A Quadratic-Vertex Problem Kernel for s-Plex Cluster Vertex Deletion

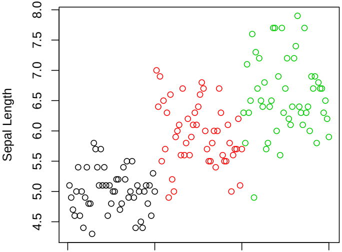
René van Bevern
rene.bevern@uni-jena.de

Institut für Informatik
Friedrich-Schiller-Universität Jena

November 9th, 2009

Talk accompanying my Studienarbeit.
Refer to arXiv:0909.2814v1 [cs.DM].

Data Clustering

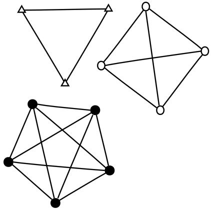


Three plant species and sepal lengths

Input: Objects and similarity values between objects

Task: Find groups (*clusters*) of similar objects, so that objects within a cluster are similar, while objects in distinct clusters are not.

Graph-Based Data Clustering



Map given objects and similarities to a graph:

- vertices of the graph: the given objects
- draw edges between similar objects

Model clusters using *cliques*:

- If all connected components are cliques, each clique forms a cluster.
- Otherwise, we transform the graph into a so-called *cluster graph*.

Cliques as a Model for Clusters

Given a graph G and an integer k, two problems to model data clustering are:

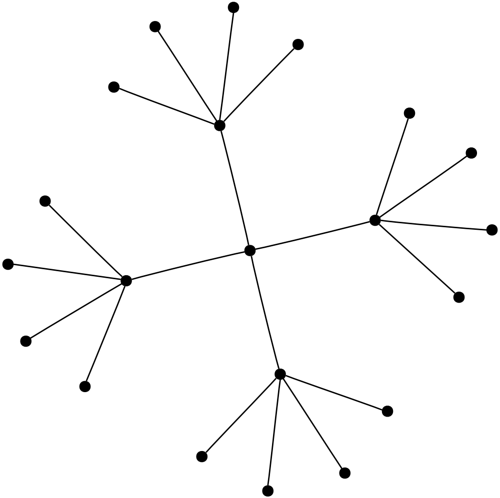
CLUSTER VERTEX DELETION: can G be transformed into a cluster graph by removing at most k vertices?

CLUSTER EDITING: can G be transformed into a cluster graph by inserting or deleting at most k edges?

Both problems are NP-complete and fixed-parameter tractable with respect to the parameter k.

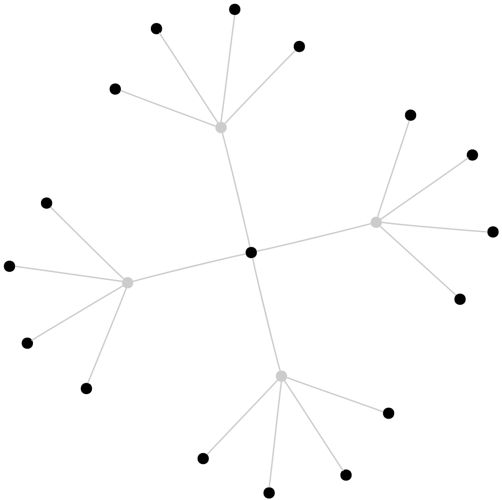
Example for Cluster Vertex Deletion

The goal is to transform this graph into a cluster graph.

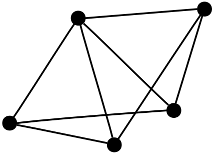


Example for Cluster Vertex Deletion

Removing four vertices results in too many, too small clusters.



A Different Model for Clusters


2-plex

We use a relaxation of the clique concept.

Definition. For $s \geq 1$, an *s-plex* is a graph in which every vertex is nonadjacent to at most $s-1$ other vertices.

- 1-plex = clique
- *s-plex cluster graph*: graph which has s-plexes as connected components

Generalizing Cluster Vertex Deletion

Given a graph G and a natural number k, we consider:

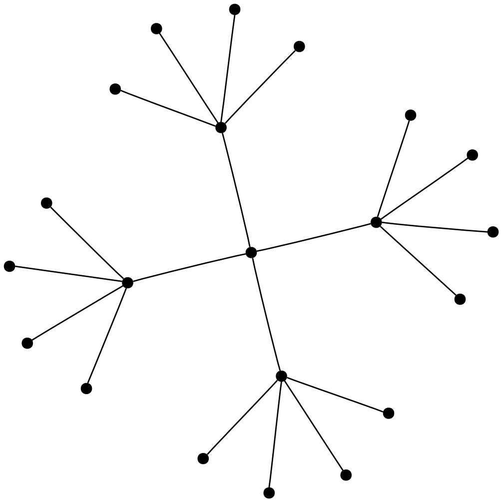
s-Plex Cluster Vertex Deletion: can G be transformed into an s-plex cluster graph by at most k vertex deletions?

s-Plex Editing: can G be transformed into an s-plex cluster graph by most k edge insertions/deletions?

Both problems are NP-complete and fixed-parameter tractable with respect to the parameter k.

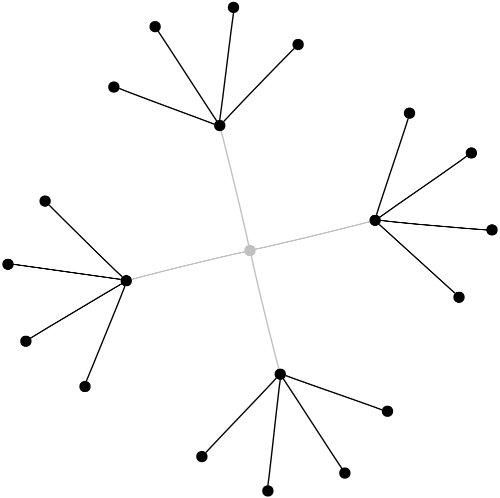
Example for 4-Plex Cluster Vertex Deletion

The goal is to transform this graph into a 4-plex cluster graph.



Example for 4-Plex Cluster Vertex Deletion

Only one vertex has to be removed.



Benefits of s-Plex Cluster Vertex Deletion

Benefits over CLUSTER VERTEX DELETION:

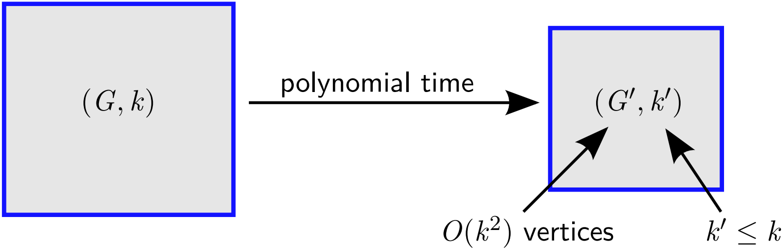
- balance number/size of clusters against number of vertices being removed

Benefits over s-PLEX EDITING:

- smaller parameter values, possibly faster algorithms
- outlier detection

2-Plex Cluster Vertex Deletion

We now present an $O(k^2)$-vertex problem kernel for 2-Plex Cluster Vertex Deletion.

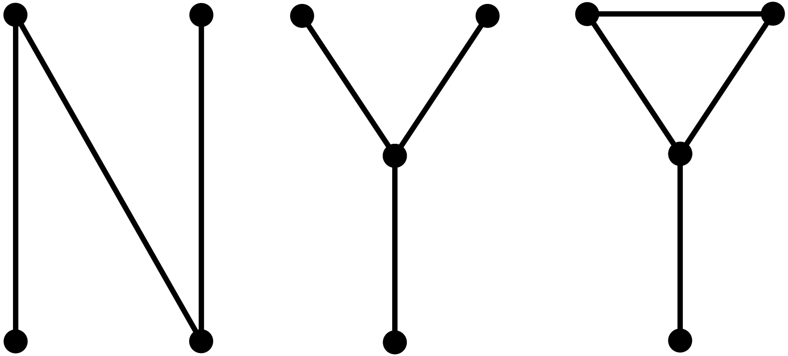


(G, k) is yes-instance $\iff$ (G', k') is yes-instance

The Challenge

- s-plex cluster graphs: characterized by forbidden induced subgraphs (FISGs) with $O(s+\sqrt{s})$ vertices [Guo et al., AAIM'09]
- minimization version of s-PLEX CLUSTER VERTEX DELETION is in MIN $\mathsf{F}^+\Pi_1$
- gives $k^{O(s+\sqrt{s})}$-vertex problem kernel [Kratsch, STACS'09]
- we can show an $O(k^2 s^3)$-vertex problem kernel

FISGs for 2-Plex Cluster Vertex Deletion



Kernelization Steps

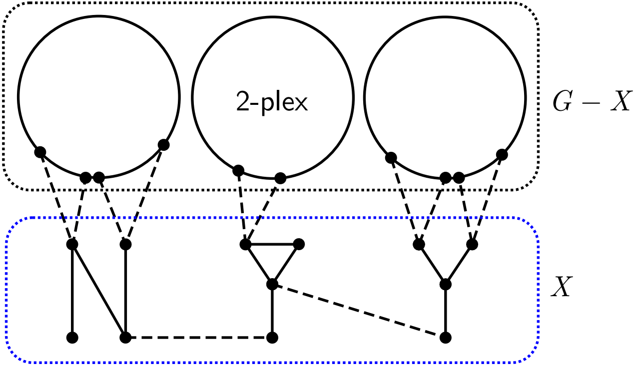
Our kernelization comprises three steps:

- Approximation Step
- Tidying Step
- Shrinking Step

We call this method “Kernelization Through Tidying”.

Approximation Step

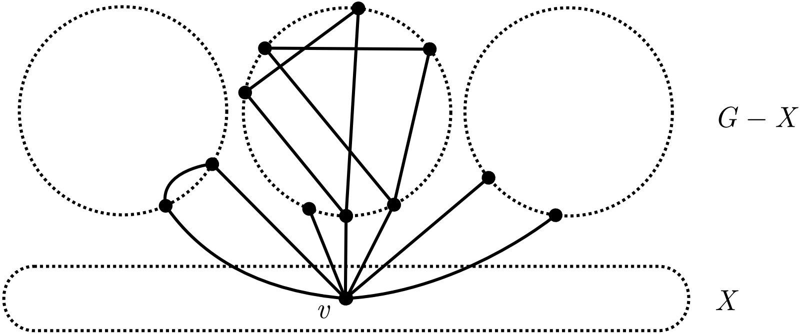
Compute a set X containing the vertices of a maximal set of pairwise vertex-disjoint FISGs.



If (G, k) is a yes-instance, we have $|X| \leq 4k$.

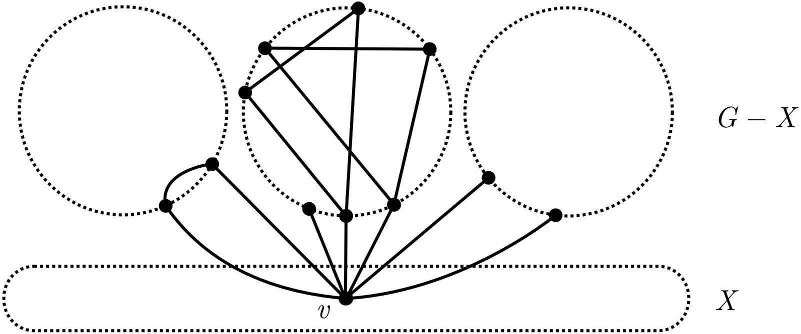
Tidying Step

Remove vertices occuring in a high number of FISGs.



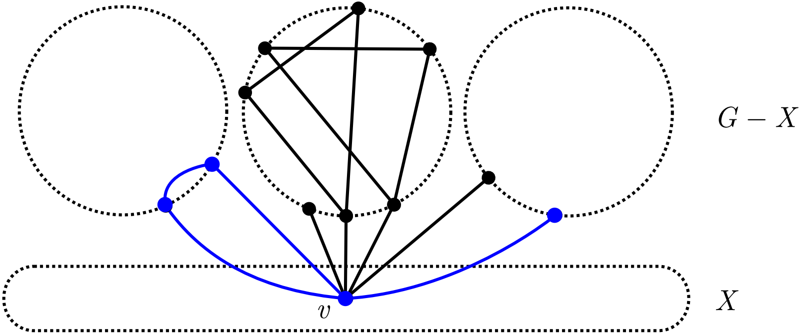
Tidying Step

To this end, find FISGs pairwise intersecting only in v.



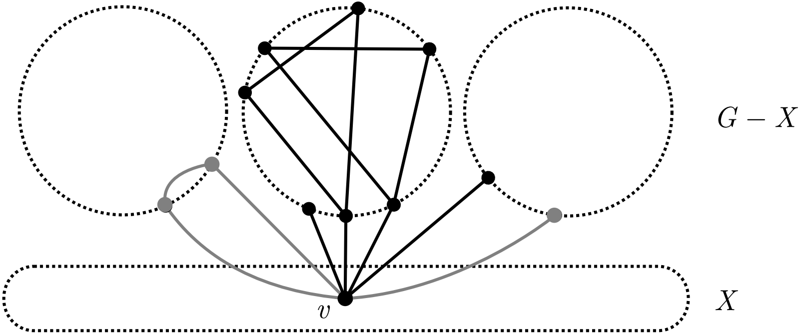
Tidying Step

Find a FISG including v.



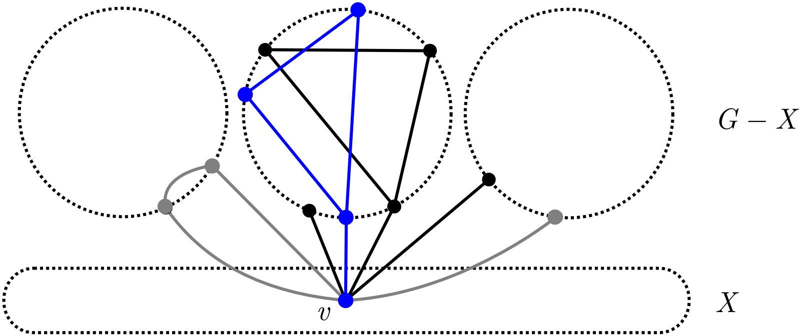
Tidying Step

Paint the vertices other than v gray.



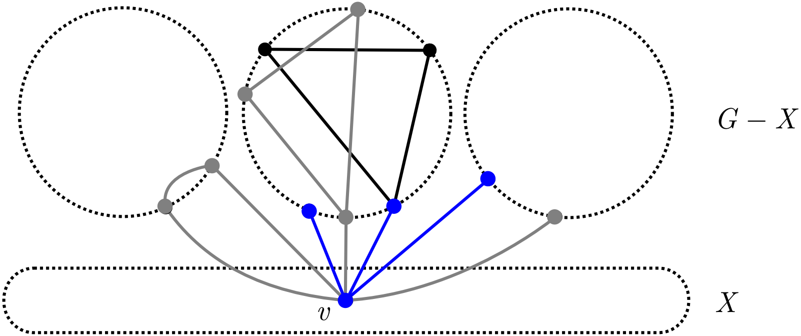
Tidying Step

Find a new FISG of only black vertices containing v.



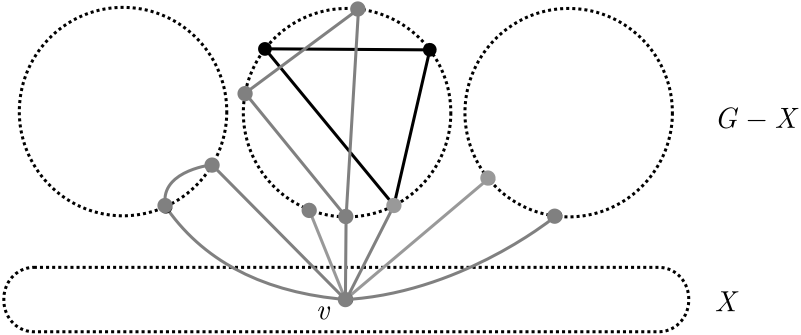
Tidying Step

Repeatedly find FISGs of only black vertices containing v...



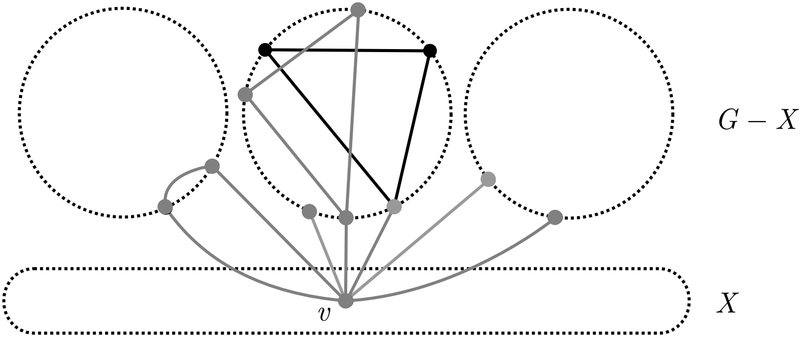
Tidying Step

... until no more black FISG can be found.



Tidying Step

Repeat this procedure for every $v \in X$.

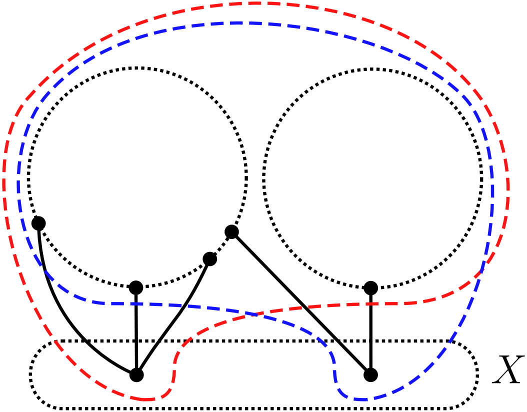


Tidying Step

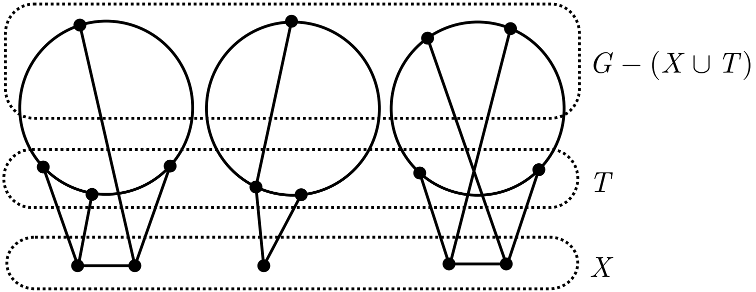
- We call the set of all grey vertices the *tidying set* T.
- If more than k FISGs pairwise intersect in a single $v \in X$: remove v; decrement k by one.
- Hence, $|T| \leq 3k \cdot |X| = 3k \cdot 4k = 12k^2$.

Local Tidiness

For each vertex $v \in X$, removing X except v from $G - T$ results in a 2-plex cluster graph. We call this property *local tidiness*.



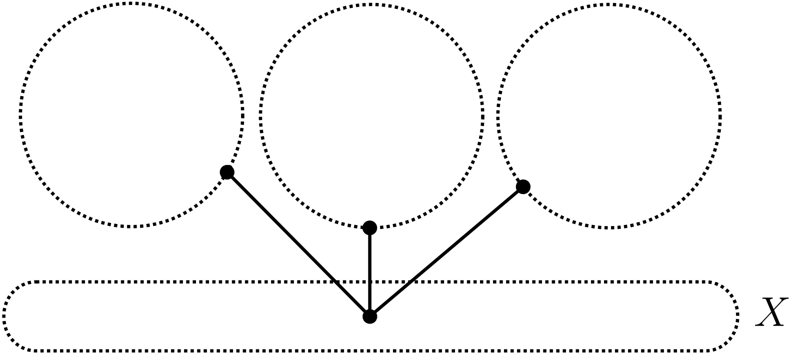
Shrinking Step



- We have $|X| + |T| \in O(k^2)$, it remains to bound the number of vertices in $G - (X \cup T)$.
- We exploit the local tidiness property.

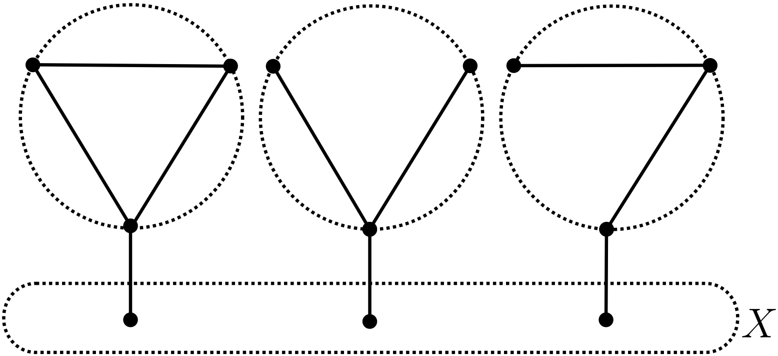
Implications of Local Tidiness

No vertex in X is adjacent to more than two 2-plexes of $G - (T \cup X)$.



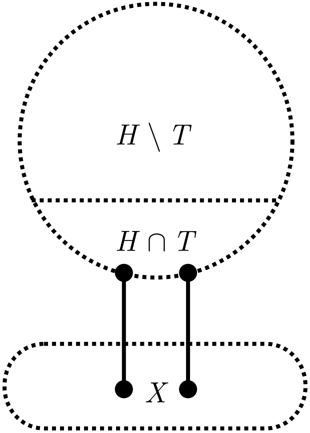
Implications of Local Tidiness

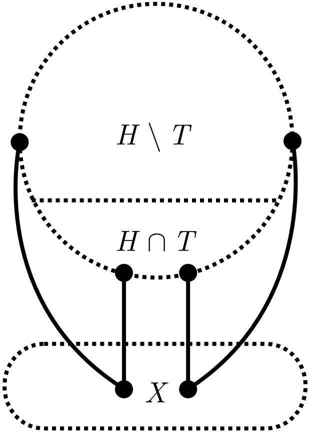
If a vertex in X is adjacent to a 2-plex in $G - (T \cup X)$, it must be adjacent to all but at most one of its vertices.



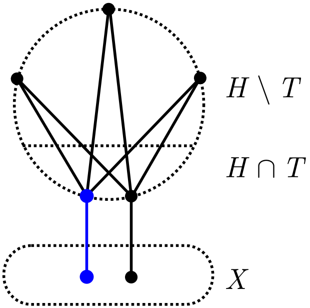
Implications of Local Tidiness

A 2-plex H in $G - X$ can be *separated* or *non-separated*. We assume that isolated 2-plexes have been removed from G.





Data Reduction Rule for Separated 2-Plexes

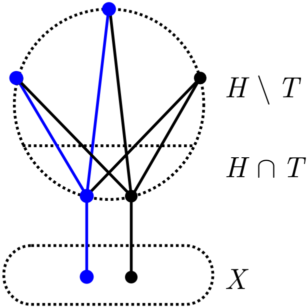


Blue: not in solution

Consider separated s-plex H with $|H \setminus T| \geq |H \cap T| + 1$.

- Assume: solution does not cover all edges between $H \cap T$ and X.

Data Reduction Rule for Separated 2-Plexes

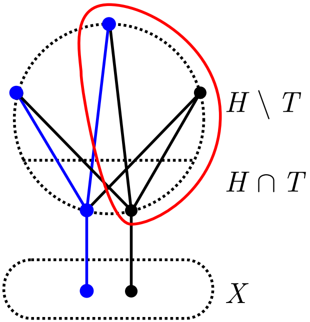


Blue: not in solution

Consider separated s-plex H with $|H \setminus T| \geq |H \cap T| + 1$.

- Assume: solution does not cover all edges between $H \cap T$ and X.
- It must contain all but at most one vertex from $H \setminus T$.

Data Reduction Rule for Separated 2-Plexes

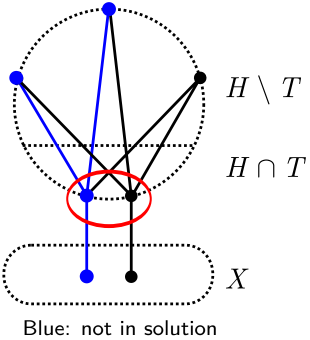


Blue: not in solution

Consider separated s-plex H with $|H \setminus T| \geq |H \cap T| + 1$.

- Assume: solution does not cover all edges between $H \cap T$ and X.
- It must contain all but at most one vertex from $H \setminus T$.

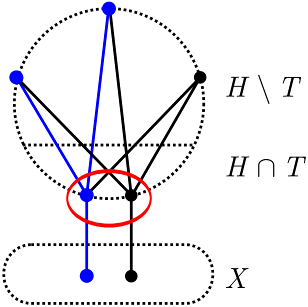
Data Reduction Rule for Separated 2-Plexes



Consider separated s-plex H with $|H \setminus T| \geq |H \cap T| + 1$.

- Assume: solution does not cover all edges between $H \cap T$ and X.
- It must contain all but at most one vertex from $H \setminus T$.
- There is always a solution that covers the edges and is not larger.

Data Reduction Rule for Separated 2-Plexes



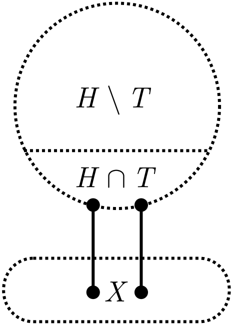
Blue: not in solution

Consider separated s-plex H with $|H \setminus T| \geq |H \cap T| + 1$.

- Assume: solution does not cover all edges between $H \cap T$ and X.
- It must contain all but at most one vertex from $H \setminus T$.
- There is always a solution that covers the edges and is not larger.

Rule: remove vertices from $H \setminus T$ while $|H \setminus T| > |H \cap T| + 1$.

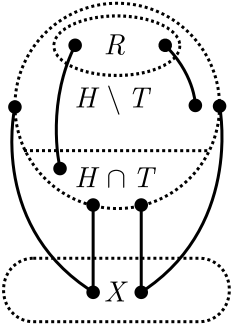
Sizes of Separated 2-Plexes



At most $O(k^2)$ vertices are contained in separated 2-plexes, because:

- The set $H \cap T$ is nonempty: there are at most $|T|$ separated 2-plexes.
- For each separated 2-plex, we have $|H \setminus T| \leq |H \cap T| + 1$.
- We have $|T| \in O(k^2)$.

Data Reduction Rule for Non-Separated 2-Plexes



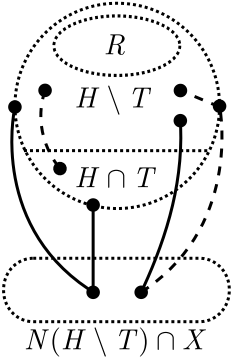
Find a *redundant* subset $R \subseteq H \setminus T$ such that

- For each $u, v \in R$, it holds that $N(u) \cap X = N(v) \cap X$.
- Each vertex $v \in R$ is adjacent to all vertices in $H \setminus R$.

Remove vertices from R while $|R| > k + 3$.

- Refer to arXiv:0909.2814v1 [cs.DM]. (Studienarbeit with the title of this talk)

Finding Redundant Subsets

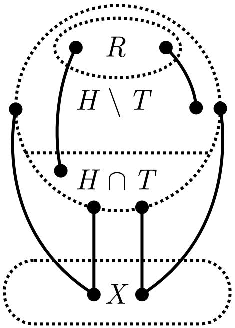


We can efficiently find a redundant subset $R \subseteq H \setminus T$ such that $|H \setminus R|$ is bounded by

$$2|H \cap T| + 2|N(H \setminus T) \cap X|.$$

This is due to local tidiness: a neighbor of $H \setminus T$ in X must be adjacent to all but at most one vertex in $H \setminus T$.

Sizes of Non-Separated 2-Plexes



- Local tidiness: each vertex in X is adjacent to at most two non-separated 2-plexes. Thus, there are at most $2|X| \in O(k)$ of them.
- Each non-separated 2-plex contains $O(k)$ vertices in a redundant set R and $O(|H \cap T| + |X|)$ other vertices.

This sums up to $O(|T| + k|X|) = O(k^2)$ for all non-separated 2-plexes.

Kernel Size

After data reduction rules, G contains at most $O(k^2)$ vertices:

- $O(k)$ vertices in approximate solution X
- $O(k^2)$ vertices in tidying set T
- $O(k^2)$ vertices in separated 2-plexes
- $O(k^2)$ vertices in non-separated 2-plexes

Conclusion

We can show an $O(k^2 s^3)$ problem kernel for s-Plex Cluster Vertex Deletion, which can be found in $O(ksn^2)$ time.

- Smarter algorithm for Tidying Step: brute force finding of Fisgs with $O(s + \sqrt{s})$ vertices would take $n^{O(s+\sqrt{s})}$ time.
- Definition of redundant sets is more difficult.

Kernelization Through Tidying is likely to be applicable to many problems allowing a characterization by Fisgs.

References

René van Bevern.
A quadratic-vertex problem kernel for s-plex cluster vertex deletion.
Studienarbeit, FSU Jena, Germany, 2009.
Available electronically, arXiv:0909.2814v1.

Jiong Guo, Christian Komusiewicz, Rolf Niedermeier, and Johannes Uhlmann.
A more relaxed model for graph-based data clustering: s-plex editing.
In *Proc. 5th AAIM*, volume 5564 of *LNCS*, pages 226–239. Springer, 2009.

Falk Hüffner, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier.
Fixed-parameter algorithms for cluster vertex deletion.
Theory Comput. Syst., 2009.
Available electronically. Preliminary version at *LATIN 2008*.

Stefan Kratsch.
Polynomial kernelizations for MIN $\mathrm{F}^{+}\Pi_1$ and MAX NP.
In *Proc. 26th STACS*, pages 601–612. IBFI Dagstuhl, Germany, 2009.