Cluster Editing on Cographs and Related Classes

Manuel Lafond **Université de Sherbrooke**

Alitzel Lopez-Sanchez Université de Sherbrooke

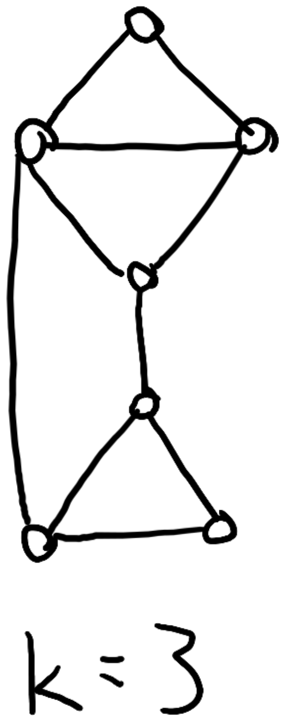
Weidong Luo Université de Sherbrooke

Cluster Editing

Input: a graph G, integer k

Goal: insert/delete $\leq k$ edges to obtain a cluster graph

(i.e., each connected component must be a clique.)

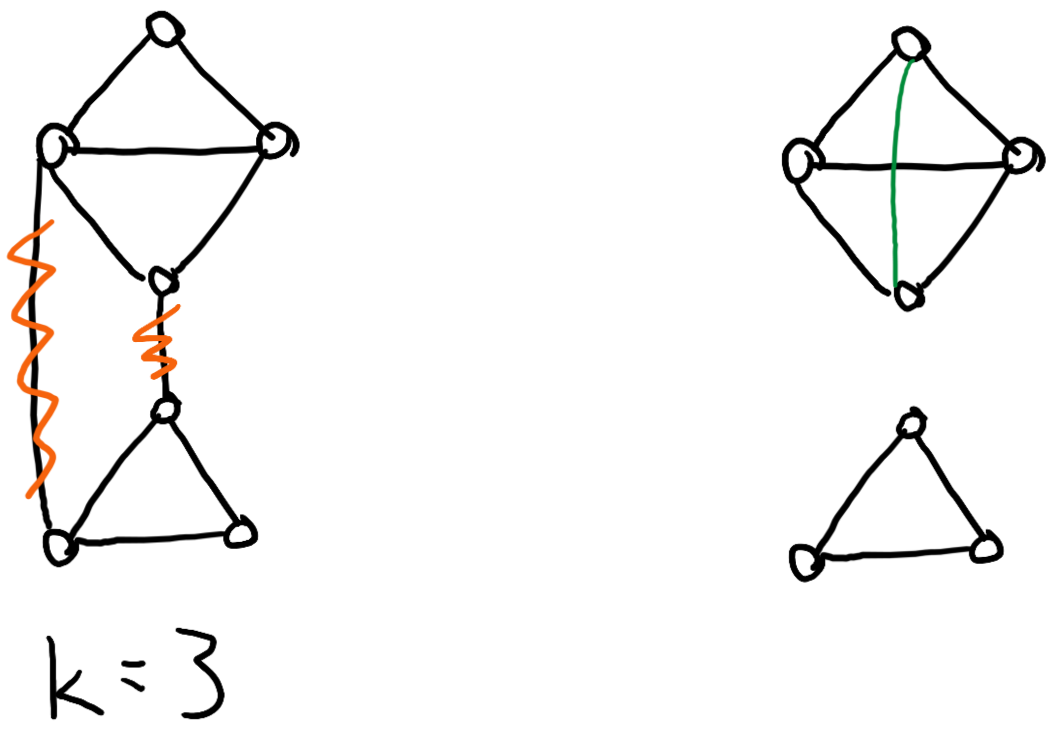


Cluster Editing

Input: a graph G, integer k

Goal: insert/delete $\leq k$ edges to obtain a cluster graph

(i.e., each connected component must be a clique.)



Cluster Editing

Input: a graph G, integer k

Goal: insert/delete $\leq k$ edges to obtain a cluster graph
(i.e., each connected component must be a clique.)

<u>Fixed-parameter perspective.</u>

- Straightforward $O^*(3^k)$ time algorithm.
- $O^*(1.618^k)$ time possible [Böcker, 2012]
- Kernel with $2k$ vertices (compressed equivalent instance) [Chen & Meng, 2012].
- FPT in parameter twin-cover [Italiano et al., 2023]

Cluster Editing

Input: a graph G, integer k

Goal: insert/delete $\leq k$ edges to obtain a cluster graph

(i.e., each connected component must be a clique.)

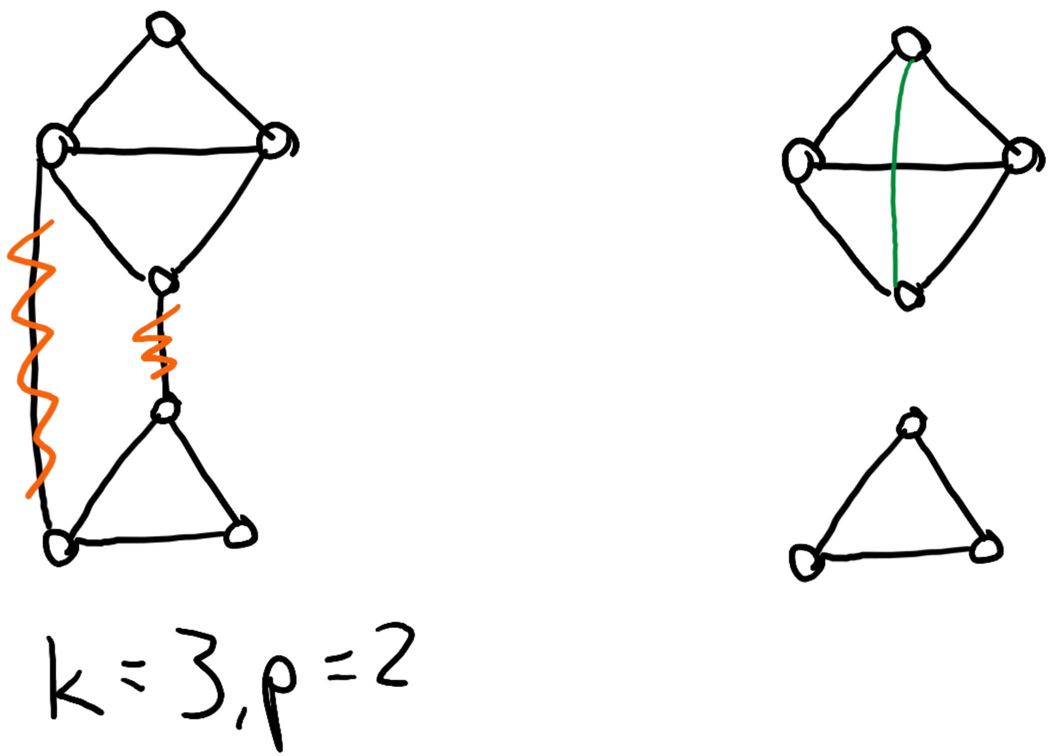
On specific graph classes:

- NP-hard on planar unit disk graphs of max degree 4 [Komusiewicz & Ullman, 2012][Ochs, 2023]
- Polytime on unit interval graphs [Mannaa, 2010]
- **Cluster Deletion** received more attention
  - studied on unit disk graphs, split graphs,...

p-Cluster Editing

Input: a graph G, integers k, p

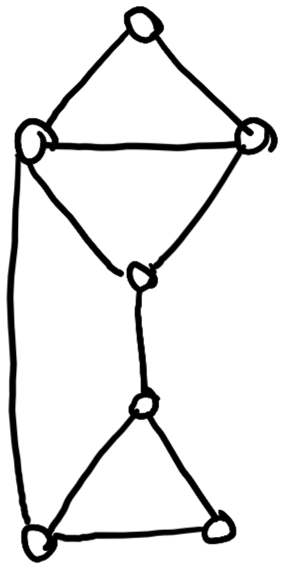
Goal: insert/delete at most k edges to obtain a cluster graph with **exactly** p connected components



p-Cluster Editing

Input: a graph G, integers k, p

Goal: insert/delete at most k edges to obtain a cluster graph with **exactly** p connected components

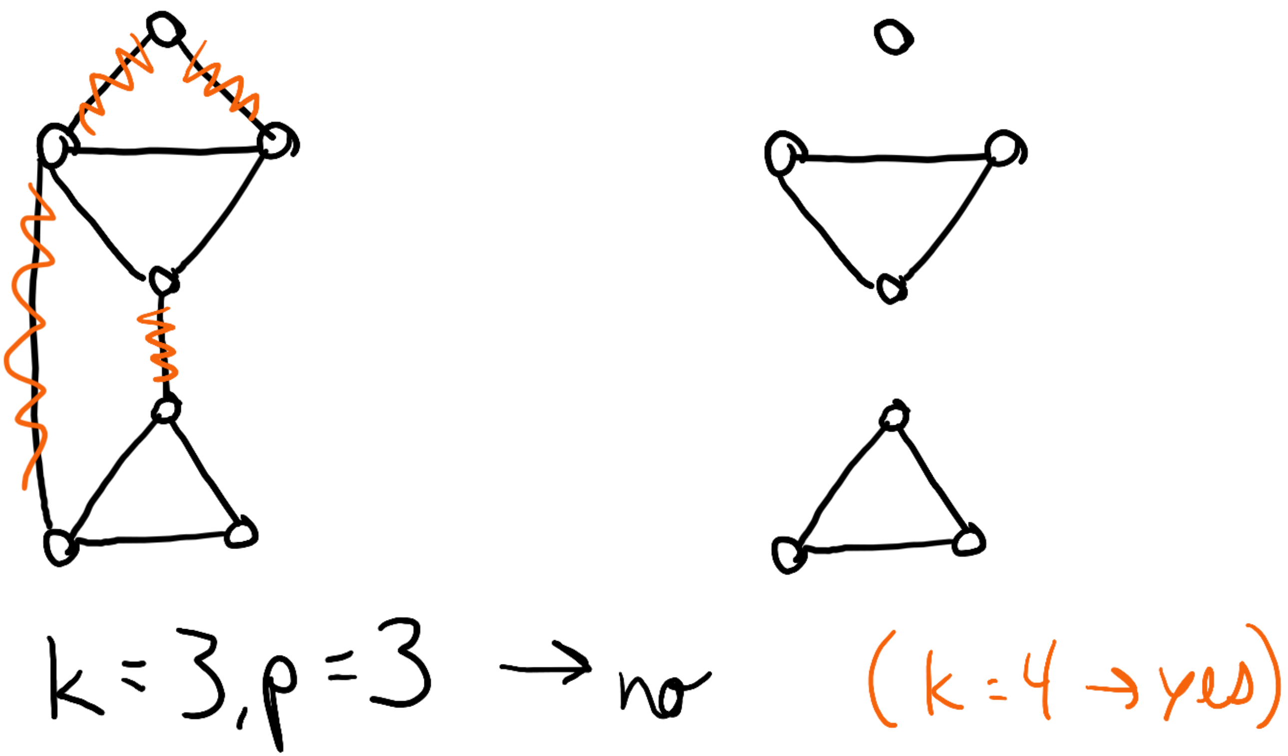


$k=3, p=3 \rightarrow$ no

p-Cluster Editing

Input: a graph G, integers k, p

Goal: insert/delete at most k edges to obtain a cluster graph with **exactly** p connected components



p-Cluster Editing

Input: a graph G, integers k, p

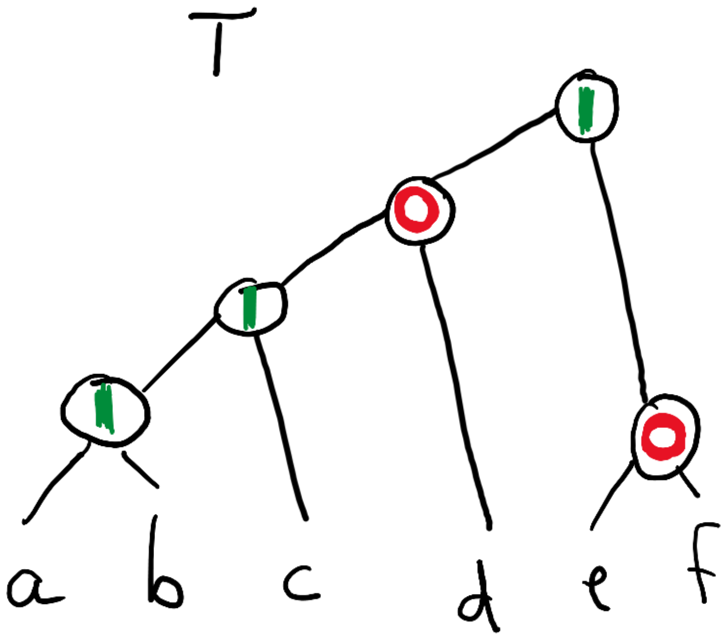
Goal: insert/delete at most k edges to obtain a cluster graph with **exactly** p connected components

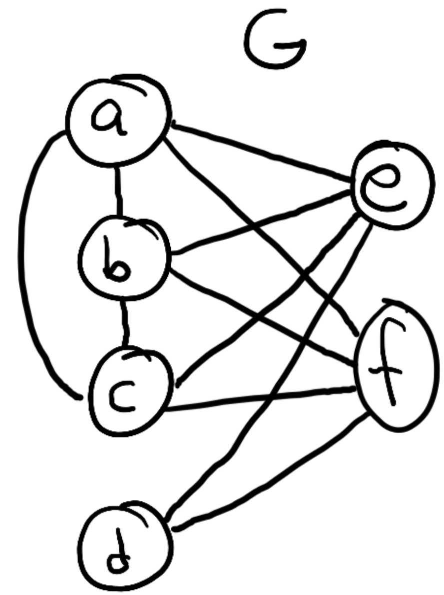
- NP-hard already when $p = 2$ [Shamir et al., 2004]
- Algorithm in time $2^{O(\sqrt{pk})} poly(n)$, tight under Exponential Time Hypothesis (ETH) [Fomin et al,2014]
- Admits a $(p+2)k \ + \ p$ kernel [Guo,2009]

Cluster Editing on Cographs

- Cograph = P_4-free graph
- Cograph = can be built using operations:
  - creating a single vertex
  - taking disjoint union of two cographs
  - taking full join of two cographs
- Cluster Deletion is in P for cographs! [Gao et al., 2013]
  - Take largest clique, make it a cluster, repeat
  - Cluster Insertion is trivially in P.
  - Cluster Editing = OPEN

Cographs and cotrees





- **Why** Cluster Editing on cographs?
  - Distance to a graph class
  - Cographs are “almost” cluster graphs – but how far?
  - Communities usually cluster graphs, but sometimes cographs.
  - Applications in Computational Biology, evolutionary history = cotree = cograph, but people use clustering

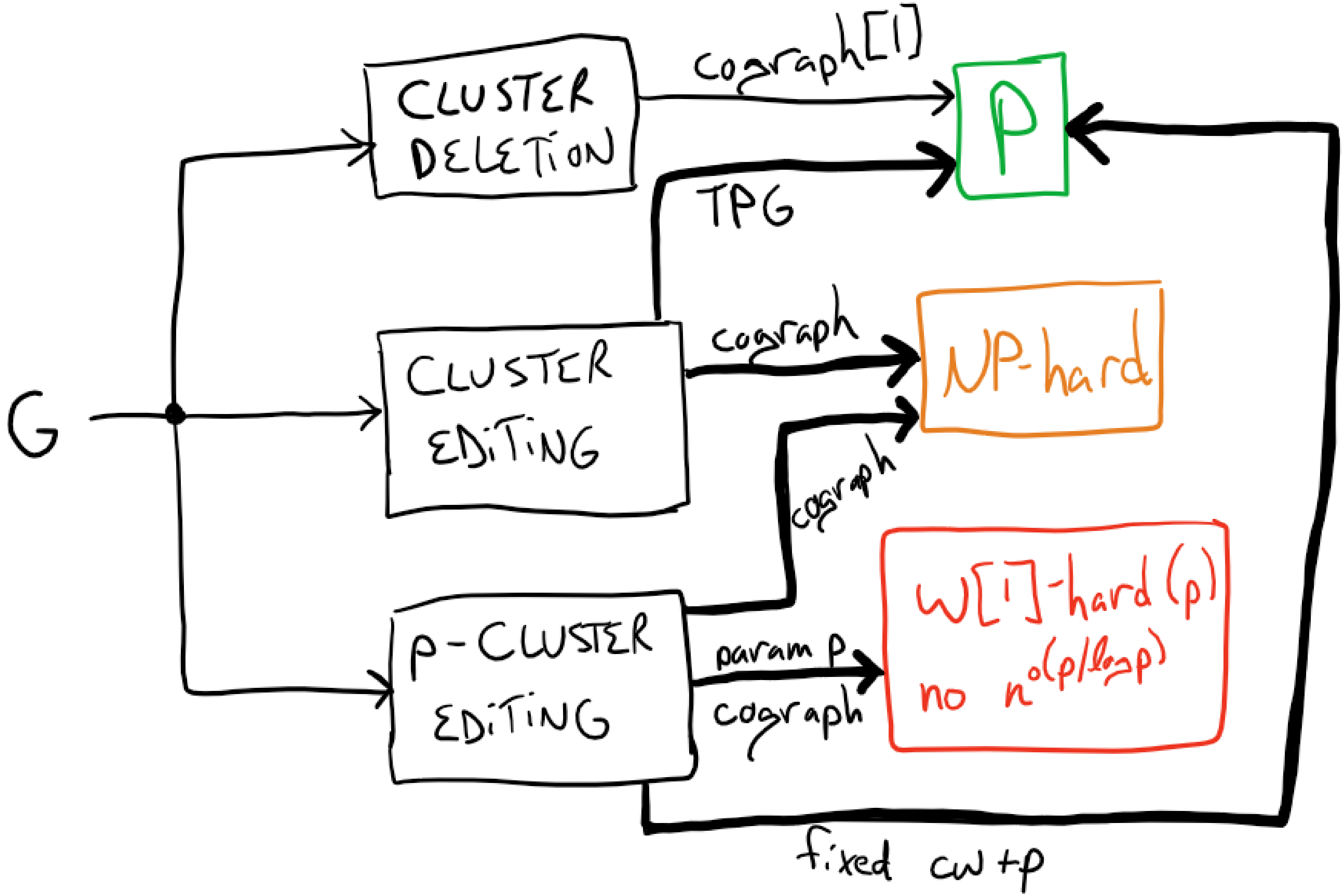
Our results

1. Cluster Editing is NP-complete on cographs.
2. p-Cluster Editing is NP-complete on cographs, and W[1]-hard in parameter p on cographs.

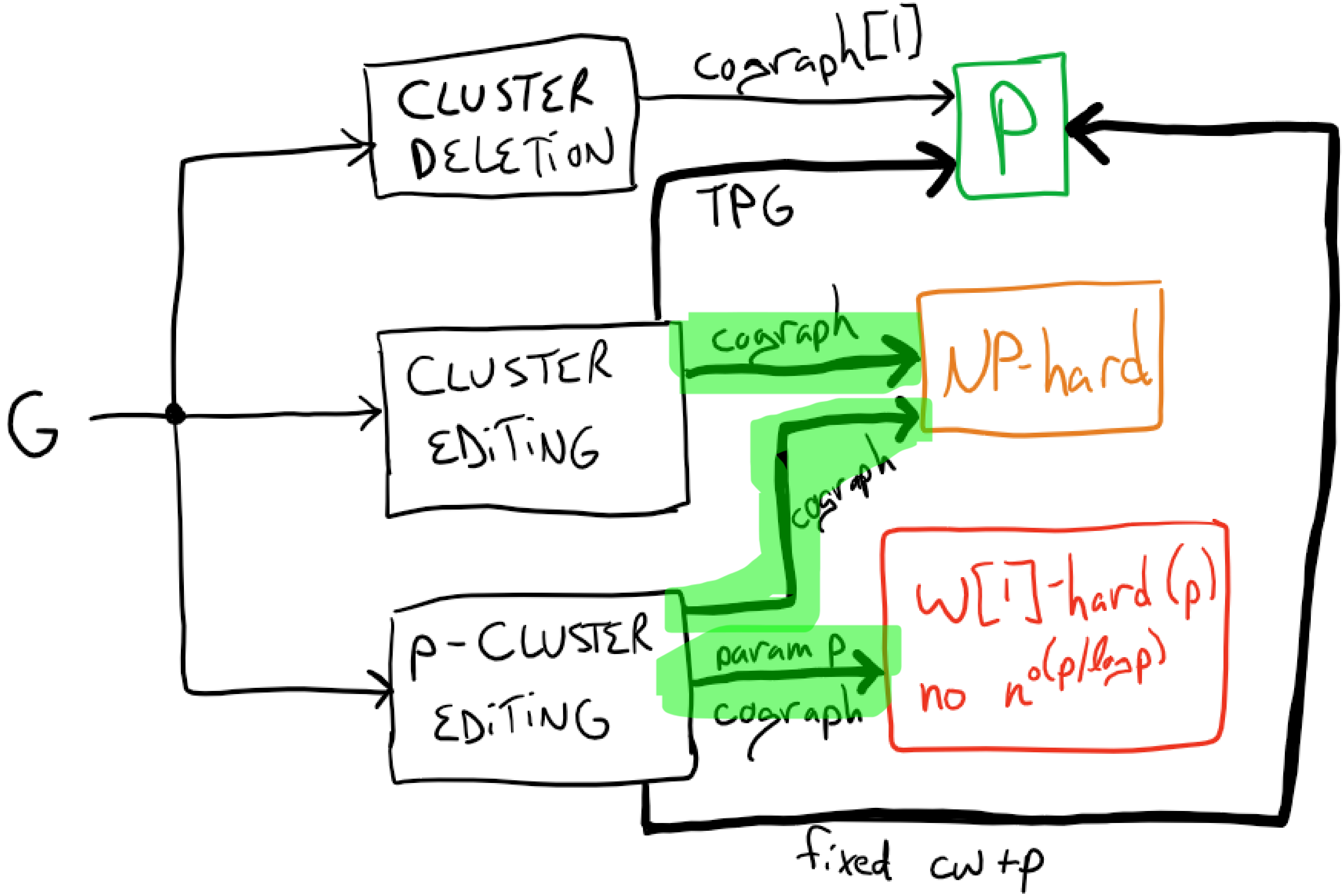
Let cw denote clique-width. Cographs have $cw = 2$.

3. p-Cluster Editing admits a $n^{O(cw\, p)}$ time algorithm. Under the ETH, no $n^{o(cw\, p/\log p)}$ time is possible.
4. For fixed p, p-Cluster Editing on cographs is in P.
5. Cluster Editing is in P on $\{P_4, C_4\}$-free graphs.

   Also known as Trivially Perfect Graphs (TPG)



G
CLUSTER DELETION
CLUSTER EDITING
p-CLUSTER EDITING
cograph[1]
TPG
cograph
cograph
param p
cograph
fixed cw+p
P
NP-hard
W[1]-hard (p)
no n^o(p/log p)

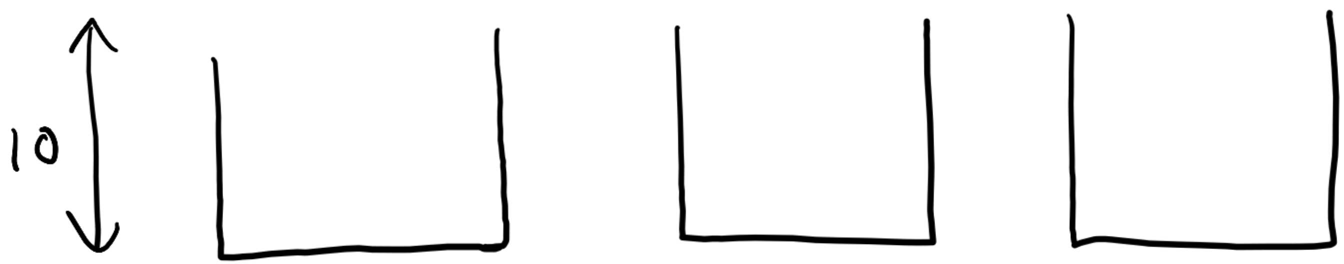

G
CLUSTER DELETION
CLUSTER EDITING
p-CLUSTER EDITING
cograph[1]
P
TPG
cograph
NP-hard
cograph
param p
cograph
W[1]-hard (p)
no n^o(p/log p)
fixed cw+p

Unary Perfect Bin Packing

Input: multiset of unary-encoded integers $A = \{a_1, \ldots, a_n\}$, bin capacity C, bin count p.

Question: can we assign items of A to p bins so that they each sum to exactly C.

$A = \{1, 2, 2, 5, 5, 6, 9\}$ $C = 10$ $p = 3$

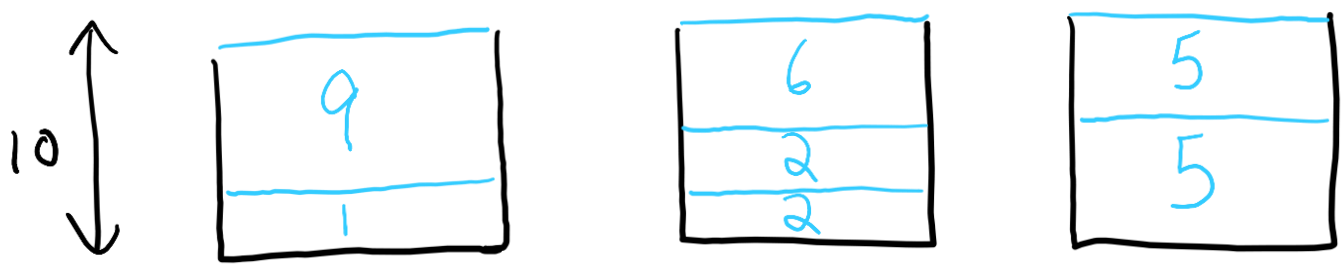


Unary Perfect Bin Packing

Input: multiset of unary-encoded integers $A = \{a_1, \ldots, a_n\}$, bin capacity C, bin count p.

Question: can we assign items of A to p bins so that they each sum to exactly C.

$A = \{1, 2, 2, 5, 5, 6, 9\} \quad C = 10 \quad p = 3$



Unary Perfect Bin Packing

Input: multiset of unary-encoded integers $A = \{a_1, \ldots, a_n\}$, bin capacity C, bin count p.

Question: can we assign items of A to p bins so that they each sum to exactly C.

In [Jansen et al., 2013], the variant where each bin sums to **at most** C is:

(1) NP-hard;

(2) W[1]-hard in parameter p; (probably no $f(p)n^c$ time)

(3) no $n^{o(p/\log p)}$ time algorithm under the ETH.

We show that the same holds for the Perfect variant.

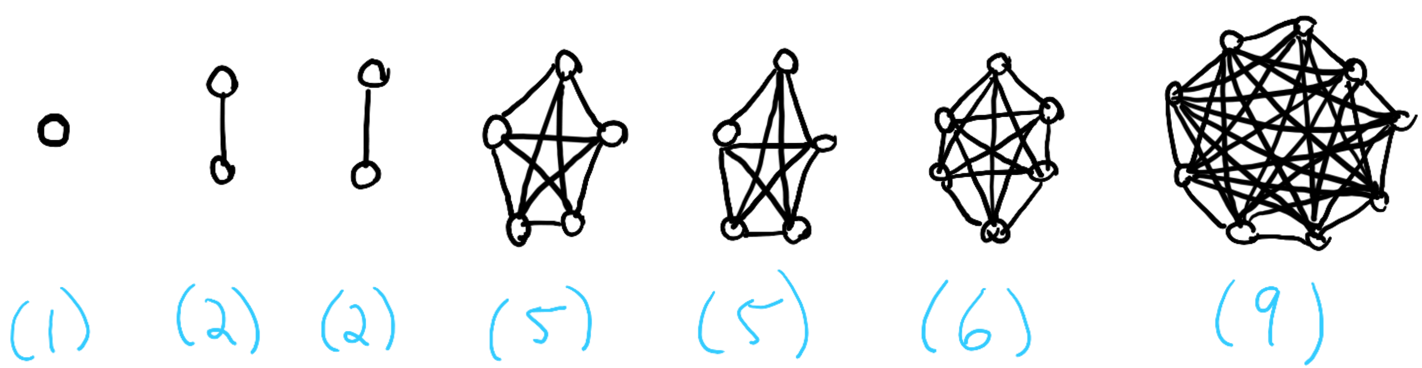
$\mathcal{A} = \{1, 2, 2, 5, 5, 6, 9\}$ $\quad C = 10 \quad p = 3$

Cluster-Editing instance

$\mathcal{A} = \{1, 2, 2, 5, 5, 6, 9\}$ $\quad C = 10 \quad p = 3$

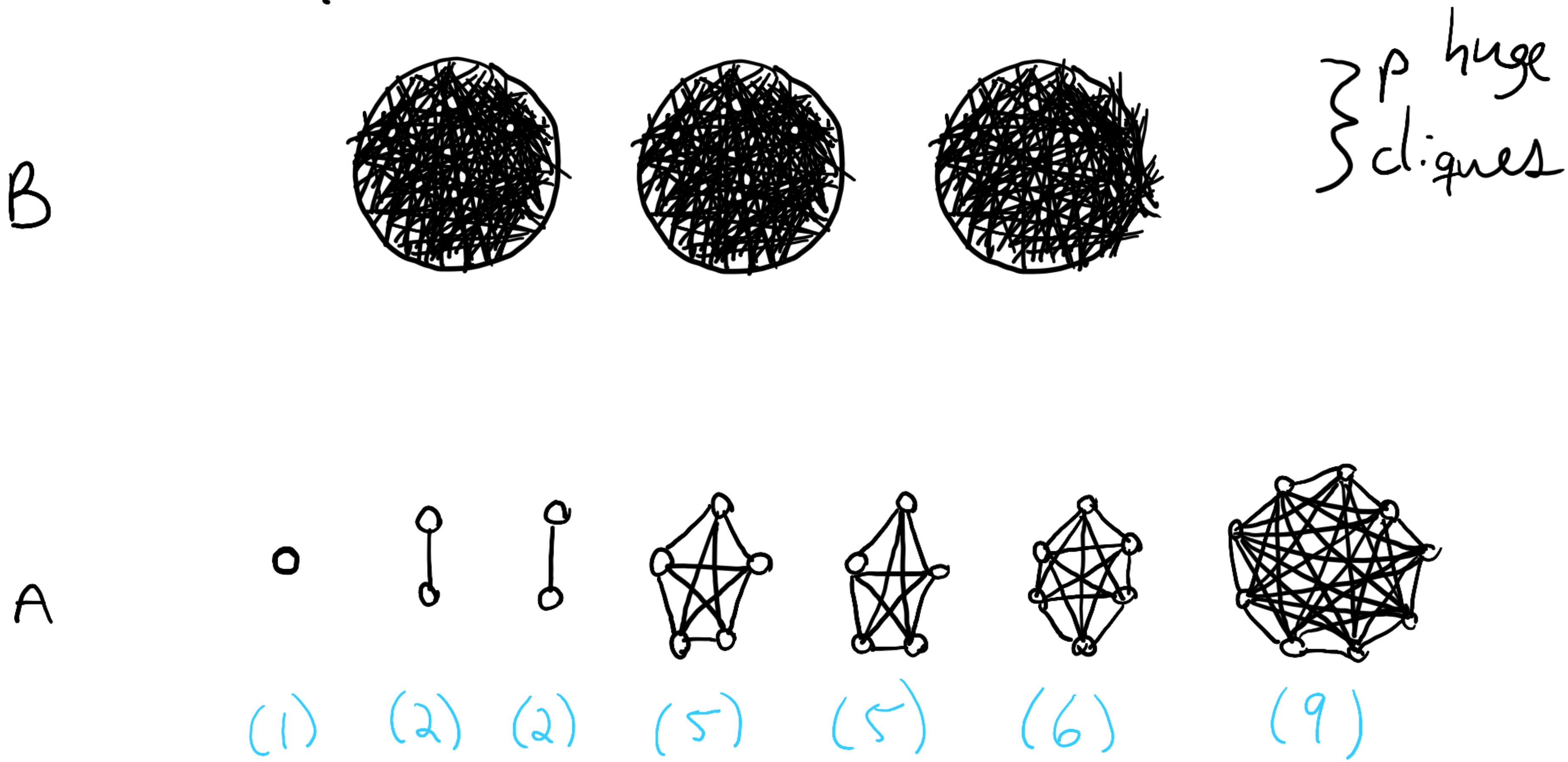
Cluster-Editing instance

A



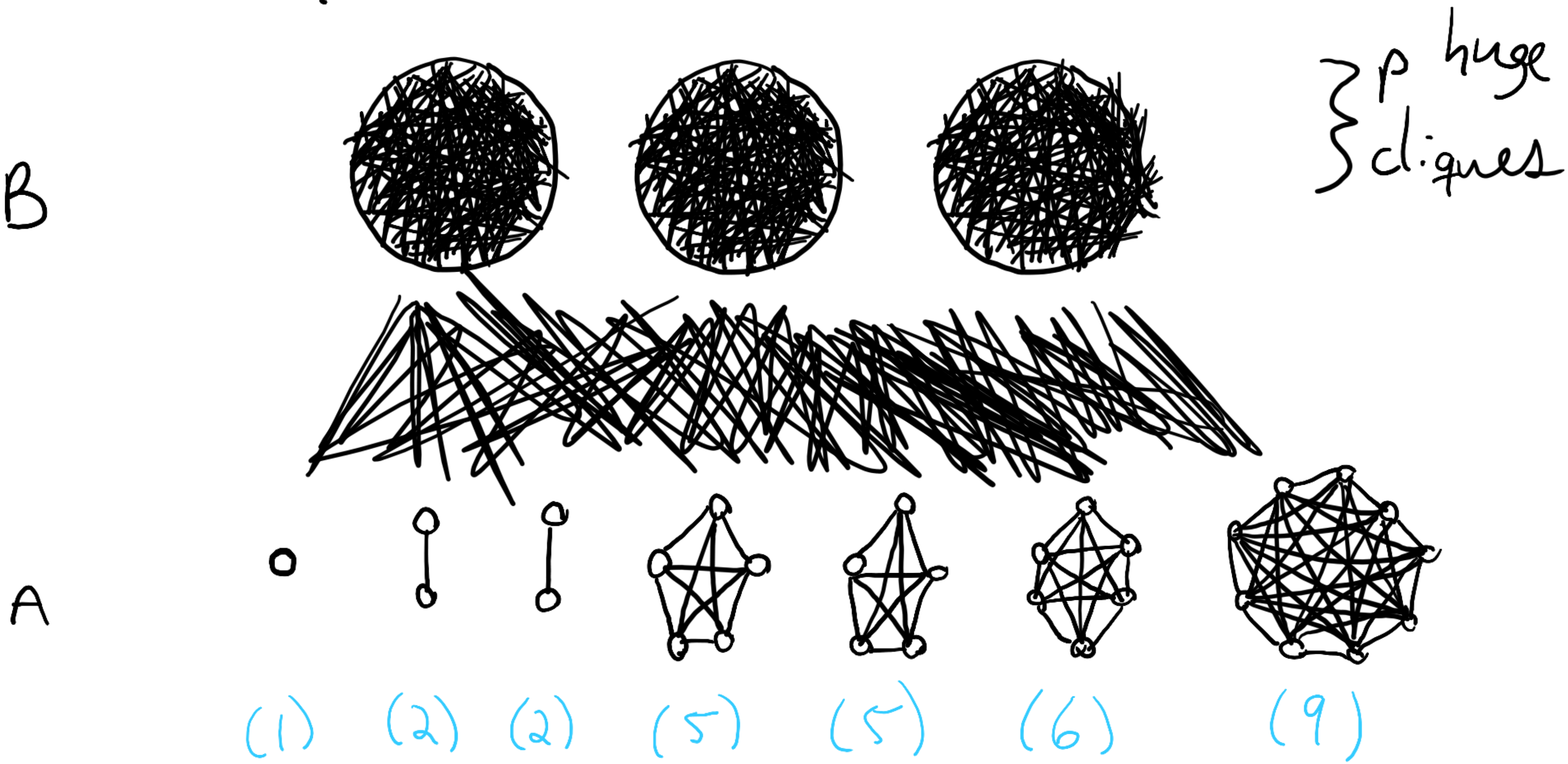
$A = \{1, 2, 2, 5, 5, 6, 9\}$ $C = 10$ $p = 3$

Cluster-Editing instance



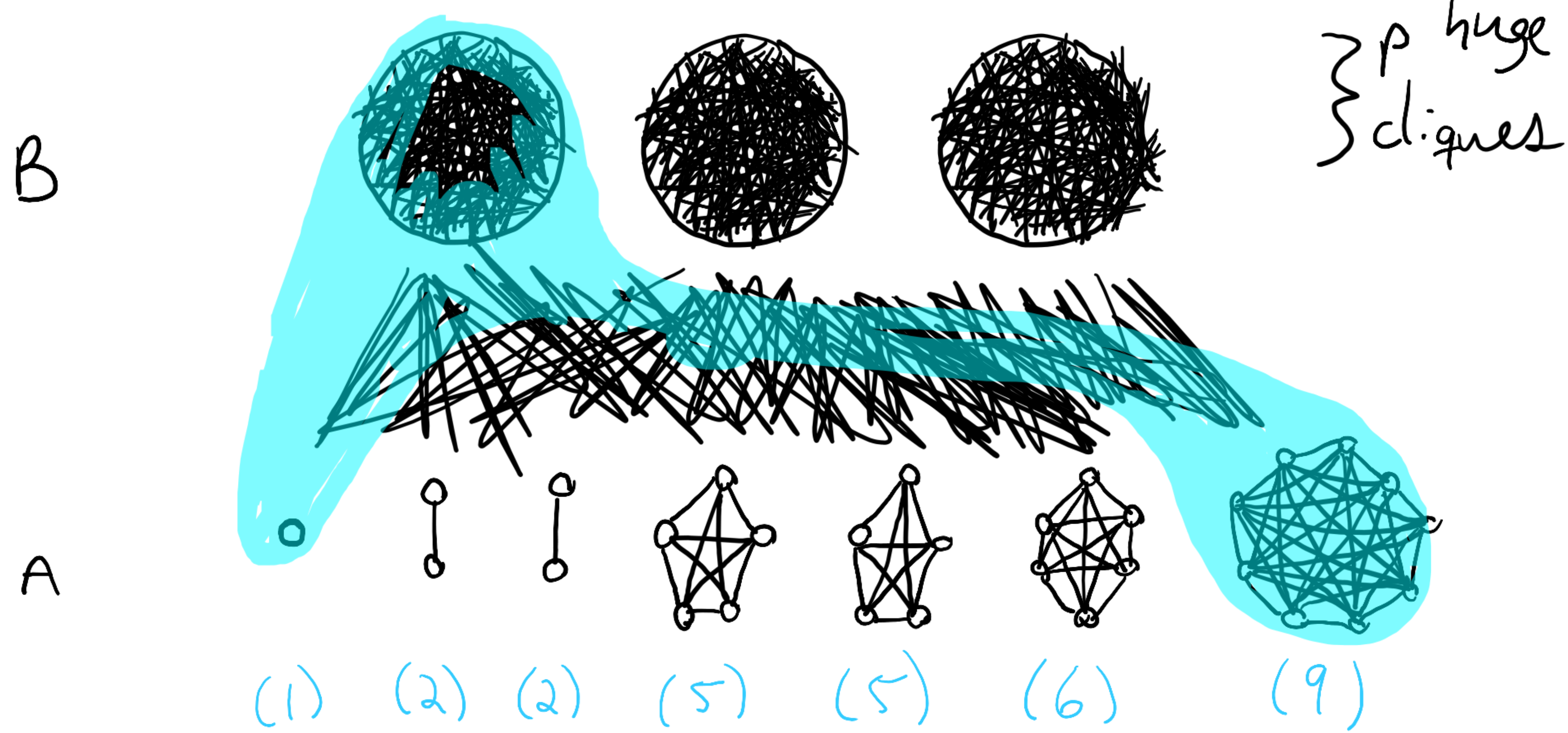
$A = \{1, 2, 2, 5, 5, 6, 9\}$ $\quad C = 10 \quad p = 3$

Cluster-Editing instance



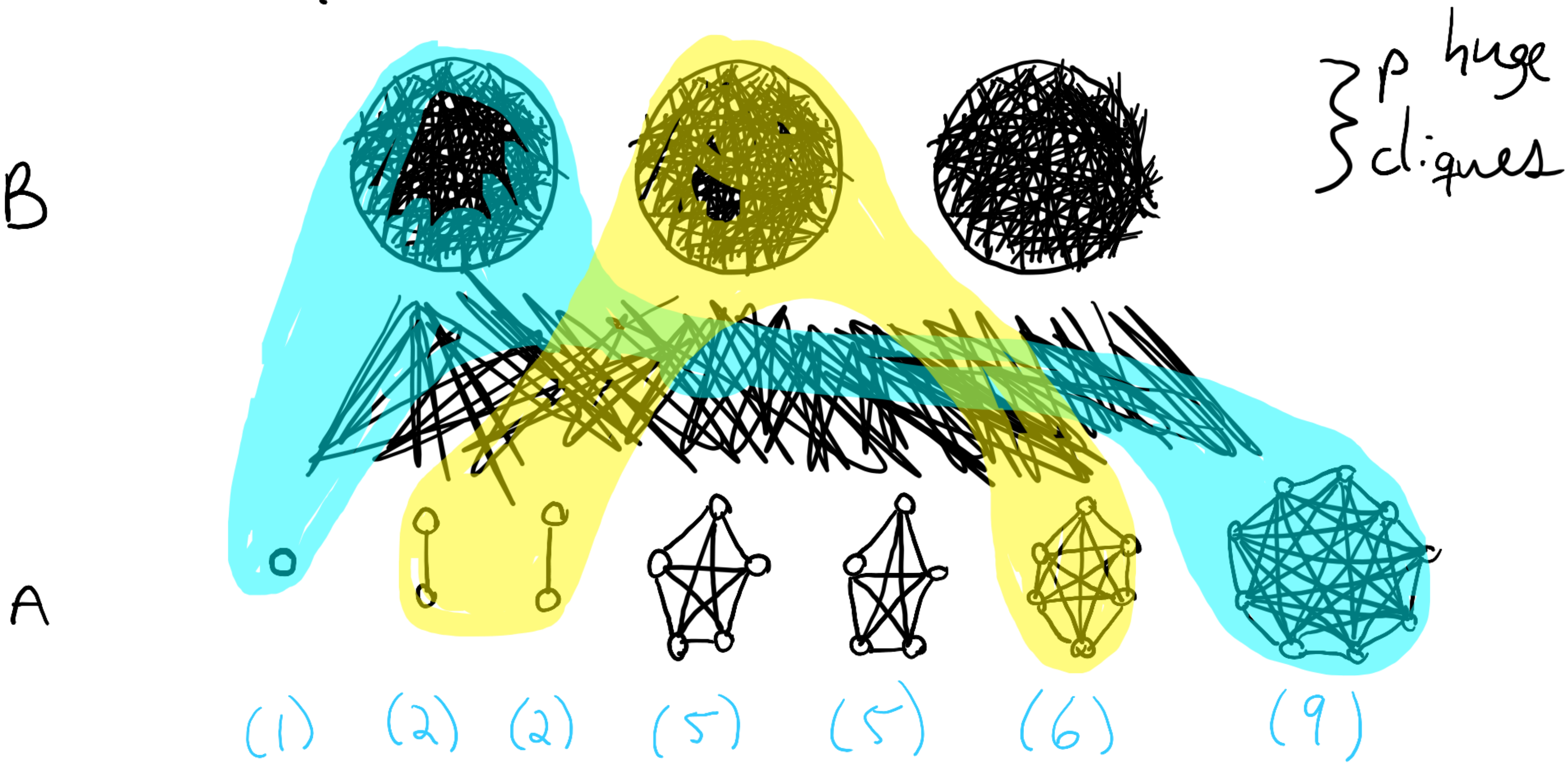
$\mathcal{A} = \{1, 2, 2, 5, 5, 6, 9\}$ $C = 10$ $p = 3$

Cluster-Editing instance



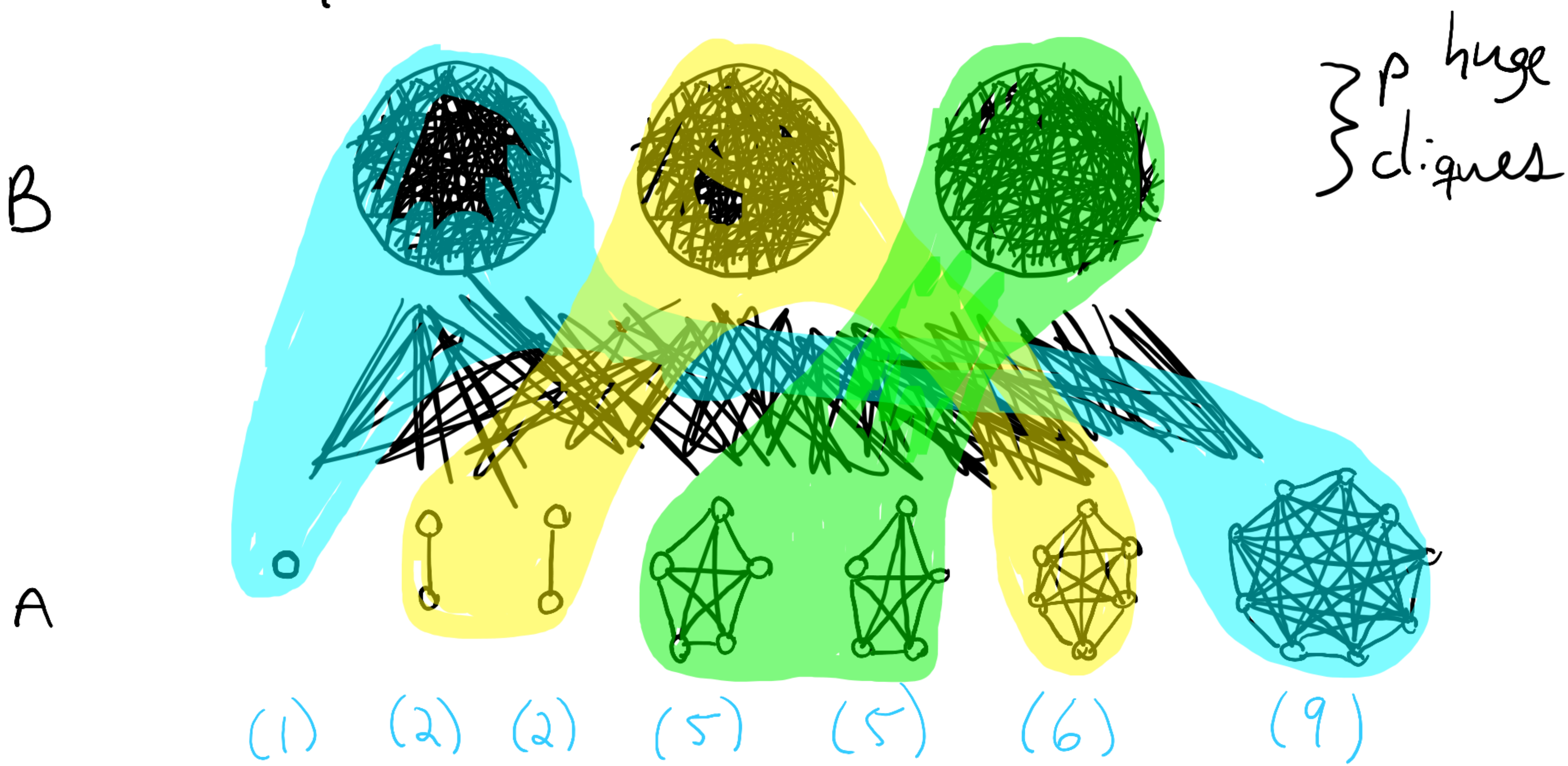
$\mathcal{A} = \{1, 2, 2, 5, 5, 6, 9\}$ $\quad C = 10 \quad p = 3$

Cluster-Editing instance



$\mathcal{A} = \{1, 2, 2, 5, 5, 6, 9\}$ $C = 10$ $p = 3$

Cluster-Editing instance



Main ideas

Huge cliques separated => p clusters

Each little clique A_i goes with a huge clique B_j

Only relevant editing cost = insertions between A_i's in same cluster.

If $A_1, \ldots, A_k$ are together in same cluster, insertions needed = $a_1 a_2 + a_1 a_3 + \cdots + a_{k-1} a_k$.

To prove: sum of edit costs is minimized if each cluster has an equal number of A_i vertices

$$
\begin{aligned}
|F| &= (k-1)ah + \frac{1}{2}\sum_{i=1}^{k}|W_i|^2 - \frac{s}{2} \\
&< (k-1)ah + \sum_{i=1}^{k}|W_i|^2 + \sum_{1\leq i,j\leq k}|W_i||W_j| \\
&= (k-1)ah + \left(\sum_{i=1}^{k}|W_i|\right)^2 \\
&= (k-1)ah + a^2 \\
&< (k-1)ah + h < h^2.
\end{aligned}
$$

The last line implies that having $|\mathcal{C}| = k$, with each element of $\mathcal{C}$ a superset of exactly one element from $\mathcal{I}$, always achieves a lower cost than the other possibilities.

Next, consider the lower bound of $|F| \geq t$ and the conditions on equality. Using the same starting point,

$$
\begin{aligned}
|F| &= (k-1)ah + \frac{1}{2}\sum_{i=1}^{k}|W_i|^2 - \frac{s}{2} \\
&= (k-1)ah + \frac{1}{2k}\left(\sum_{i=1}^{k}|W_i|^2\right)\left(\sum_{i=1}^{k}1^2\right) - \frac{s}{2} \\
&\geq (k-1)ah + \frac{1}{2k}(\sum_{i=1}^{k}|W_i|)^2 - \frac{s}{2} \qquad (1) \\
&= (k-1)ah + \frac{1}{2k}a^2 - \frac{s}{2} = t.
\end{aligned}
$$

In (1), we used the Cauchy-Schwarz inequality, and the two sides are equal if and only if $|W_1| = \cdots = |W_k|$. In addition, $|W_1| = \cdots = |W_k|$ if and only if $|P_1| = \cdots = |P_k|$ (recall that

$$= (k-1)ah + \frac{1}{2k}\left(\sum_{i=1}^{k}|W_i|^2\right)\left(\sum_{i=1}^{k}1^2\right) - \frac{s}{2}$$

$$\geq (k-1)ah + \frac{1}{2k}(\sum_{i=1}^{k}|W_i|)^2 - \frac{s}{2} \quad (1)$$

$$= (k-1)ah + \frac{1}{2k}a^2 - \frac{s}{2} = t.$$

In (1), we used the Cauchy-Schwarz inequality, and the two sides are equal if and only if $|W_1| = \cdots = |W_k|$. In addition, $|W_1| = \cdots = |W_k|$ if and only if $|P_1| = \cdots = |P_k|$ (recall that

Our results

1. Cluster Editing is NP-complete on cographs.
2. p-Cluster Editing is NP-complete on cographs, and W[1]-hard in parameter p on cographs.

Let cw denote clique-width. Cographs have $cw = 2$.

3. p-Cluster Editing admits a $n^{O(cw\, p)}$ time algorithm. Under the ETH, no $n^{o(cw\, p/ \log p)}$ time is possible.
4. For fixed p, p-Cluster Editing on cographs is in P.
5. Cluster Editing is in P on $\{P_4, C_4\}$-free graphs.

Our results

1. ~~Cluster Editing is NP-complete on cographs.~~
2. ~~p-Cluster Editing is NP-complete on cographs, and W[1]-hard in parameter p on cographs.~~

Let cw denote clique-width. Cographs have $cw = 2$.

3. p-Cluster Editing admits a $n^{O(cw\,p)}$ time algorithm. ~~Under the ETH, no $n^{o(cw\,p/\log p)}$ time is possible.~~
4. For fixed p, p-Cluster Editing on cographs is in P.
5. Cluster Editing is in P on $\{P_4, C_4\}$-free graphs.

Our results

1. ~~Cluster Editing is NP-complete on cographs.~~

2. ~~p-Cluster Editing is NP-complete on cographs, and W[1]-hard in parameter p on cographs.~~

Let cw denote clique-width. Cographs have $cw = 2$.

3. p-Cluster Editing admits a $n^{O(cw\, p)}$ time algorithm. ~~Under the ETH, no $n^{o(cw\, p/\log p)}$ time is possible.~~

4. For fixed p, p-Cluster Editing on cographs is in P.

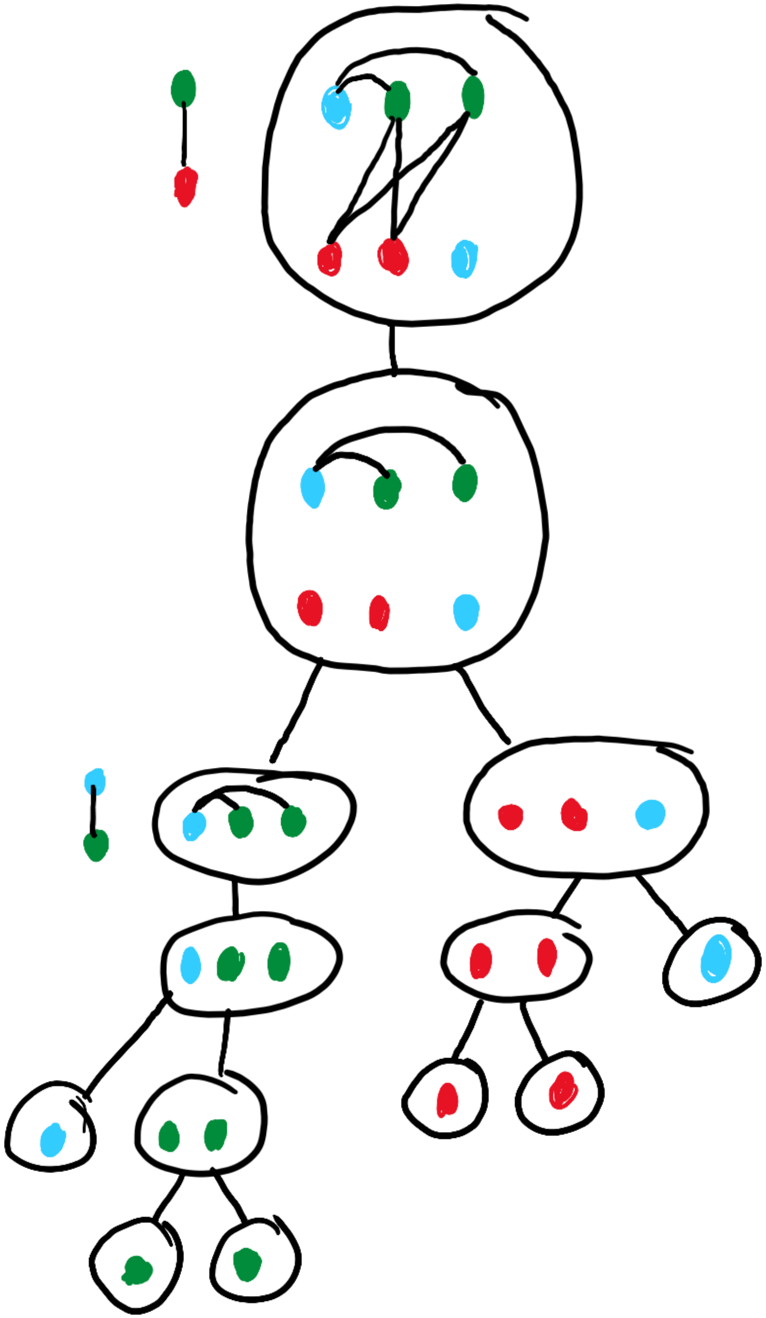
5. Cluster Editing is in P on $\{P_4, C_4\}$-free graphs.

Clique-width

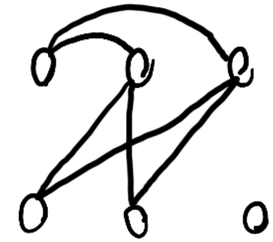
Clique-width uses colored vertices.

A graph G has clique-width k if it can be constructed using k colors and the following operations:

- create a graph with a single vertex colored i
- disjoint union of two colored graphs
- recolor all vertices with color i to color j
- add all edges between vertices of distinct color i and j



G:

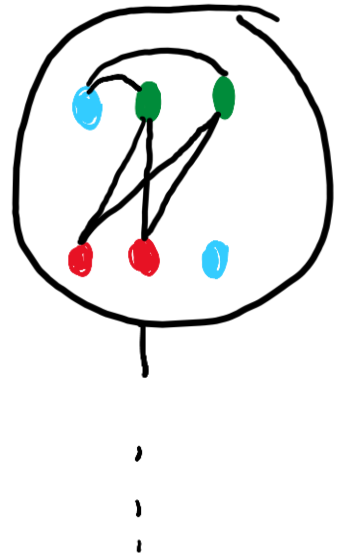


Suppose G is constructed using k colors.

- for each graph encountered during the construction, consider all $p \times k$ matrices M
- $M[i, j] = t$ means "the i-th cluster must have exactly t vertices of color j" (note, $t \leq n$)
- $opt(M) =$ min # edges to edit to achieve a cluster graph that meets all the $M[i, j]$ requirements.

Suppose G is constructed using k colors.

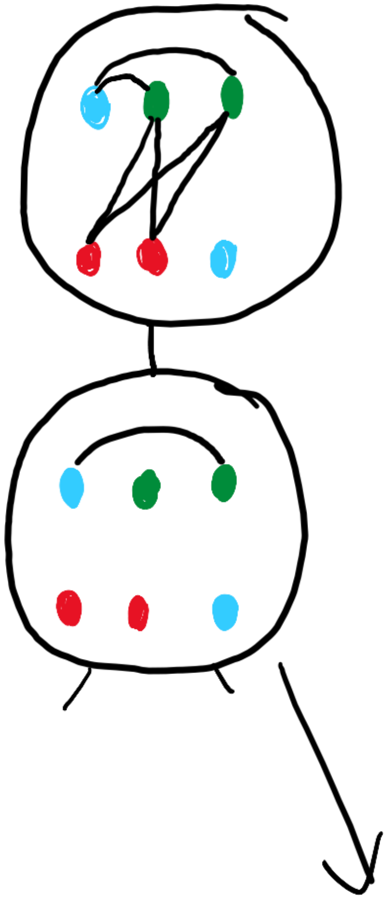
- for each graph encountered during the construction, consider all $p \times k$ matrices M
- $M[i, j] = t$ means “the i-th cluster must have exactly t vertices of color j” (note, $t \leq n$)
- $opt(M) =$ min # edges to edit to achieve a cluster graph that meets all the $M[i, j]$ requirements.
- Compute $opt(M)$ for every possible M and every graph encountered.
- There are $n^{cw \cdot p}$ possible M’s.
- Dynamic programming gives $n^{2cw \cdot p+4}$



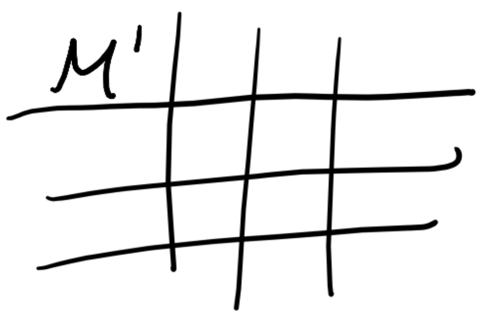
$p = 3$

M	#blue	#gr	#red
cl 1	2	1	3
cl 2	1	0	2
cl 3	2	3	2

$p = 3$



M	#blue	#gr	#red
cl 1	2	1	3
cl 2	1	0	2
cl 3	2	3	2



compute M from appropriate M' at children

Our results

1. ~~Cluster Editing is NP-complete on cographs.~~
2. ~~p-Cluster Editing is NP-complete on cographs, and W[1]-hard in parameter p on cographs.~~

Let cw denote clique-width. Cographs have $cw = 2$.

3. p-Cluster Editing admits a $n^{O(cw\ p)}$ time algorithm. ~~Under the ETH, no $n^{o(cw\ p/\log p)}$ time is possible.~~
4. For fixed p, p-Cluster Editing on cographs is in P.
5. Cluster Editing is in P on $\{P_4, C_4\}$-free graphs.

Our results

1. ~~Cluster Editing is NP-complete on cographs.~~

~~2. p-Cluster Editing is NP-complete on cographs, and W[1]-hard in parameter p on cographs.~~

Let cw denote clique-width. Cographs have $cw = 2$.

3~~. p-Cluster Editing admits a $n^{O(cw\,p)}$ time algorithm. Under the ETH, no $n^{o(cw\,p/\log p)}$ time is possible.~~

~~4. For fixed p, p-Cluster Editing on cographs is in P.~~

5. Cluster Editing is in P on $\{P_4, C_4\}$-free graphs.

Our results

1. ~~Cluster Editing is NP-complete on cographs.~~
2. ~~p-Cluster Editing is NP-complete on cographs, and W[1]-hard in parameter p on cographs.~~

Let cw denote clique-width. Cographs have $cw = 2$.

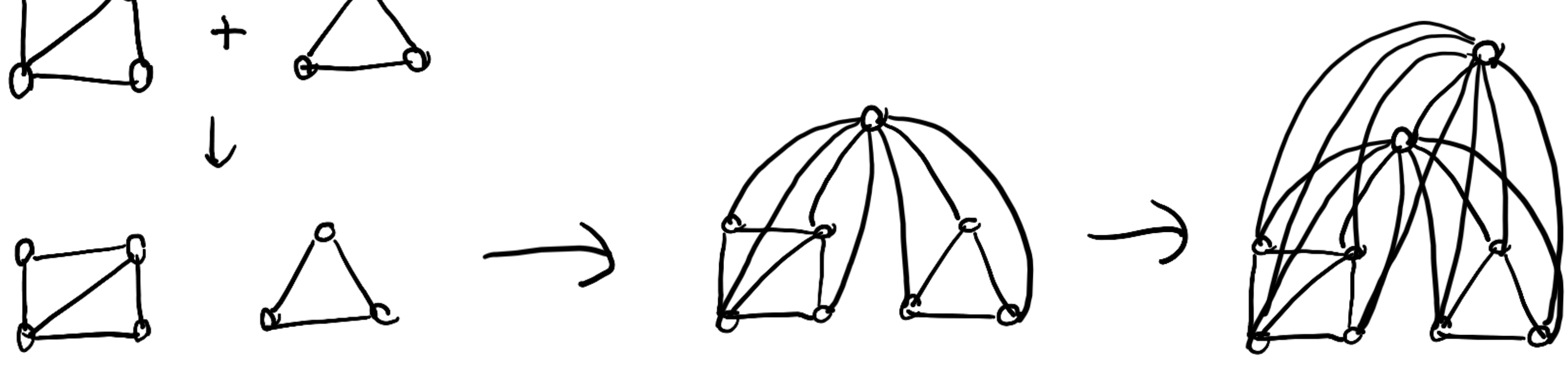
3. ~~p-Cluster Editing admits a $n^{O(cw\ p)}$ time algorithm. Under the ETH, no $n^{o(cw\ p/\log p)}$ time is possible.~~
4. ~~For fixed p, p-Cluster Editing on cographs is in P.~~
5. Cluster Editing is in P on $\{P_4, C_4\}$-free graphs.

$\{P_4, C_4\}$-free graphs

Also known as Trivially Perfect Graphs (TPG).

Can be built with the operations:

- create a single vertex
- disjoint union of TPGs.
- add a universal vertex (adjacent to all vertices currently there)



$\{P_4, C_4\}$-free graphs

Idea: when adding a universal vertex v,

- Take an optimal solution of $G - v$.
- Add v in the largest cluster (minimizes deletions)

$\{P_4, C_4\}$-free graphs

Idea: when adding a universal vertex v,

- Take an optimal solution of $G - v$.
- Add v in the largest cluster (minimizes deletions)
- Problem: optimal solution may not be good later on.

$\{P_4, C_4\}$-free graphs

Idea: when adding a universal vertex v,

- Take an optimal solution of $G - v$.
- Add v in the largest cluster (minimizes deletions)
- Problem: optimal solution may not be good later on.
- For all sizes q, compute

$opt(G, q)$ = min # editions in G s.t. the largest cluster has exactly q vertices.

- Easy to update when adding v, just use $opt(G - v, q)$

$\{P_4, C_4\}$-free graphs

- Difficult part: update tables when taking disjoint unions, i.e., $G = G_1 \cup G_2$.
- $opt(G, q)$ = min # editions in G s.t. the largest cluster has exactly q vertices.

$\{P_4, C_4\}$-free graphs

- Difficult part: update tables when taking disjoint unions, i.e., $G = G_1 \cup G_2$.
- $opt(G, q)$ = min # editions in G s.t. the largest cluster has exactly q vertices.
- Argue that we can just take

$$opt(G, q) = min_{q=q_1+q_2} opt(G_1, q_1) + opt(G_2, q_2) + \delta$$

i.e., it is safe to merge the largest clusters of G_1 and G_2

Future directions

- Is p-Cluster Editing in P for TPGs?
- cw is a bad parameter for Cluster Editing. Treewidth? Modular-width? Other?
- Challenge: get a dichotomy theorem to characterize graph classes on which Cluster Editing/Deletion is in P, or NP-hard.