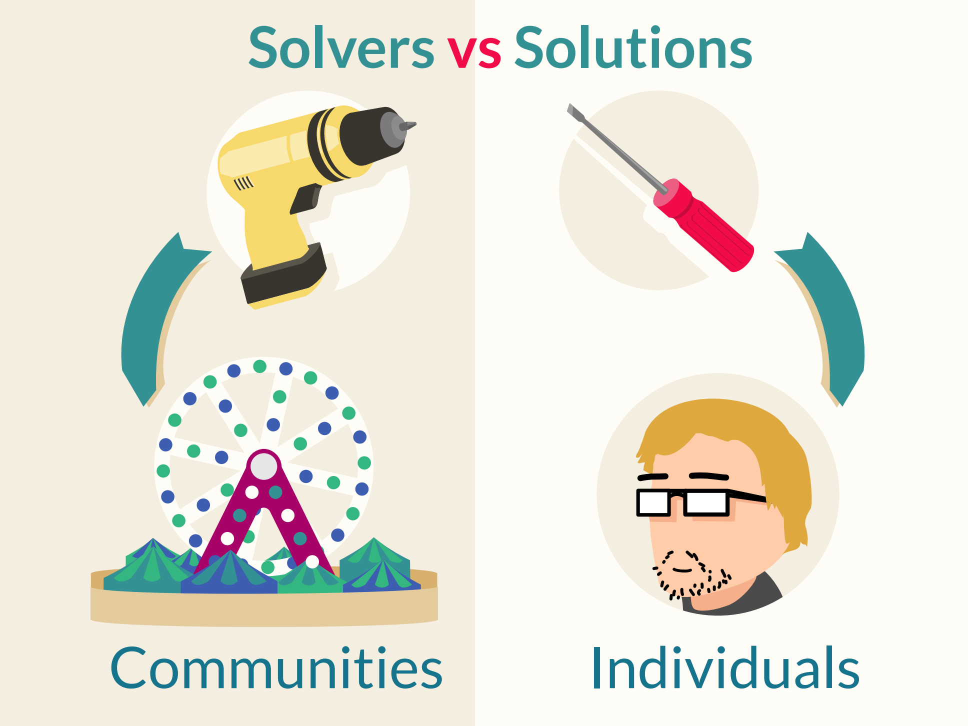
Reality*



*Highly subjective opinion

Solvers vs Solutions




Solvers vs Solutions
Communities
Individuals

Solvers vs Solutions

Diffuse problems

Competing views

Often sceptical of new approaches

Must be very user-friendly

High maintenance

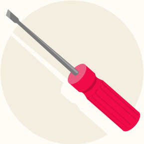
Concrete problems

Single perspective

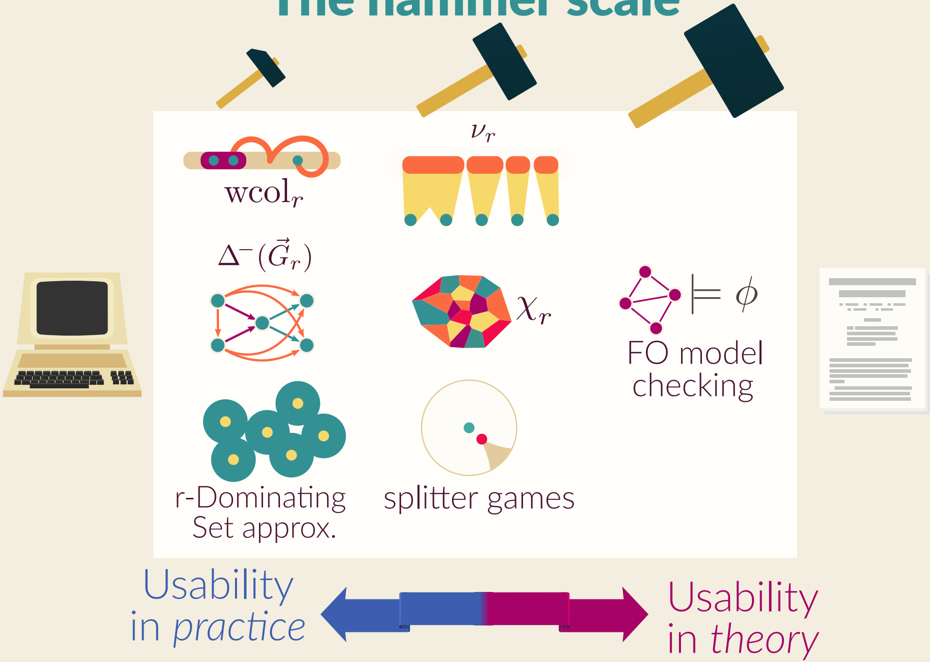
Appreciate new approaches

Usable for stakeholder

Maintenance by stakeholder



The hammer scale



Big hammers don't implement

Medium hammers don't scale

O'Brien MP, Sullivan BD.
Experimental evaluation of counting subgraph isomorphisms in classes of bounded expansion.
arXiv preprint arXiv:1712.06690. 2017 Dec 18.

Small hammers might just work!

Nadara W, Pilipczuk M, Rabinovich R, Reidl F, Siebertz S.
Empirical evaluation of approximation algorithms for generalized graph coloring and uniform quasi-wideness.
Journal of Experimental Algorithmics (JEA). 2019 Dec 10;24:1-34.

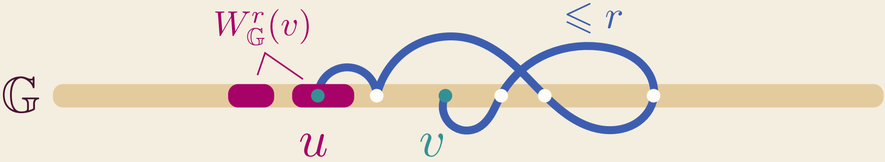
Brown CT, Moritz D, O'Brien MP, Reidl F, Reiter T, Sullivan BD.
Exploring neighborhoods in large metagenome assembly graphs using spacegraphcats reveals hidden sequence diversity. Genome biology. 2020 Dec;21(1):1-6.

github.com/spacegraphcats/spacegraphcats

Reidl F, Sullivan BD. **A color-avoiding approach to subgraph counting in bounded expansion classes**.
arXiv preprint arXiv:2001.05236. 2020 Jan 15.

github.com/theoryinpractice/mandoline

Weak colouring & bounded expansion

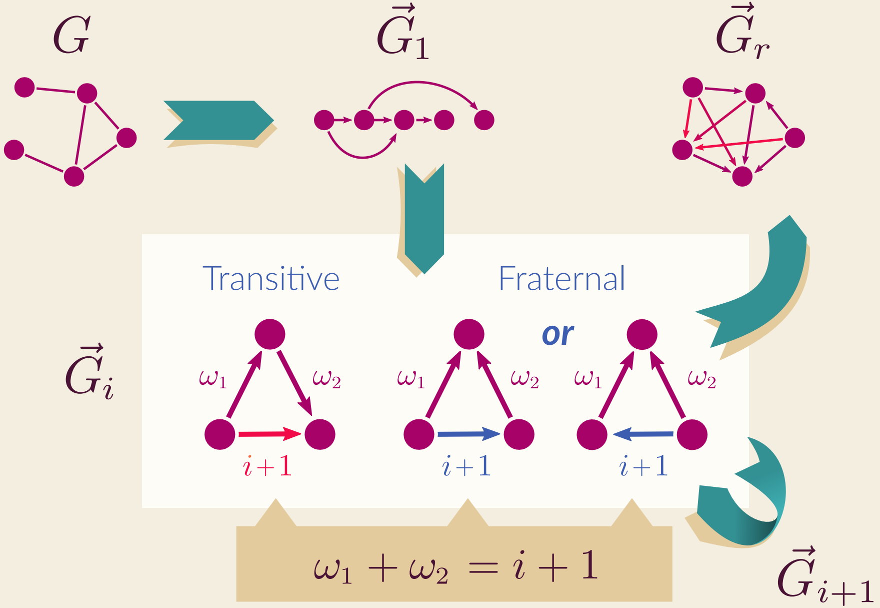


u is **weakly r-reachable** from v if there exists a path from v to u of length at most r such that u is the path's leftmost vertex.

$$\mathrm{wcol}_r(G) := \min_{\mathbb{G} \in \Pi(G)} \max_{v \in G} |W^r_{\mathbb{G}}(v)|$$

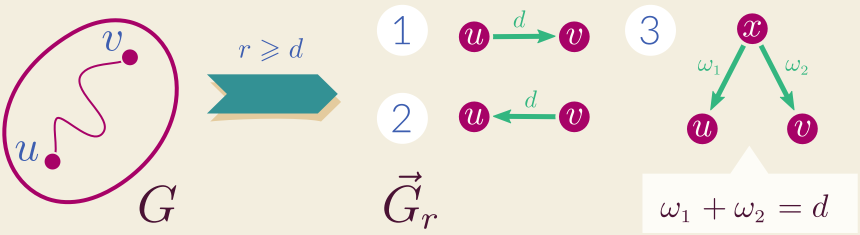
A graph class has bounded expansion iff it is wcol_r-bounded.

dtf-augmentations



Distances under dtf-augmentations

Let u and v be at distance d in G :



Pairs at distance at most r in the original graph have distance at most two in the r^th augmentation.

B.E. & dtf-augmentations

There exist two (horrible) polynomials P and Q such that:

$$\chi_r(G) \leqslant P\big(\tilde{\nabla}_{(2\log r)^r}(G)\big)$$

$$\Delta^-(\vec{G}_r) \leqslant Q\big(\tilde{\nabla}_r(G)\Delta^-(\vec{G}_1)\big)$$

A graph class has bounded expansion iff it is $\Delta^-(\vec{G}_r)$-bounded.

We can compute dtf-augmenations in linear time (in bounded expansion classes)

Applications & Algorithms

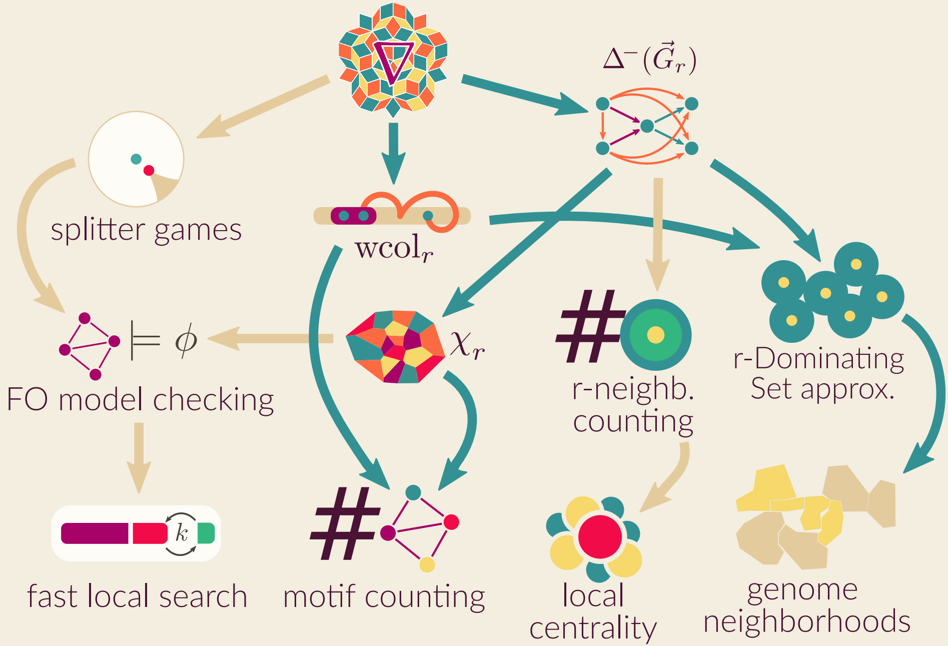


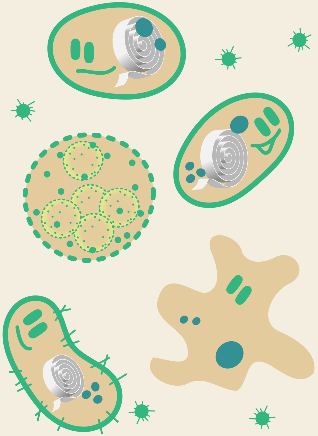
Exhibit A

CATLAS

Metagenome exploration
using hierarchical domination
of de-Bruijn graphs

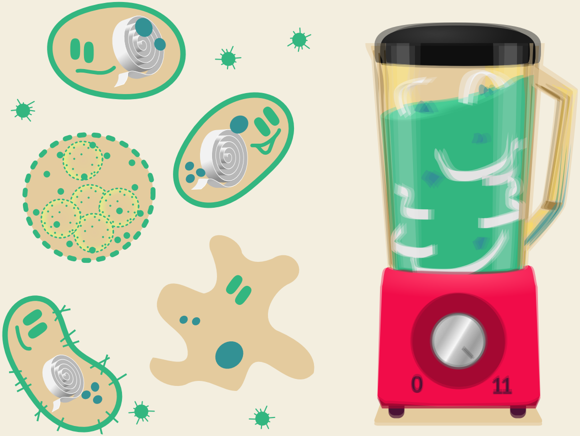
Joint work with C. Titus Brown, Dominik Moritz,
Michael P. O'Brien, Taylor Reiter, Blair D. Sullivan

Metagenomics



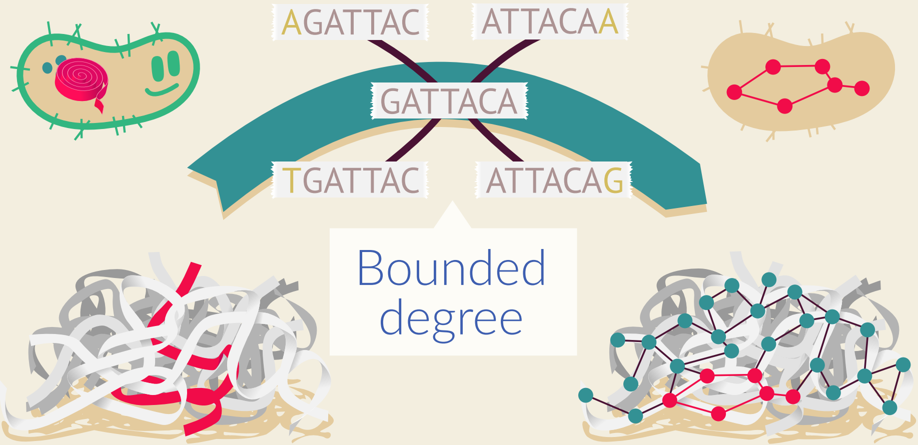
Metagenomics

Metagenomics

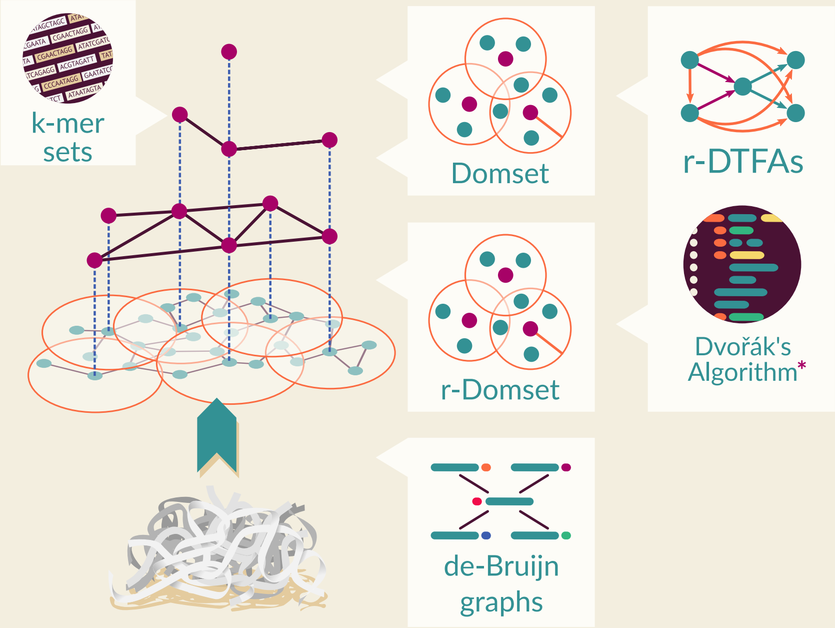


Metagenomics

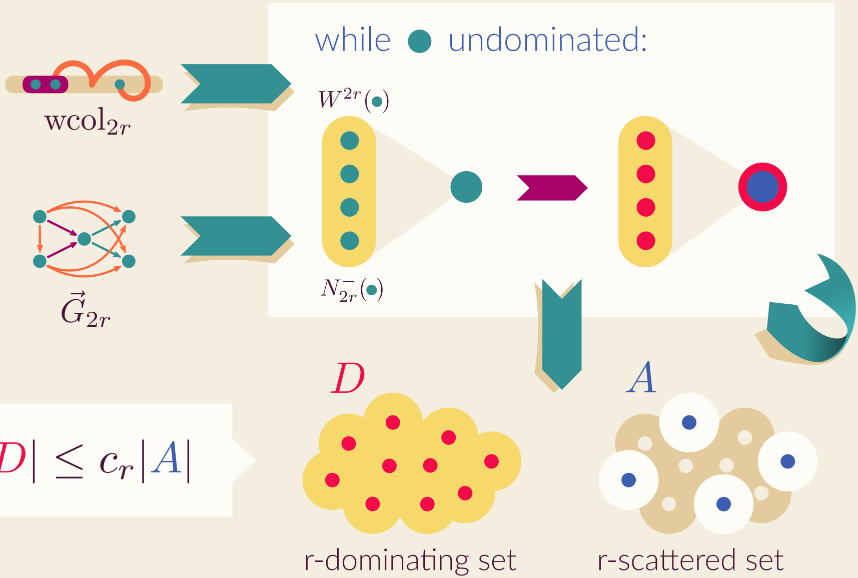
De-Bruijn graphs



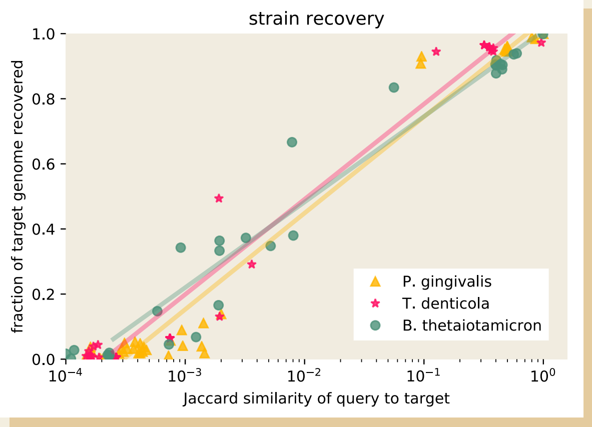
CATLAS Overview

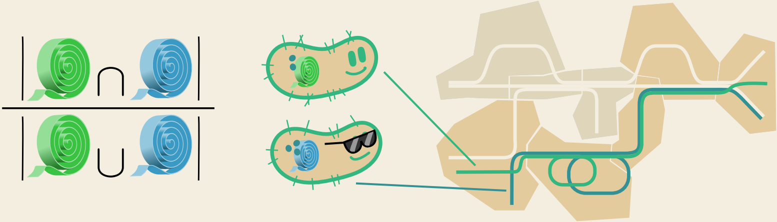


Reminder: Dvořák's algorithm



CATLAS-1 Results





What really matters

“The efficient computation of dominating sets will open up a whole new range of possibilities in bioinformatics.”

C. Titus Brown, Associate Professor at UC Davis

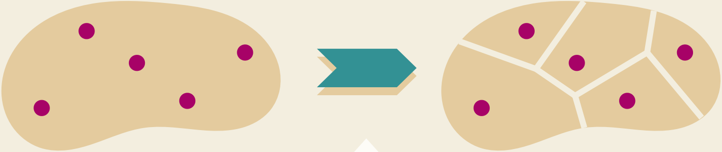
“spacegraphcats will transform the way biologists interact with genome assemblies.
It allows us to access previously discarded sequencing information thereby allowing more robust functional characterization.”

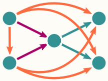
Taylor Reiter, his much more eloquent post-doc

Engineering: efficiency

Lesson: Stick to the 'sparsity methodology'.

Example: computing partition from r-domset

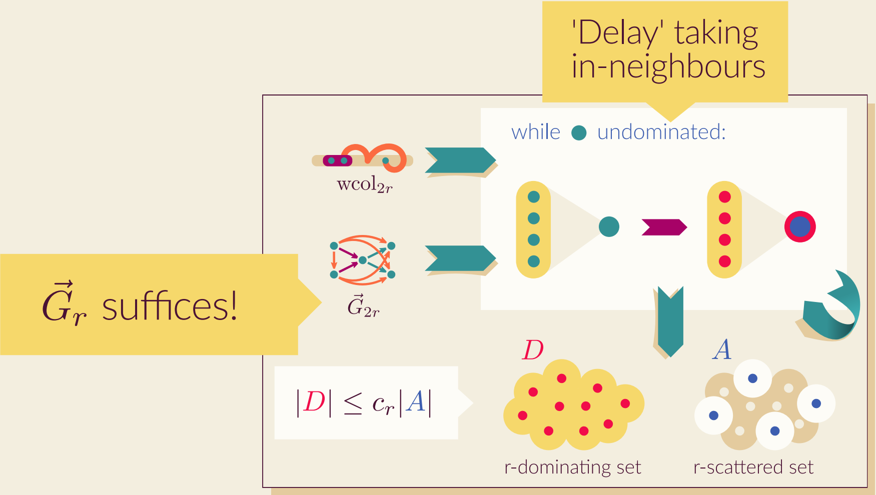




We can do this in linear time using the already computed dtf-augmentation

Engineering: efficiency

Lesson: Practical issues inspire theoretical questions.



Collaboration

Lesson: Expect roles to shift over time.

Then

Now

Next steps

1 Faster language

Believe it or not, so far this is all done in Python!

2 Improve network partition

E.g. consider additional constraints

3 Whatever our collaborators need!

? Contracted de-Bruijn graphs

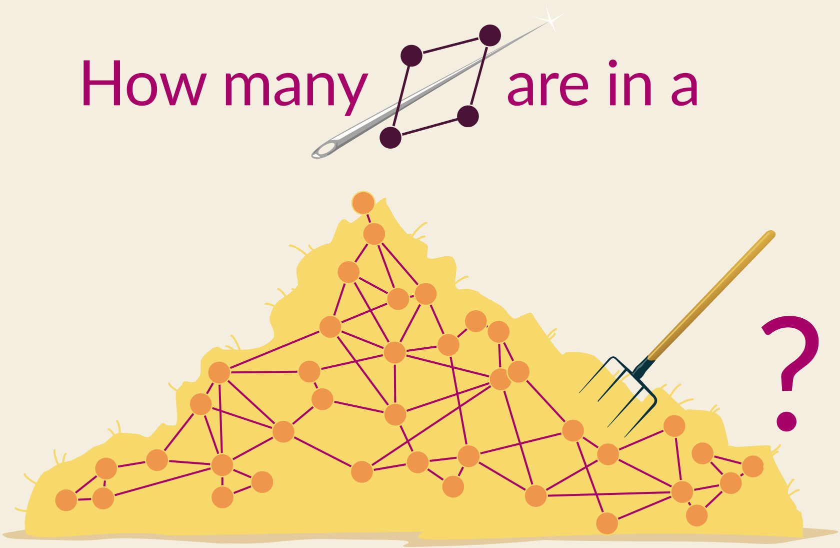
These graphs have bounded degree, but probably much more structure that we could exploit!

Exhibit B

Mandoline

Motif counting using generalized colourings

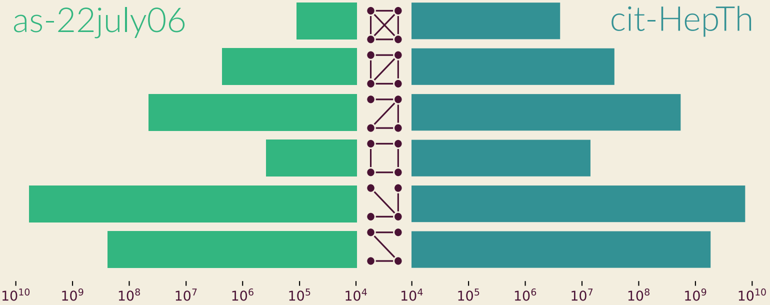
Joint work with Blair D. Sullivan


How many
are in a
?


How many
are in a
?

Graphlets

We want to count all (connected) induced subgraphs up to a given size.



The **graphlet degree distribution** or the **graphlet degree** can be used to compare networks.

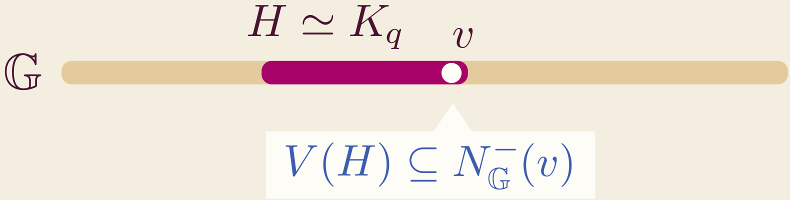
Pržulj N, Corneil DG, Jurisica I.
Modeling interactome: scale-free or geometric?.
Bioinformatics. 2004 Jul 29;20(18):3508-15

Pržulj N. B**iological network comparison using graphlet degree distribution.**
Bioinformatics. 2007 Jan 15;23(2):e177-83.

Let's start with something easy!

We count cliques in a d-degenerate graph.

Observation: every clique is contained in the left-neighbourhood of its *last* vertex.



Let's start with something easy!

We count cliques in a d-degenerate graph.

Observation: every clique is contained in the left-neighbourhood of its *last* vertex.

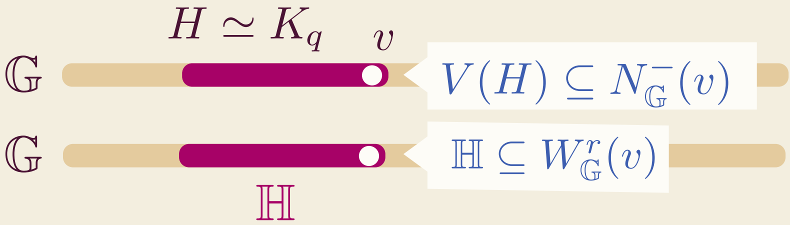
$$V(H) \subseteq N^-_{\mathbb{G}}(v)$$

Therefore we can enumerate all cliques by enumerating all cliques in $N^-(v)$ for all $v \in G$!

$O(2^d n)$ time!

Does it blend?

Can we ‘lift’ this algorithm to wcol?

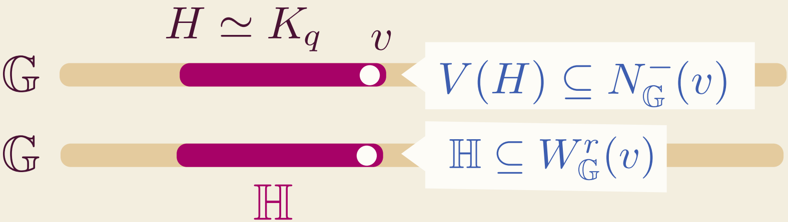


1. What is the ‘last’ vertex of H?

Enumerate all orderings $\mathbb{H}$ of H.

Does it blend?

Can we ‘lift’ this algorithm to wcol?



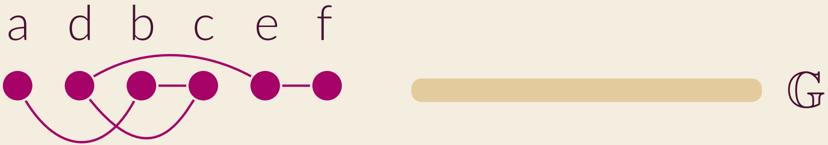
1. What is the ‘last’ vertex of H?

   Enumerate all orderings $\mathbb{H}$ of H.

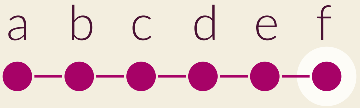
2. Does $\mathbb{H} \subseteq W^r_{\mathbb{G}}(v)$ actually hold?

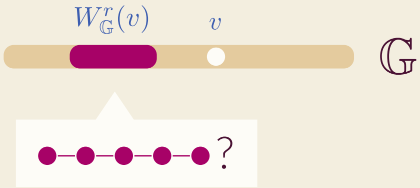
   Only sometimes!

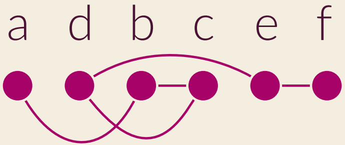
Two ways to order a P_6

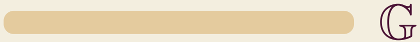


Two ways to order a P_6

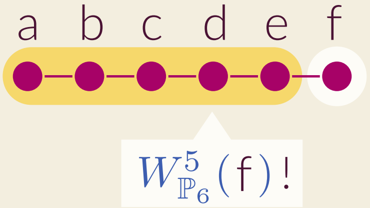


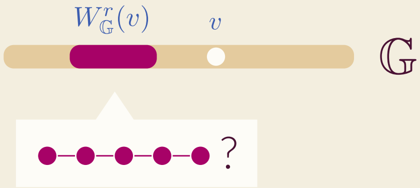


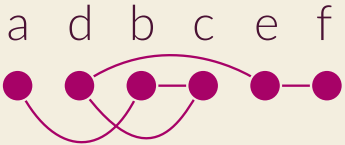


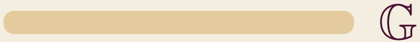


Two ways to order a P_6

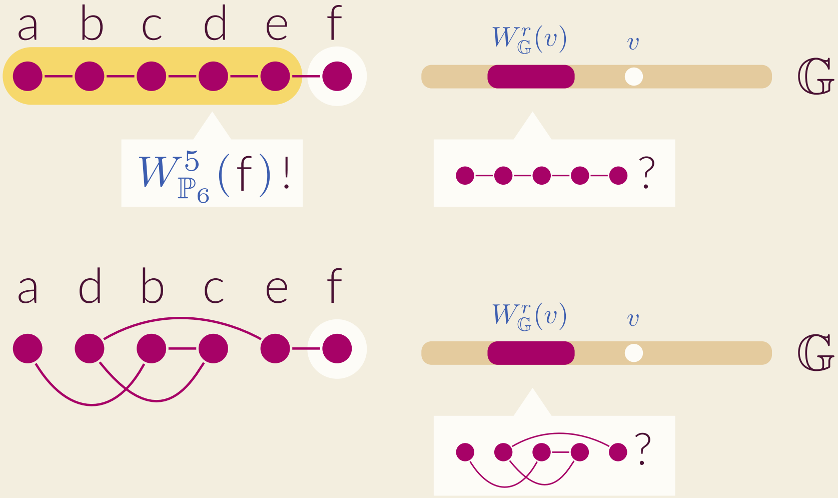




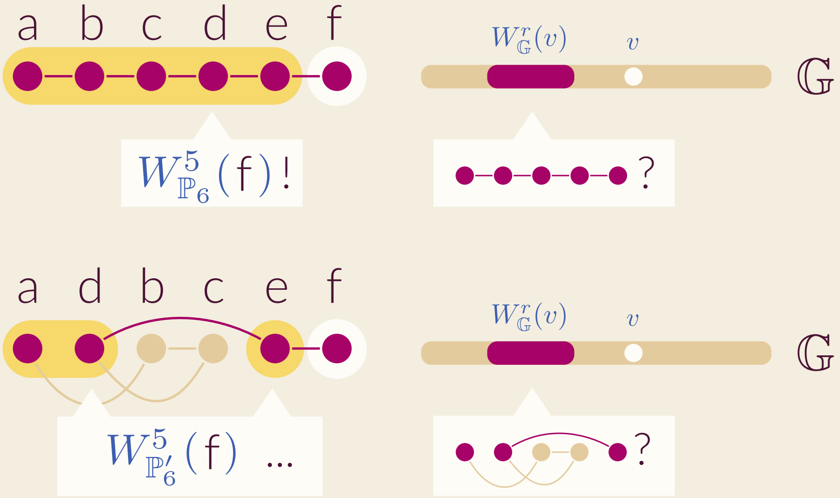




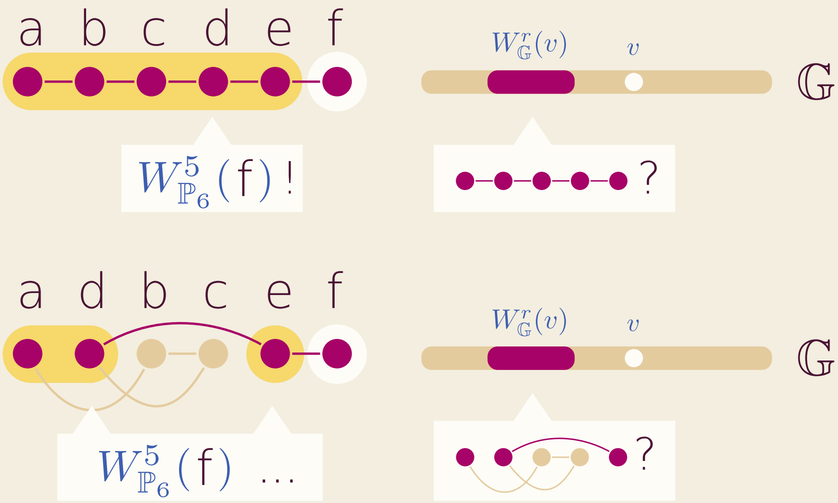
Two ways to order a P_6



Two ways to order a P_6

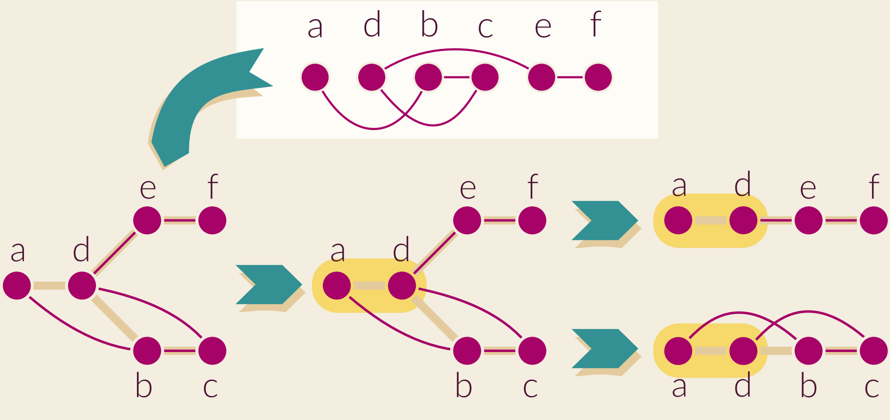


Two ways to order a P_6



Is there a nice formalization of this property?

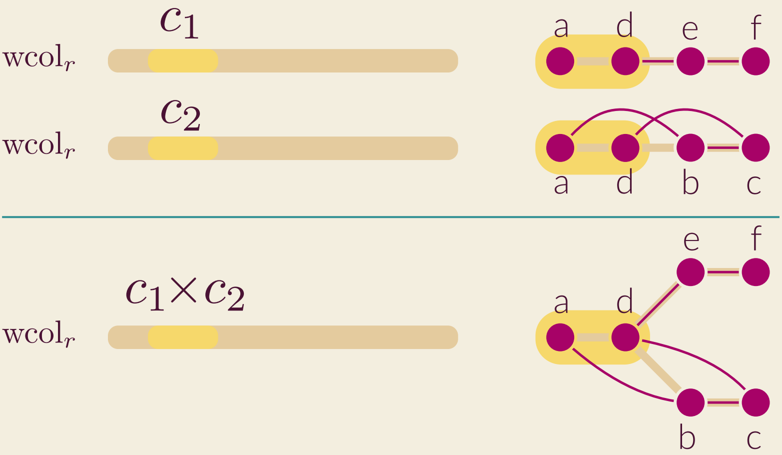
Decomposition!



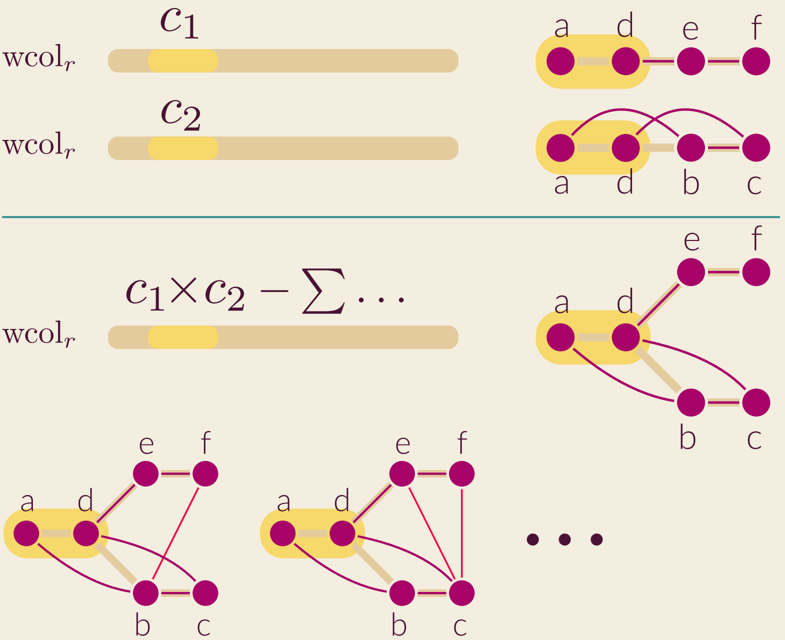
We can count linear pieces!

Progress! These pieces are linear!

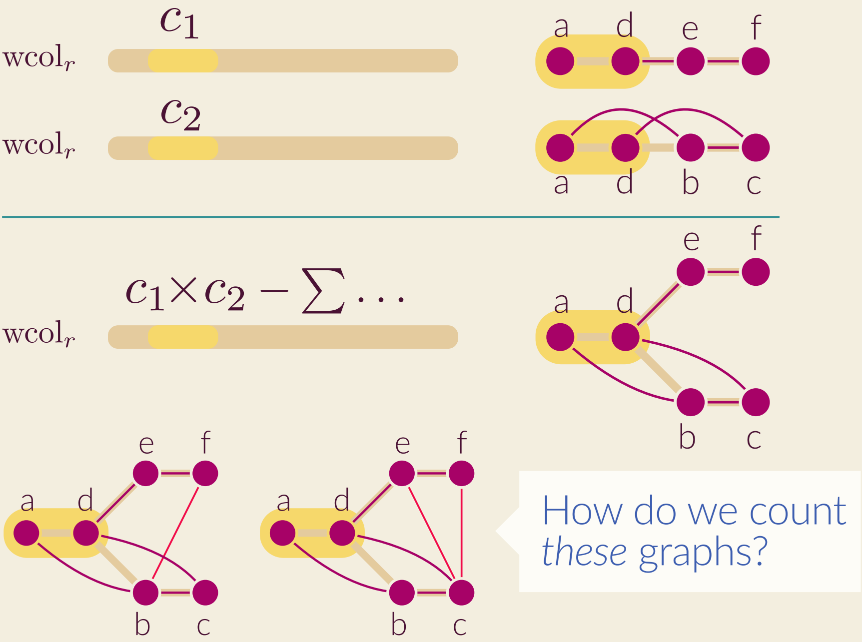
Count & combine!



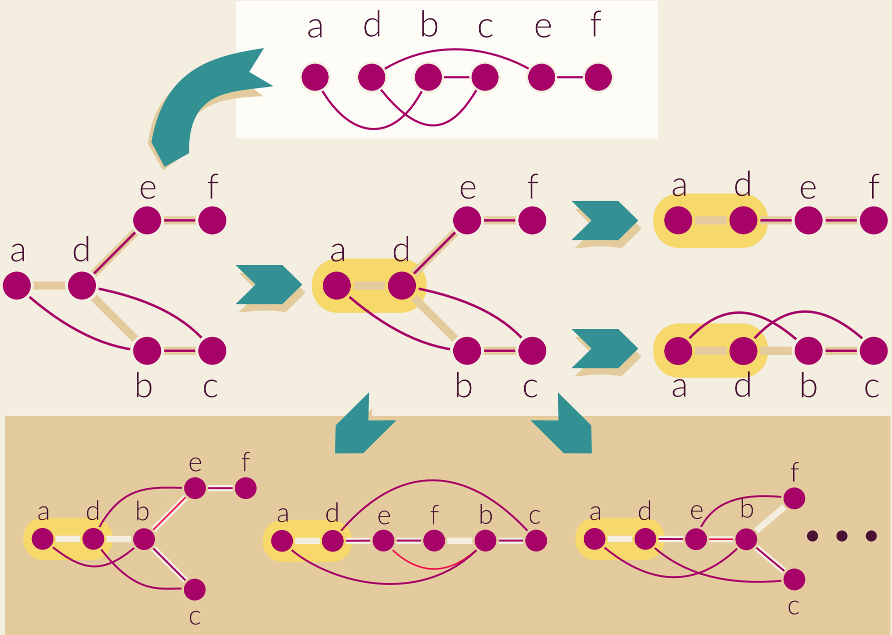
Count & combine!



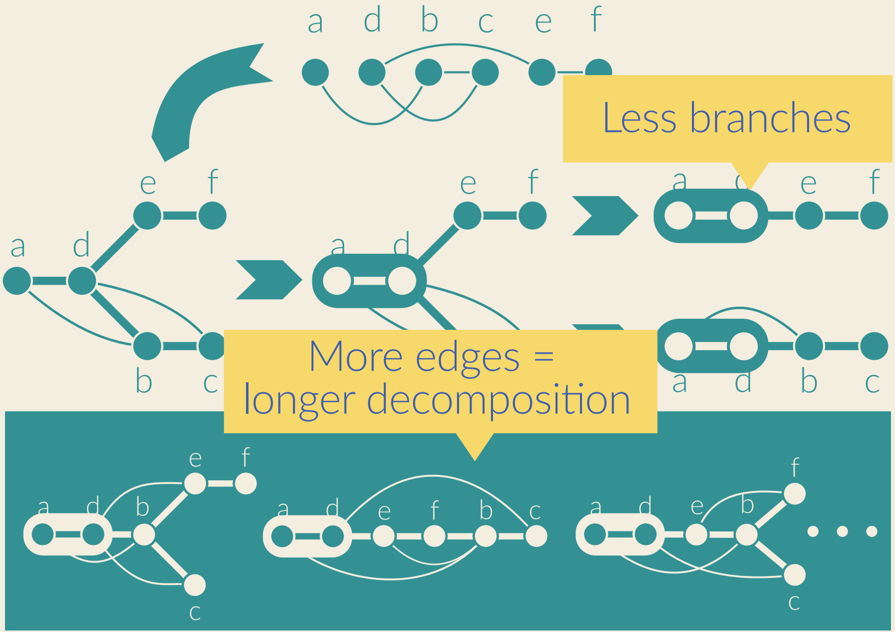
Count & combine!



Decomposition!



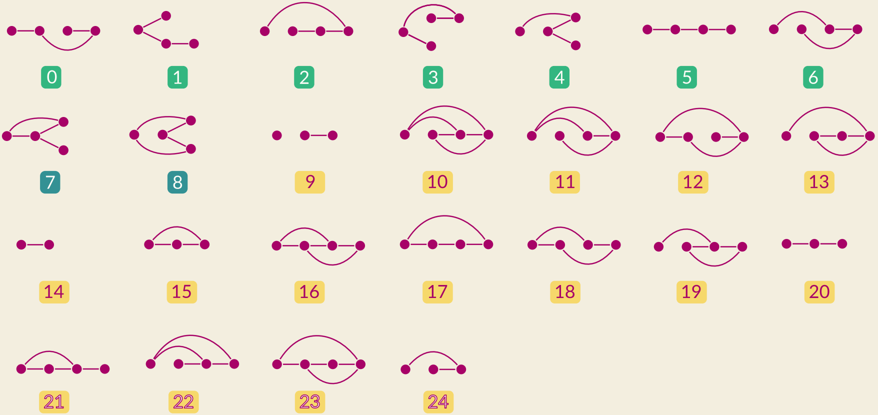
Decomposition!



Counting P_4s using wcol_3

Lemma 6. *Let* $\mathbf{H} \in \mathcal{H}$ *be a (non-linear) pattern relaxation and let* $\mathbf{H}_1 \oplus_{\bar{x}} \mathbf{H}_2 = \mathbf{H}$*. Fix an ordered vertex set* $\bar{y} \in \mathbb{G}$ *such that* $\mathbf{H}[\bar{x}] \simeq \mathbb{G}[\bar{y}]$*. Then*

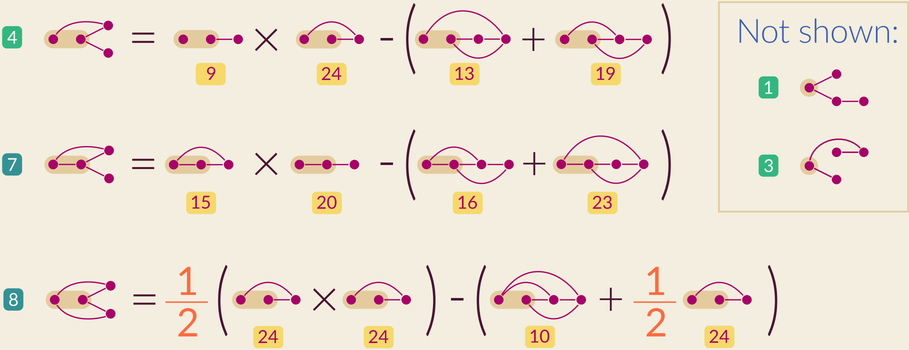
$$\underset{\bar{x}\mapsto\bar{y}}{\#}(\mathbf{H},\mathbb{G}) = \underset{\bar{x}\mapsto\bar{y}}{\#}(\mathbf{H}_1,\mathbb{G})\underset{\bar{x}\mapsto\bar{y}}{\#}(\mathbf{H}_2,\mathbb{G}) - \sum_{\mathbf{D}\in\mathcal{D}(\mathbf{H}_1,\mathbf{H}_2)} \frac{\underset{\bar{x}\mapsto\bar{x}}{\#}(\mathbf{H},\mathbf{D}\mid\mathbf{H}_1,\mathbf{H}_2)\underset{\bar{x}\mapsto\bar{y}}{\#}(\mathbf{D},\mathbb{G})}{\underset{\bar{x}\mapsto\bar{x}}{\#}(\mathbf{D},\mathbf{D})}.$$



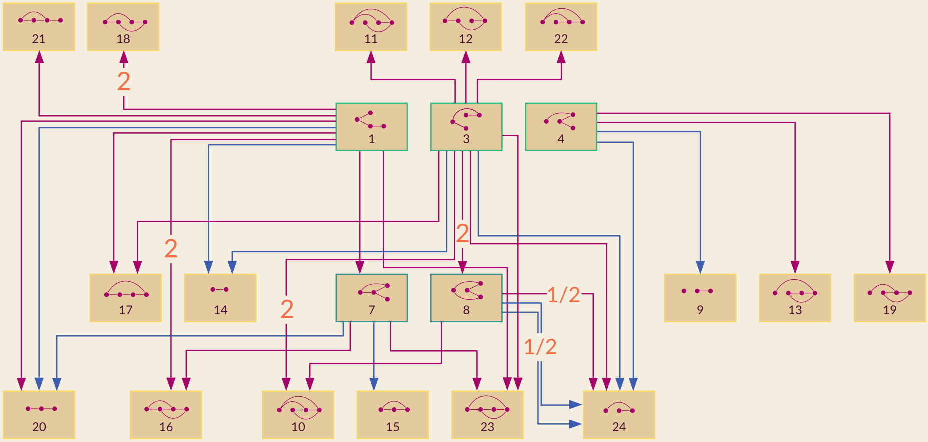
Counting P_4s using wcol_3

Lemma 6. *Let* $\mathbf{H} \in \mathcal{H}$ *be a (non-linear) pattern relaxation and let* $\mathbf{H}_1 \oplus_{\bar{x}} \mathbf{H}_2 = \mathbf{H}$. *Fix an ordered vertex set* $\bar{y} \in \mathbb{G}$ *such that* $\mathbf{H}[\bar{x}] \simeq \mathbb{G}[\bar{y}]$. *Then*

$$\underset{\bar{x}\mapsto\bar{y}}{\#}(\mathbf{H},\mathbb{G}) = \underset{\bar{x}\mapsto\bar{y}}{\#}(\mathbf{H}_1,\mathbb{G})\underset{\bar{x}\mapsto\bar{y}}{\#}(\mathbf{H}_2,\mathbb{G}) - \sum_{\mathbf{D}\in\mathcal{D}(\mathbf{H}_1,\mathbf{H}_2)} \frac{\underset{\bar{x}\mapsto\bar{x}}{\#}(\mathbf{H},\mathbf{D}\mid\mathbf{H}_1,\mathbf{H}_2)\underset{\bar{x}\mapsto\bar{y}}{\#}(\mathbf{D},\mathbb{G})}{\underset{\bar{x}\mapsto\bar{x}}{\#}(\mathbf{D},\mathbf{D})}.$$

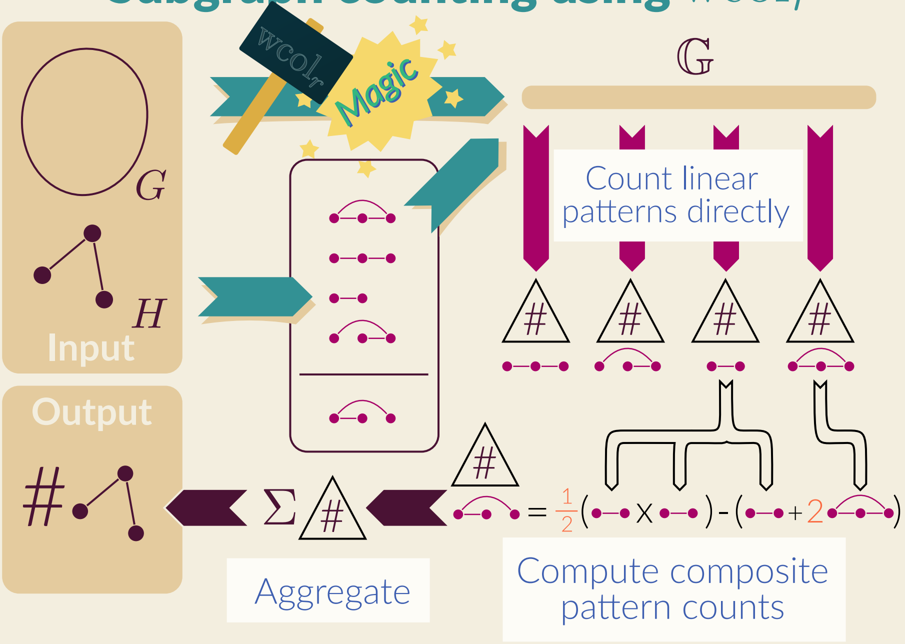


Counting P_4s using wcol_3



This 'counting-DAG' has bounded depth

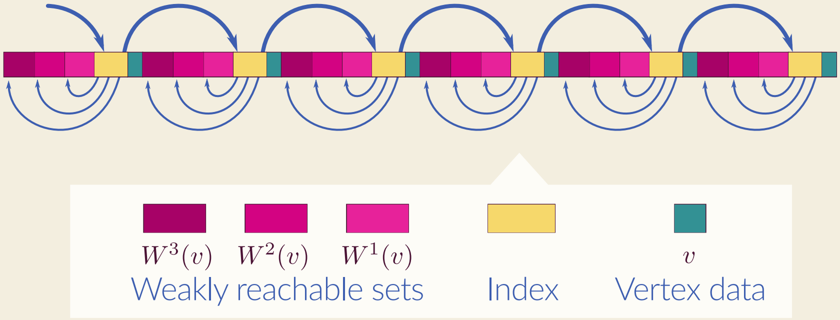
Subgraph counting using wcol_r



Engineering: memory

Lesson: We cannot ignore memory locality.
But this is really hard when working with graphs.

① Flatten everything



Engineering: memory

Lesson: We cannot ignore memory locality.
But this is really hard when working with graphs.

2 Be as specific as possible

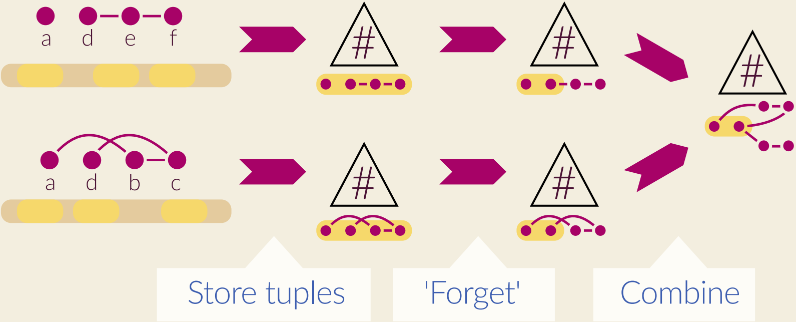
Vertex ids	(int16)	int32	int64
n	$\approx 60\text{K}$	$\approx 4.3\text{Mio}$	All we could ever need
Node ids	3 bit	4 bit	char
k			All we could ever need

Tough design choices

Engineering: data structures

Lesson: Common data structures not enough

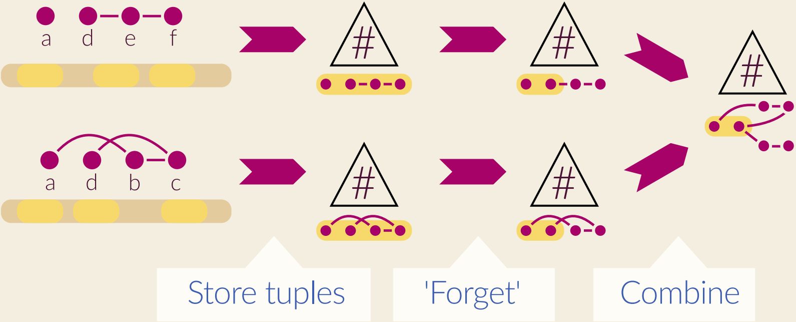
3 Design special-purpose data structures



Engineering: data structures

Lesson: Common data structures not enough

3 Design special-purpose data structures



Our solution: specialized trie. Better options?

Next steps

1 More features

Count coloured graphs, sum-of-weights, other types of embeddings (homomorphisms, non-induced, etc.)

2 Implement for $\mathbf{col}_r(\mathbb{G})$

We know how to do this in theory and $\mathbf{col}_r(\mathbb{G})$ seems to be smaller in practice.

? Preprocessing!

Simple preprocessing rules (based e.g. min-degree of pattern) help a lot. Can we generate more elaborate, pattern-dependent rules?

Solvers vs Solutions

Part III

Future work

The big open question

THANKS

Exhibit X

Local centrality measures via neighbourhood surveys

(Secret slides)

Close-to-Closeness Centralities

	$C(v)$	r-Local version
Closeness	$\left(\sum_{u \in G} \mathrm{dist}(u,v)\right)^{-1}$	$\left(\sum_{u \in N^r[v]} \mathrm{dist}(v,u)\right)^{-1}$
Harmonic	$\sum_{u \in G} \mathrm{dist}(u,v)^{-1}$	$\sum_{u \in N^r[v]} \mathrm{dist}(v,u)^{-1}$
Lin's index	$\dfrac{\lvert\{u \mid \mathrm{dist}(u,v) < \infty\}\rvert^2}{\sum_{\mathrm{dist}(u,v)<\infty} \mathrm{dist}(u,v)}$	$\dfrac{\lvert N^r[v]\rvert^2}{\sum_{u \in N^r[v]} \mathrm{dist}(v,u)}$

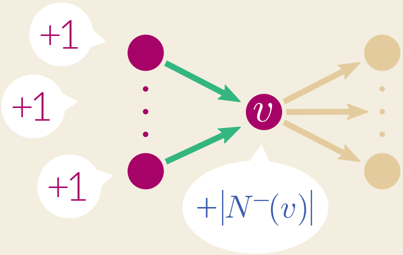
All three measures can be computed quickly if we know $|N^d(v)|$ for $1 \leqslant d \leqslant r$.

Can we compute this quickly in sparse graphs?

Warm-up: Counting with degeneracy

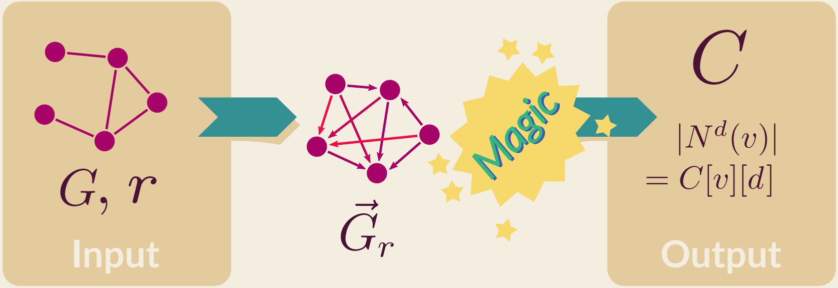
Let G be (d-1)-degenerate.

1. Compute orientation $\vec{G}$ with $\Delta^{-}(\vec{G}) \leqslant d$ in linear time.
2. Initialize counter $C[v] = 0$ for all $v \in G$.
3. For every $v \in G$, increment $C[v]$ *and* $C[u]$ for every in-neighbour $u \in N^{-}(v)$.



Degeneracy to dtf-augmentations

Thm. Given a graph G and an integer r, we can compute the size of $|N^d(v)|$ for all $v \in G$ and $1 \leqslant d \leqslant r$ in total time $O(2^{\Delta^-(\vec{G}_r)}n)$.



Counting using dtf-augmentations

We compute the size of the rth nbhds:

1. Compute dtf-augm. $\vec{G}_r$ with small $\Delta^-(\vec{G}_r)$ in linear time.

Counting using dtf-augmentations

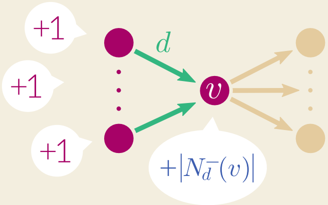
We compute the size of the r[th] nbhds:

1. Compute dtf-augm. $\vec{G}_r$ with small $\Delta^-(\vec{G}_r)$ in linear time.
2. Initialize counter $C[v][d] = 0$ for all $v \in G$ and $d \leqslant r$.

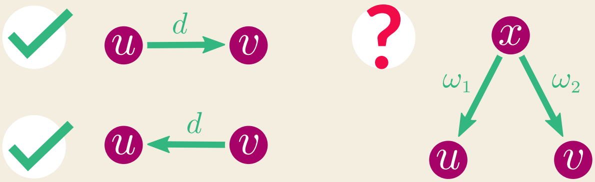
Counting using dtf-augmentations

We compute the size of the r[th] nbhds:

1. Compute dtf-augm. $\vec{G}_r$ with small $\Delta^-(\vec{G}_r)$ in linear time.
2. Initialize counter $C[v][d] = 0$ for all $v \in G$ and $d \leqslant r$.
3. For every $v \in G$, increment $C[v][d]$ *and* $C[u][d]$ for every in-neighbour $u \in N_d^-(v)$.



Counting using dtf-augmentations



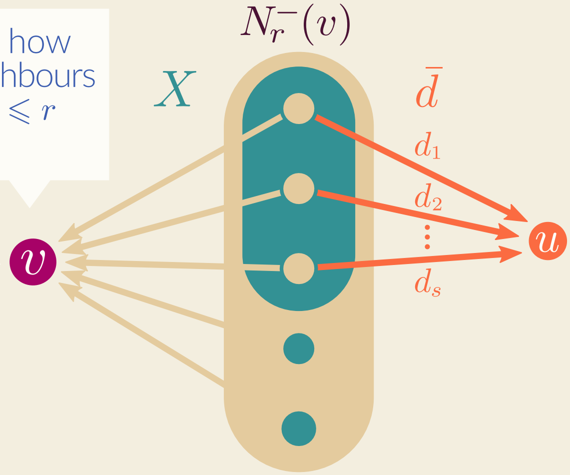
The counting so far takes care of the first two cases, but what about the *indirect* neighbours?

This is where the algorithm becomes **interesting**.

Counting using dtf-augmentations

v needs to know how many indirect neighbours at distance $2 \leqslant d \leqslant r$ there are.

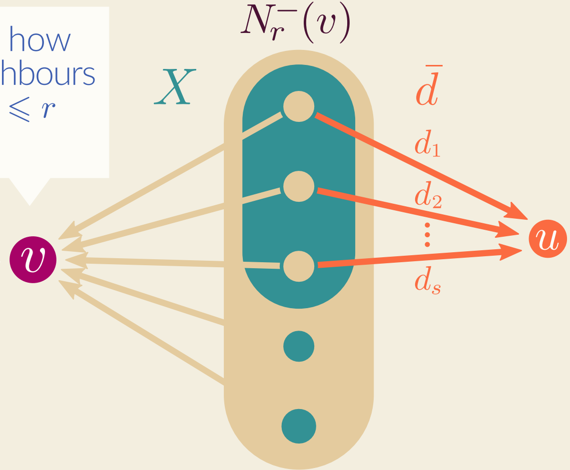
Indirect neighbours connect via $N_r^-(v)$.



Counting using dtf-augmentations

v needs to know how many indirect neighbours at distance $2 \leqslant d \leqslant r$ there are.

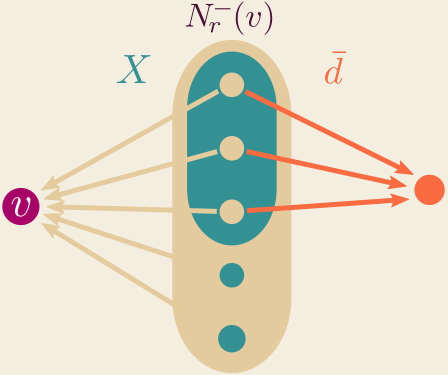
Indirect neighbours connect via $N_r^-(v)$.



We compute the distance between v, u as follows:

$$\mathrm{dist}(u,v) = \min\big(\mathrm{dist}(v,X) + \mathrm{dist}(u,X)\big)$$

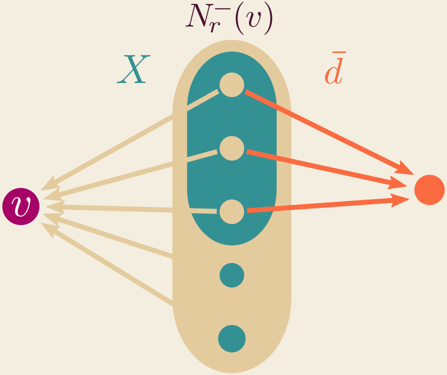
Counting using dtf-augmentations



We need to compute for every set $X \subseteq N_r^-(v)$ and every possible dist.-vector $\bar d \in [r]^{|X|}$ the number of vertices u such that:

1. $N_r^-(u) \cap N_r^-(v) = X$
2. $\mathrm{dist}(u, X) = \bar d$

Counting using dtf-augmentations



We need to compute for every set $X \subseteq N_r^-(v)$ and every possible dist.-vector $\bar{d} \in [r]^{|X|}$ the number of vertices u such that:

1. $N_r^-(u) \cap N_r^-(v) = X$
2. $\mathrm{dist}(u, X) = \bar{d}$

Let us call this number $c(v, X, \bar{d})$. Our first goal is to compute it for every vertex.

A data structure for $c(v, X, \bar{d})$

1. For every $v \in \vec{G}_r$, $X \subseteq N_r^-(v)$ and $\bar{d} \in [r]^{|X|}$, initialize $R[X][\bar{d}] = 0$.

A data structure for $c(v, X, \bar{d})$

1. For every $v \in \vec{G}_r$, $X \subseteq N_r^-(v)$ and $\bar{d} \in [r]^{|X|}$, initialize $R[X][\bar{d}] = 0$.

2. For every $v \in \vec{G}_r$, $X \subseteq N_r^-(v)$, increment

$$R[X][\mathrm{dist}(v, X)]$$

by one.

A data structure for $c(v, X, \bar{d})$

1. For every $v \in \vec{G}_r$, $X \subseteq N_r^-(v)$ and $\bar{d} \in [r]^{|X|}$, initialize $R[X][\bar{d}] = 0$.
2. For every $v \in \vec{G}_r$, $X \subseteq N_r^-(v)$, increment
$$R[X][\mathrm{dist}(v, X)]$$
by one.

Claim.

$$c\,(v, X, \bar{d}) = \sum_{X \subseteq Y \subseteq N_r^-(v)} (-1)^{|Y \setminus X|} \sum_{\bar{d}' : \bar{d}'|_X = \bar{d}} R[Y][\bar{d}'].$$

Counting using dtf-augmentations

Given $c(v, \bullet, \bullet)$ we can now count the number of indirect neighbours of v. For every subset $X \subseteq N_r^-(v)$ and distance-vector $\bar{d} \in [r]^{|X|}$, apply the update:

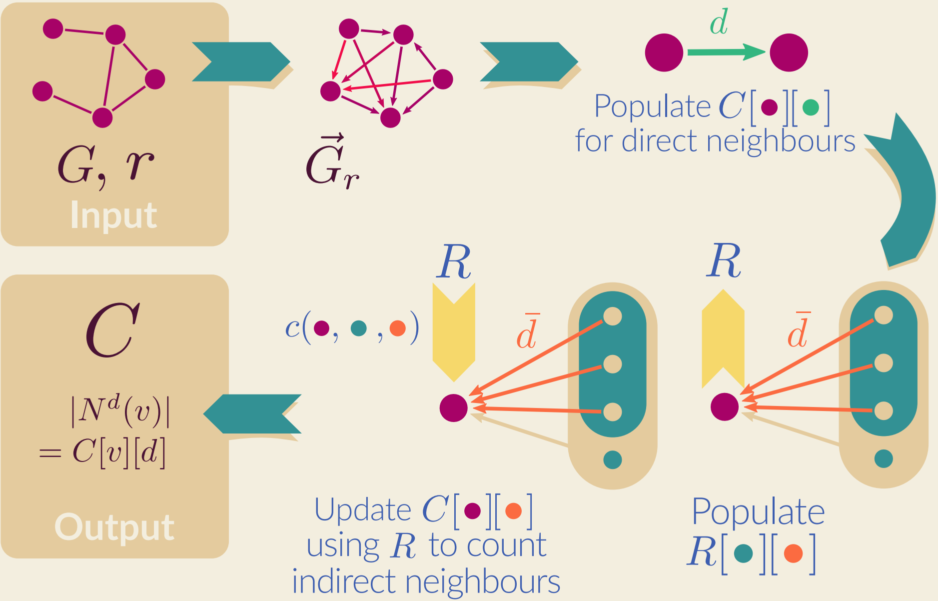
$$C[v][\min(\bar{d} + \operatorname{dist}(v, X))] \mathrel{+}= c(v, X, \bar{d})$$

Since the above counts v as a neighbour of itself, we apply the following correction:

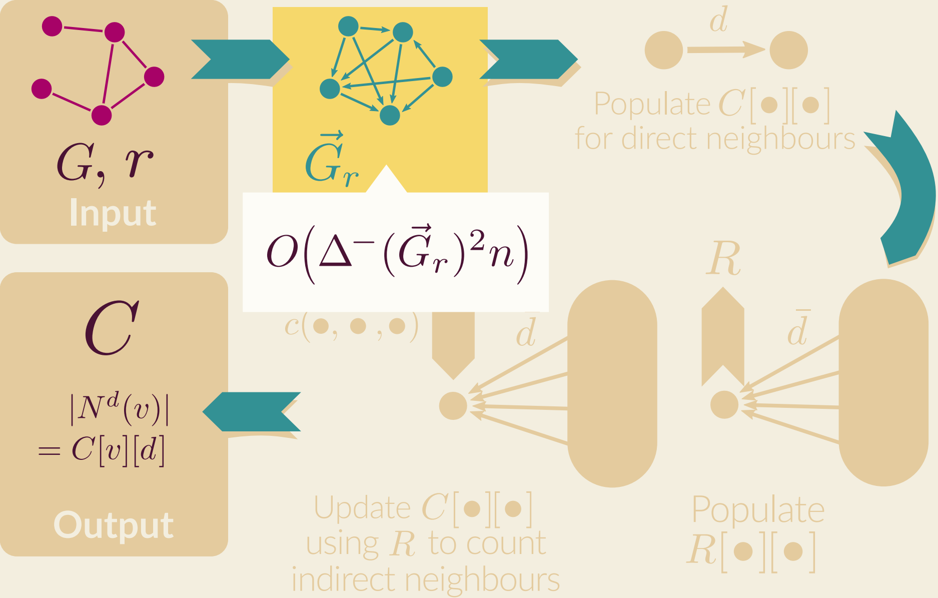
$$C[v][\min(\operatorname{dist}(v, X) + \operatorname{dist}(v, X))] \mathrel{-}= 1$$

There are a few more corrections concerning direct neighbours, see paper.

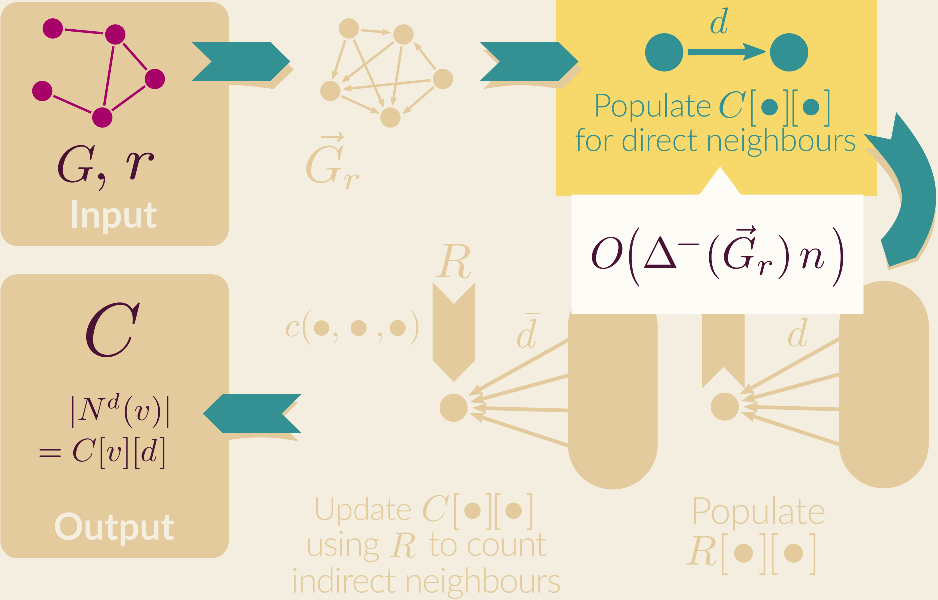
Counting using dtf-augmentations



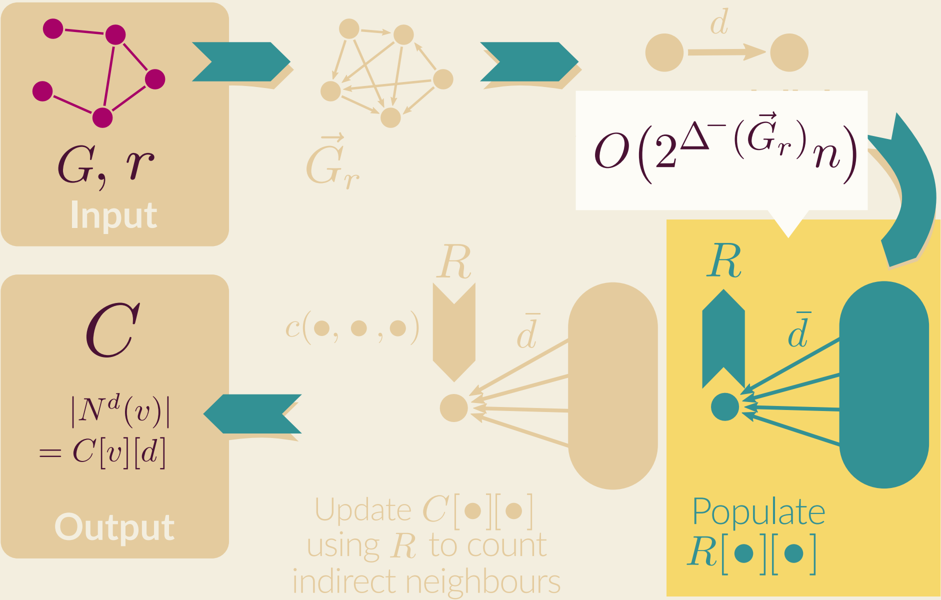
Counting using dtf-augmentations



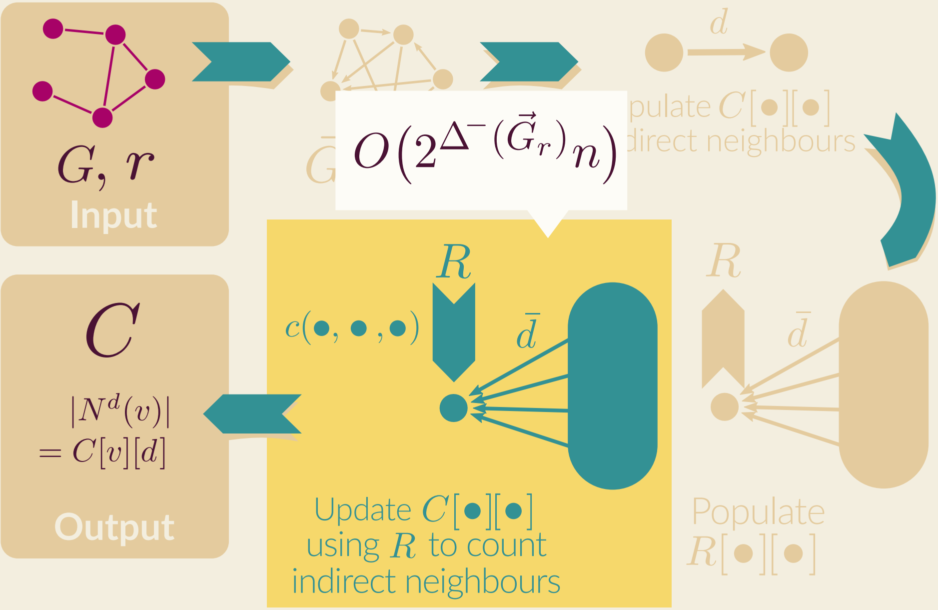
Counting using dtf-augmentations



Counting using dtf-augmentations



Counting using dtf-augmentations



Counting using dtf-augmentations

Thm. Given a graph G and an integer r, we can compute the size of $|N^d(v)|$ for all $v \in G$ and $1 \leqslant d \leqslant r$ in total time $O(2^{\Delta^-(\vec{G}_r)} n)$.

- Exponential vs quadratic?
- Does not scale to nowhere dense graphs!

$2^{n^\varepsilon} n$

Can we do ***better***?

Can we do better?

Closed 2-Neighbourhood Sizes

Input: A graph G.

Output: $|N^2[v]|$ for every $v \in G$.

Thm. Unless SETH fails, 2-CNBS cannot be solved in time

1. $O(|G|^{2-\varepsilon})$
2. $O(2^{o(\Delta^-(\vec{G}_2))} n^{2-\varepsilon})$

Gutin G, Mertzios GB, Reidl F.
Lower and Upper Bound for Computing the Size of All Second Neighbourhoods.
arXiv preprint arXiv:1805.01684. 2018 May 4

Engineering: compromises

Lesson: We cannot be *too* idealistic.
Some vertices will have large in-neighbourhoods.

1. Mix algorithms

Conduct regular bfs for vertices with large in-neighbourhood, use dtf-magic for everything else.

Luckily, these two approaches mix!

Next steps

1 More features

We can easily count the weights of neighbourhoods or neighbourhoods restricted to certain colours.

2 Optimize for small r

Most real-world cases will be for small distances. In particular for r=2 we can simplify the algorithm.

? Nowhere dense?

Combine with neighbourhood complexity, handle large intersections differently.

? Approximate?

The Möbius inversion is the bottleneck. Sacrifice precision for speed?