CIFellows 2020-2021

Computing Innovation Fellows

Putting Parameterization into Practice

Shweta Jain

University of Utah

It's NP-Hard! What now?

- Many problems are NP-Hard
- **Fixed Parameter Tractability (FPT)**: Some NP-Hard problems can be solved in $f(k)n^{O(1)}$ for parameter k
  - Example: Vertex Cover parameterized by solution size (k) can be solved in $O(n2^k)$

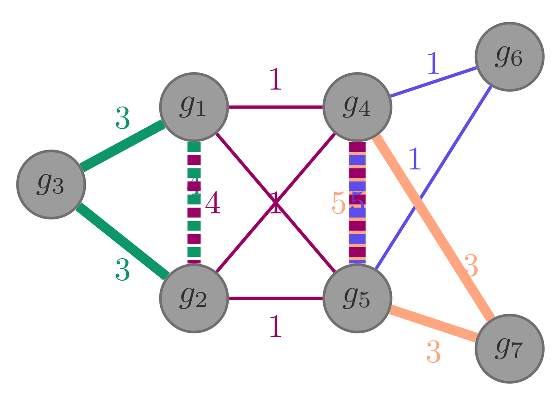
$$2^n \quad \text{vs} \quad n2^k$$

Which is better in practice?

- If k is small, problem may be tractable in practice even for large graphs
- Can we recognize good parameters for applied problems and give FPT algorithms for them?

Genes that act together

- Interaction between proteins and genes represented as graphs
- Edge weights represent strength of pairwise correlation
- Knowing which groups (modules) of genes consistently co-act is critical in understanding **disease mechanisms** and in developing **new therapies**.
- Gene module discovery modeled as *Edge Weighted Clique Decomposition (EWCD)*:



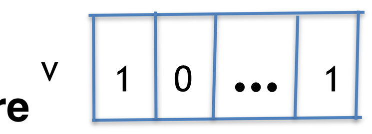
Gene-gene interaction network with edges weighted by the sum of the strengths of all modules that contain both end-points (indicated by color coding).

> ***Edge Weighted Clique Decomposition:*** *Given a graph G with positive edge weights, is there a decomposition of G into at most k weighted cliques such that the weight of each edge = sum of weights of cliques that the edge participates in?*

Story so far

Prior work by Cooley et al., 2020

- **cricca**: uses kernelization to obtain a **smaller**, **equivalent** graph G'
- Runs a clique decomposition algorithm on G'. Determines a **signature** (binary vector of length k) for each vertex.
- Converts a clique decomposition of G' into a clique decomposition of G

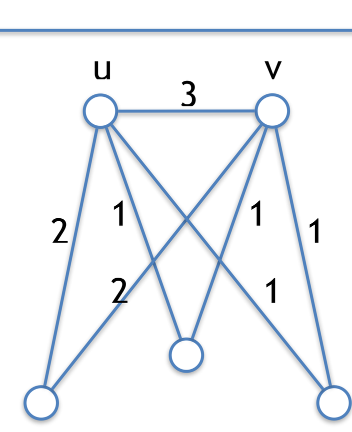


i^{th} bit represents whether the vertex participates in the i^{th} clique

Kernelization

- Applies reduction rules to **prune vertices**
- **Guarantees** that G' is a YES instance **iff** G is a YES instance

Curious case of the twins



- Twins: Neighbors with the **same neighborhood**
- Twins form **equivalence** classes
- **Observation**: Vertices in **different classes** have **different signatures**
- **Observation**: At most 2^k **unique** signatures possible

> **Reduction rule 1**: If more than 2^k classes => G is a NO instance

- Two kinds of twins:
  - **Identical** twins: Twins that **must** have **identical signatures**
  - **Fraternal** twins: Twins that **may not** have identical signatures

- **Observation**: If two vertices in a class are identical, then entire class must be identical
- **Observation**: For an **identical** class, suffices to **keep one** vertex and **prune the rest**

> **Reduction rule 2**: Class with more than 2^k vertices must be identical. Keep just one vertex and prune away the rest.

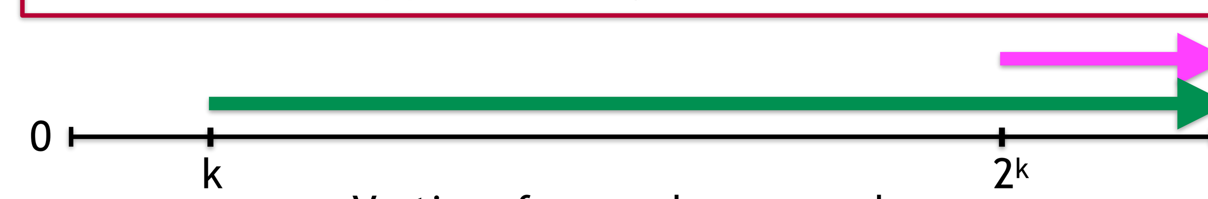
- At most 2^k classes, at most 2^k vertices in each class => **cricca** kernel size 4^k

Some coffee to speed things up

Joint work with Y. Mizutani, B. Sullivan, 2022

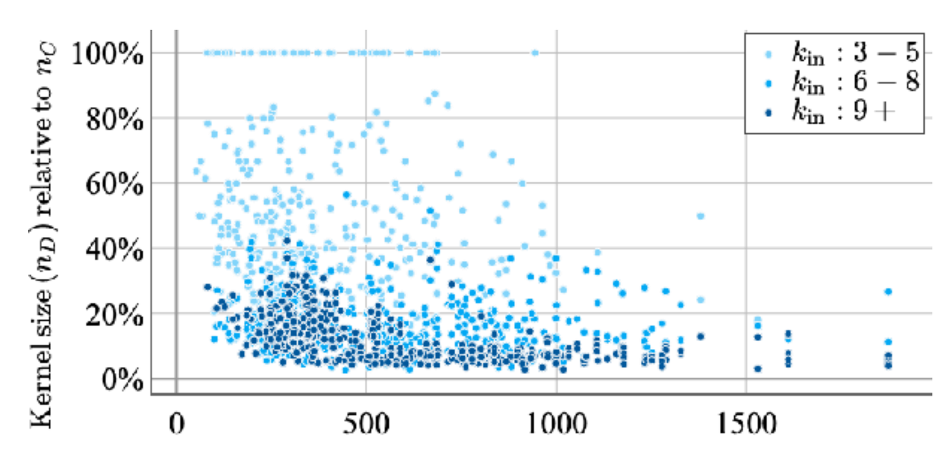
- **Observation**: If G is a YES instance, then there exists a solution in which vertices in **every** class have either unique signatures or identical signatures.
- **Theorem**: Any class larger than k vertices must be **identical**
  - Uses a heavy hammer from combinatorial design theory called *Fisher's inequality*

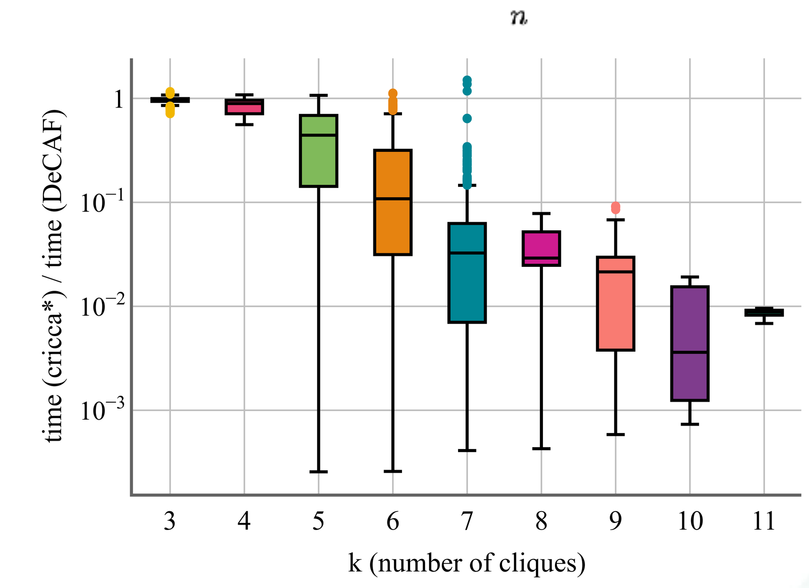
> Reduction rule 2 applies to every class that has more than k vertices



- At most 2^k classes, each with at most k vertices. **DeCAF** kernel size: $k2^k$
- Exponential reduction in kernel size compared to **cricca**!
- Smaller kernel + smarter search space exploration = **DeCAF**

Small step for kernel size, giant leap for running time!





- n_C: number of vertices in kernel obtained using **cricca**
- n_D: number of vertices in kernel obtained using **DeCAF**
- n: number of vertices in G
- **80% reduction** in kernel **size**
- Larger reduction for larger k
- Downstream decomposition algorithm gets a smaller graph
- **Exponential reduction** in **running time** of decomposition algorithm

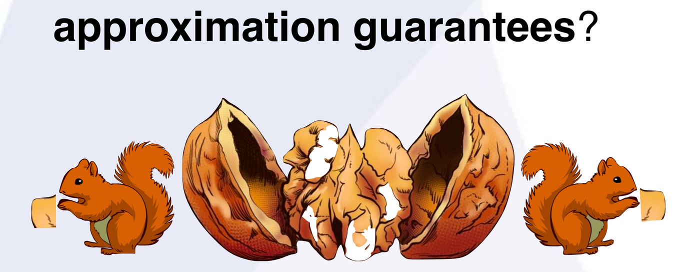
What is that noise?

Optimization version of EWCD:

- Graphs often **noisy** (especially biological graphs)
- Clique weights may not add up exactly to the edge weights
- Can we efficiently find a decomposition such that the **discrepancy** is minimized?

Lossy kernels

- Often, we do not care about exact solution. **Approximation** is enough.
- Some problems do not admit an exact kernel
- What if we allow **controlled noise** in the kernel for such problems to give a kernel with **approximation guarantees**?



CRA Computing Research Association

CCC Computing Community Consortium Catalyst

NSF