Fixed-Parameter and Integer Programming Approaches for Clustering Problems

Falk Hüffner

joint work with

Sharon Bruckner[1] Christian Komusiewicz[2]
Adrian Liebtrau[3] Rolf Niedermeier[2] Sven Thiel[3]
Johannes Uhlmann[2]

[1]Institut für Mathematik, Freie Universität Berlin

[2]Institut für Softwaretechnik und Theoretische Informatik, TU Berlin

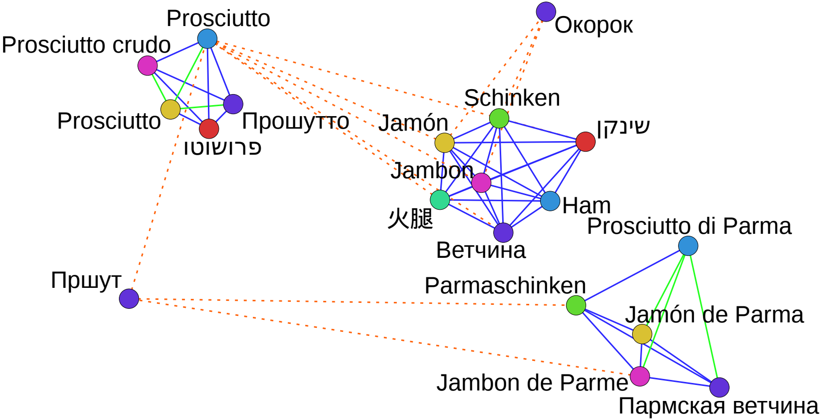
[3]Institut für Informatik, Friedrich-Schiller-Universität Jena

27 September 2013

Wikipedia interlanguage links

Wikipedia interlanguage links

Wikipedia interlanguage link graph example



Model

Colorful Components

Instance: An undirected graph $G = (V, E)$ and a coloring of the vertices $\chi : V \to \{1, \ldots, c\}$.
Task: Delete a minimum number of edges such that all connected components are *colorful*, that is, they do not contain two vertices of the same color.

Complexity of Colorful Components

- COLORFUL COMPONENTS with two colors can be solved in $O(\sqrt{n}m)$ time by matching techniques.

Complexity of Colorful Components

- COLORFUL COMPONENTS with two colors can be solved in $O(\sqrt{n}m)$ time by matching techniques.
- COLORFUL COMPONENTS is NP-hard already with three colors.

Complexity of Colorful Components

- COLORFUL COMPONENTS with two colors can be solved in $O(\sqrt{n}m)$ time by matching techniques.
- COLORFUL COMPONENTS is NP-hard already with three colors.
- COLORFUL COMPONENTS can be approximated by a factor of $4\ln(c+1)$.

Fixed-parameter algorithm

Observation

COLORFUL COMPONENTS can be seen as the problem of destroying by edge deletions all bad paths, that is, simple paths between equally colored vertices.

Fixed-parameter algorithm

Observation

COLORFUL COMPONENTS can be seen as the problem of destroying by edge deletions all bad paths, that is, simple paths between equally colored vertices.

Observation

Unless the graph is already colorful, we can always find a bad path with at most c edges, where c is the number of colors.

Fixed-parameter algorithm

Observation

COLORFUL COMPONENTS can be seen as the problem of destroying by edge deletions all bad paths, that is, simple paths between equally colored vertices.

Observation

Unless the graph is already colorful, we can always find a bad path with at most c edges, where c is the number of colors.

Theorem

COLORFUL COMPONENTS can be solved in $O(c^k \cdot m)$ time, where k is the number of edge deletions.

Limits of fixed-parameter algorithms

Exponential Time Hypothesis (ETH)

3-SAT cannot be solved within a running time of $2^{o(n)}$ or $2^{o(m)}$.

Limits of fixed-parameter algorithms

Exponential Time Hypothesis (ETH)

3-SAT cannot be solved within a running time of $2^{o(n)}$ or $2^{o(m)}$.

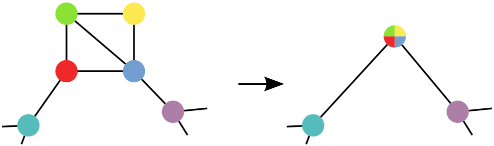
Theorem

Colorful Components with three colors cannot be solved in $2^{o(k)} \cdot n^{O(1)}$ unless the ETH is false.

Data reduction

Data reduction

Let $V' \subseteq V$ be a colorful subgraph. If the cut between V' and $V \setminus V'$ is at least as large as the connectivity of V', then merge V' into a single vertex.



Method 1: Implicit Hitting Set

Hitting Set

Instance: A ground set U and a set of *circuits* $S_1, \ldots, S_n$ with $S_i \subseteq U$ for $1 \leqslant i \leqslant n$.
Task: Find a minimum-size *hitting set*, that is, a set $H \subseteq U$ with $H \cap S_i \neq \emptyset$ for all $1 \leqslant i \leqslant n$.

Method 1: Implicit Hitting Set

HITTING SET

Instance: A ground set U and a set of *circuits* $S_1, \ldots, S_n$ with $S_i \subseteq U$ for $1 \leqslant i \leqslant n$.
Task: Find a minimum-size *hitting set*, that is, a set $H \subseteq U$ with $H \cap S_i \neq \emptyset$ for all $1 \leqslant i \leqslant n$.

Observation

We can reduce COLORFUL COMPONENTS to HITTING SET: The ground set U is the set of edges, and the circuits to be hit are the paths between identically-colored vertices.

Method 1: Implicit Hitting Set

Hitting Set

Instance: A ground set U and a set of *circuits* $S_1, \ldots, S_n$ with $S_i \subseteq U$ for $1 \leqslant i \leqslant n$.
Task: Find a minimum-size *hitting set*, that is, a set $H \subseteq U$ with $H \cap S_i \neq \emptyset$ for all $1 \leqslant i \leqslant n$.

Observation

We can reduce Colorful Components to Hitting Set: The ground set U is the set of edges, and the circuits to be hit are the paths between identically-colored vertices.

Problem

Exponentially many circuits!

Method 1: Implicit Hitting Set

In an *implicit hitting set* problem, the circuits have an implicit description, and a polynomial-time oracle is available that, given a putative hitting set H, either confirms that H is a hitting set or produces a circuit that is not hit by H.

Method 1: Implicit Hitting Set

In an *implicit hitting set* problem, the circuits have an implicit description, and a polynomial-time oracle is available that, given a putative hitting set H, either confirms that H is a hitting set or produces a circuit that is not hit by H.

Several approaches to solving implicit hitting set problems are known, which use an ILP solver as a black box for the HITTING SET subproblems.

Method 2: Row generation

Idea

Instead of using the ILP solver as a black box, we can use *row generation* ("*lazy constraints*") to add constraints inside the solver.

Method 3: Clique Partitioning ILP formulation

- 0/1 variables for each vertex pair indicates whether it is contained in a cluster
- Constraints ensure consistency

Cutting Planes

Definition

A *cutting plane* is a valid constraint that cuts off fractional solutions.

Cutting Planes

Definition

A *cutting plane* is a valid constraint that cuts off fractional solutions.

Tree cut

Let $T = (V_T, E_T)$ be a subgraph of G that is a tree such that all leaves L of the tree have color c, but no inner vertex has. Then

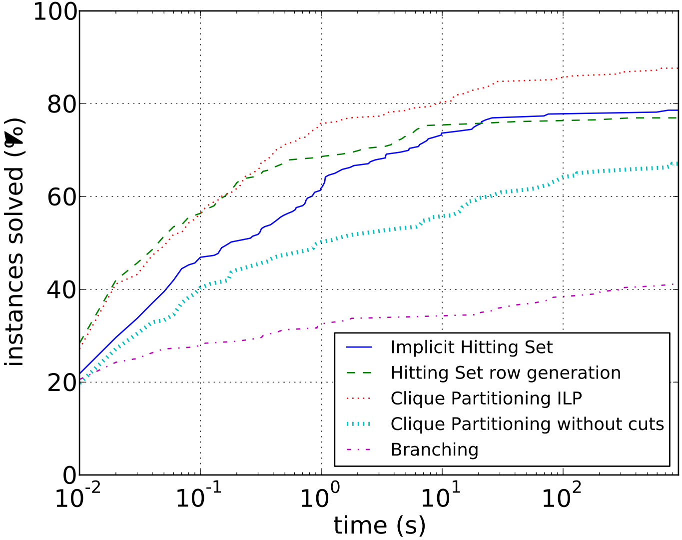
$$\sum_{e \in E_T} d_e \geqslant |L| - 1$$

is a valid inequality.

Wikipedia interlanguage links

- 30 most popular languages
- 11,977,500 vertices, 46,695,719 edges
- 2,698,241 connected components, of which 2,472,481 are already colorful
- largest connected component has 1,828 vertices and 14,403 edges
- solved optimally by data reduction + CLIQUE PARTITIONING algorithm in about 80 minutes
- 618,660 edges deleted, 434,849 inserted.

Random graph model



Greedy Heuristics (random instances)

	optimal	average error	max. error
move-based	25.8 %	4.9 %	38.7 %
merge-based	58.2 %	0.9 %	12.5 %

Clustering

Graph-based clustering

Find a partition of the vertices of a graph into clusters such that

- Vertices within a cluster have many connections;
- Vertices in different clusters have few connections.

Clustering

Graph-based clustering

Find a partition of the vertices of a graph into clusters such that

- Vertices within a cluster have many connections;
- Vertices in different clusters have few connections.

Definition ([Hartuv & Shamir '00])

A graph with n vertices is called *highly connected* if more than $n/2$ edges need to be deleted to make it disconnected.

Clustering

Graph-based clustering

Find a partition of the vertices of a graph into clusters such that

- Vertices within a cluster have many connections;
- Vertices in different clusters have few connections.

Definition ([Hartuv & Shamir ’00])

A graph with n vertices is called *highly connected* if more than $n/2$ edges need to be deleted to make it disconnected.

Lemma ([Chartrand ’66])

A graph with n vertices is highly connected iff each vertex has degree more than $n/2$.

Clustering algorithm

Min-cut algorithm [Hartuv & Shamir '00]

If the graph is highly connected, terminate; otherwise, delete the edges of a minimum cut and recurse on the two sides.

Clustering algorithm

Min-cut algorithm [Hartuv & Shamir '00]

If the graph is highly connected, terminate; otherwise, delete the edges of a minimum cut and recurse on the two sides.

Applications

- Clustering cDNA fingerprints;
- Finding complexes in protein–protein interaction (PPI) data;
- Grouping protein sequences hierarchically into superfamily and family clusters;
- Finding families of regulatory RNA structures.

Maximizing Edge Coverage

Highly Connected Deletion

Instance: An undirected graph.
Task: Delete a minimum number of edges such that each remaining connected component is highly connected.

Maximizing Edge Coverage

Highly Connected Deletion

Instance: An undirected graph.
Task: Delete a minimum number of edges such that each remaining connected component is highly connected.

Goal

Find optimal solutions for Highly Connected Deletion.

Maximizing Edge Coverage

Highly Connected Deletion

Instance: An undirected graph.
Task: Delete a minimum number of edges such that each remaining connected component is highly connected.

Goal

Find optimal solutions for Highly Connected Deletion.

Lemma

The min-cut algorithm can delete $\Omega(k^2)$ edges, where k is the optimal solution size.

Complexity

Theorem

Highly Connected Deletion is NP-hard even on 4-regular graphs.

Complexity

Theorem

Highly Connected Deletion is NP-hard even on 4-regular graphs.

Theorem

If the Exponential Time Hypothesis (ETH) is true, then Highly Connected Deletion cannot be solved in subexponential time (that is, $2^{o(k)} \cdot n^{O(1)}$ or $2^{o(n)} \cdot n^{O(1)}$ time).

Data reduction

Lemma

In a highly connected graph, if two vertices are connected by an edge, they have at least one common neighbor; otherwise, they have at least three common neighbors.

Data reduction

Lemma

In a highly connected graph, if two vertices are connected by an edge, they have at least one common neighbor; otherwise, they have at least three common neighbors.

Reduction rule

If there are two vertices that are connected by an edge but have no common neighbors, then delete the edge.

Data reduction

Reduction rule

If G contains a vertex set S such that

- $|S| \geqslant 4$,
- $G[S]$ is highly connected, and
- $|D(S)| \leqslant 0.3 \cdot \sqrt{|S|}$,

then remove S from G. Here, $D(S)$ is the size of the edge cut between S and the rest of the graph.

Data reduction

Reduction rule

If G contains a vertex set S such that

- $|S| \geqslant 4$,
- $G[S]$ is highly connected, and
- $|D(S)| \leqslant 0.3 \cdot \sqrt{|S|}$,

then remove S from G. Here, $D(S)$ is the size of the edge cut between S and the rest of the graph.

Theorem

Highly Connected Deletion admits a problem kernel with at most $10 \cdot k^{1.5}$ vertices, which can be computed in $O(n^2 \cdot mk \log n)$ time.

Fixed-parameter algorithm

Using a combination of kernelization and dynamic programming, we obtain:

Theorem

Highly Connected Deletion can be solved in $O(3^{4k} \cdot k^2 + n^2 mk \cdot \log n)$ *time.*

Column generation

Idea

Use a 0/1-variable to indicate that a particular cluster is in the solution, and successively add only those variables (“columns”) that are “needed”, that is, their introduction improves the objective.

PPI networks: data reduction

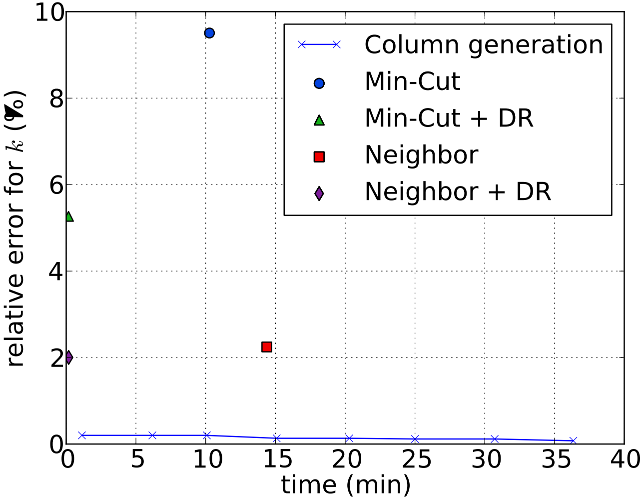
	n	m	Δk	Δk [%]	n'	m'
C. elegans phys.	157	153	100	92.6	11	38
C. elegans all	3613	6828	5204	80.1	373	1562
M. musculus phys.	4146	7097	5659	85.3	426	1339
M. musculus all	5252	9640	7609	84.8	595	1893
A. thaliana phys.	1872	2828	2057	83.1	187	619
A. thaliana all	5704	12627	8797	79.5	866	3323

n', m': size of largest connected component after data reduction

PPI networks: Column generation

- Using column generation, an solve optimally e. g. PPI network of *A. thaliana* with 5 704 vertices and 12 627 edges, in a few hours ($k = 12096$ edges deleted)
- Cannot solve network of *S. pombe* with 3 735 vertices and 51 620 edges

Heuristics (A. thaliana network)



FPT and ILP

Observations

- FPT algorithms give useful running time bounds and are often fast in practice
- ILP approaches are often even faster in practice, but do not have useful running time bounds
- Combining kernelization and ILPs works quite well