On the Complexity of 2-Club Cluster Editing with Vertex Splitting[1]

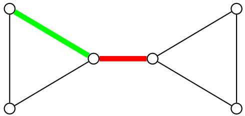
Faisal N. Abu-Khzam, Tom Davot, Lucas Isenmann, Sergio Thoumi

September 19, 2025

[1] This research project was supported by the Lebanese American University under the President's Intramural Research Fund PIRF0056.

Cluster Editing

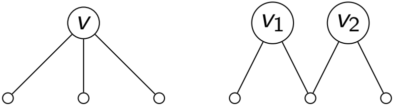
- Given a graph G and an integer k, the objective is to transform G into a disjoint union of cliques (the clusters) via at most k edge editing/modification operations (add/delete).



- Models correlation clustering: partition the input data set so that elements of the same set are close-enough and pairs from different sets are not close according to a given similarity measure (represented by edges of a graph...)

Vertex Splitting

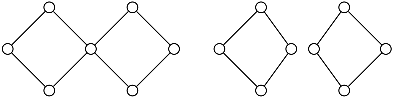
- A **vertex splitting** consists of replacing a vertex v by two non-adjacent vertices such that each vertex adjacent to v is adjacent to at least one of the copies.



Example of a split of v into v_1 and v_2. In general, it is possible that the copies share some neighbors.

2CCVS

- A **2-club** is a graph having diameter at most 2.
- Given a graph G and an integer k, the objective is to transform G into a disjoint union of 2-clubs via at most k vertex splittings.

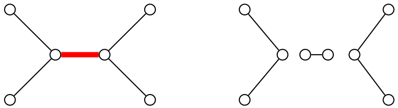


2CCEDVS

- Given a graph G and an integer k, the objective is to transform G into a disjoint union of 2-clubs via at most k edge deletions and vertex splittings.

$$2ccedvs(G) \leq 2ccvs(G) \leq 2 \cdot 2ccedvs(G)$$

because we can replace an edge deletion by two vertices splittings.



Why Vertex Splitting

- Allows data elements to belong to more than one cluster/group.
- Allows clustering of data that is hard to cluster (e.g. due to hubness).

Our results

Our results for 2CCVS and 2CCEDVS:

- NP-complete
- APX-hard
- Polynomial on trees
- FPT parameterized by solution size

Reduction from 3-SAT

Using a reduction from 3-SAT, we obtain the following:

Theorem

2CCEDVS and 2CCVS are NP-complete even when restricted to bipartite planar graphs of maximum degree three.

Theorem

Assuming the ETH, there is no $O^*(2^{o(n)})$ (resp. $O^*(2^{o(\sqrt{n})})$)-time algorithm for 2CCEDVS (resp. planar) graphs with maximum degree 3 where n is the number of vertices of the graph.

Theorem

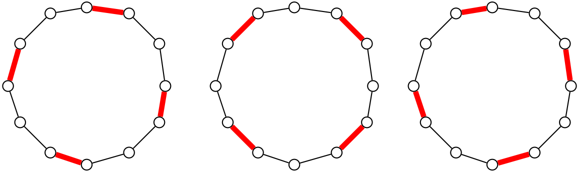
*Assuming the ETH, there is no $O^*(2^{o(n)})$ (resp. $O^*(2^{o(\sqrt{n})})$)-time algorithm for 2CCVS (resp. planar) graphs with maximum degree **4** where n is the number of vertices of the graph.*

Theorem

2CCEDVS and 2CCVS are APX-hard.

Reduction from 3-SAT

Transforming a cycle of length $6k$ into a disjoint union of 2-clubs requires at least $2k$ edge deletions or vertex splittings. This is realized by three possible sequences of $2k$ edge deletions.

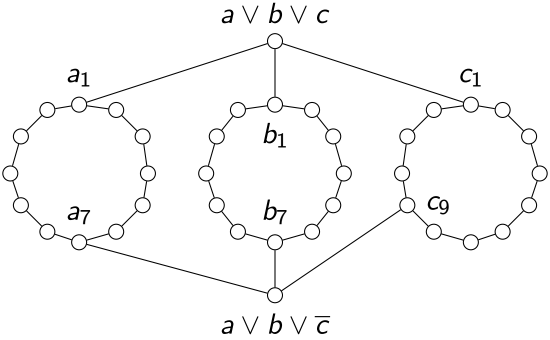


Applying vertex splitting (in this case) increases the number of operations.

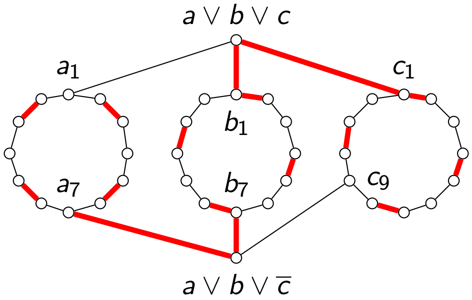
Reduction from 3-SAT for 2CCEDVS

Sketch:

- For every variable we create a cycle
- For every clause we create a vertex
- For every variable appearing in a clause we connect the vertex of the clause to a specific vertex of the cycle of the variable



Reduction from 3-SAT for 2CCEDVS



From a satisfying assignment $a = True$, $b = False$, $c = False$.

Theorem

A formula ϕ with n variables and m clauses is satisfiable if and only if G_ϕ can be turned into a union of 2-clubs with a sequence of length at most $2m + \sum_{v \in V} 2d(v)$ of operations (where $d(v)$ denotes the number of clauses where v appears).

Sequence to assignment

Given a sequence of length $2m + \sum_{v \in V} 2d(v)$, we prove that

- Each variable v cycle requires $2d(v)$ operations
- Each clause gadget requires 2 operations

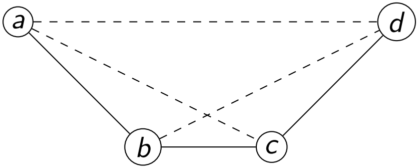
Because of the previous Lemma, if we delete all the edges $v_{2+3k}v_{3+3k}$ for every k for every variable v, then we set v to True. Otherwise we set v to False.

We show that this assignment is satisfying the formula.

FPT algorithm

Based on a branching algorithm.

If there exists a shortest path (a, b, c, d) of length 3:



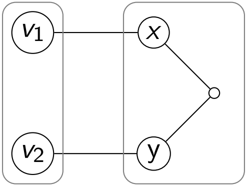
One of these operations must be done to get an union of 2-clubs:

- Delete one the three edges ab, bc, cd.
- Split b so that we separate a and c.
- Split c so that we separate b and d.

Lemma: Consider an optimal sequence. Let S be the set of vertices that are split. If $v \in S$ and C is a connected component of $G[V \setminus S]$, then each copy of v is either adjacent to all vertices of $C \cap N(v)$ or to none of them.

$C \cap N(v)$

Proof. By contradiction there exists v_1 and v_2 two copies of v s.t.

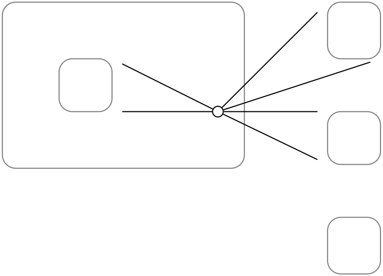


As C is connected and as the sequence is such that the obtained components are 2-clubs, we deduce that we can merge v_1 and v_2 and we still have got a 2-club. It contradicts minimality.

Lemma: Given a subset S of vertices and e the number of extra splits such that $s + e \leq k$. There are $O((4 \cdot 2^{6k})^{2k})$ ways to split the marked vertices S.

Proof.

- We can split at most $s + e \leq k$ times
- There are at most $s + s + e \leq 2k$ vertices which can be split (S and the copies of the splits)
- There is at most 2^{2k} subsets in S and its copies
- There is at most $k + 1$ connected components
- There is at most 2^{k+1} choices for the neighbors of a copy



Algorithm:

- Until there is no shortest path of length 3, we branch on the 5 ways to solve such a path.
- If there is still such a path after k operations, then we stop.
- Otherwise, we try to find a way to split every vertices marked for splitting at least once according to the previous Lemma.

Theorem

The complexity is $O(n^3 5^k \cdot (4 \cdot 2^{6k})^{2k})$.

Proof.

There are 5 branches for each shortest path. Detecting a shortest path can be done in $O(n^3)$. Finding a good splittings can be done in $O((4 \cdot 2^{6k})^{2k})$. $\square$

Other/Future Results/Work

- We further show that both 2CCVS and 2CCEDVS are solvable in polynomial-time on trees.
- It is FPT parameterized by treewidth
- Is there a polynomial kernel for 2CCVS or 2CCEDVS?
- How about other parameterizations?
- Can we complement our approximation hardness result by an approximation algorithm?

Thank you for your attention!