Iterative Compression: Some Case Studies

Hannes Moser

Institut für Informatik
Friedrich-Schiller-Universität Jena

Project ITKO (NI 369/5-1)
DFG Schwerpunktprogramm 1126 – Jahreskolloquium 2007

Parameterized Approach to Hard Problems

Exact algorithm: Exponential running time for NP-hard problems.

Parameterized approach

Try to confine the combinatorial explosion to a parameter k.

Fixed-Parameter Tractability

A problem is *fixed-parameter tractable* if it can be solved in $f(k) \cdot n^{O(1)}$ time.

Example

VERTEX COVER can be solved in time $O(1.28^k + k|V|)$.
k: size of the vertex cover

Iterative Compression Framework

Idea

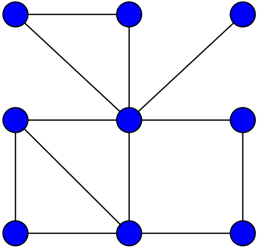
Use a *compression routine* iteratively:
Given a solution of size $k + 1$, compute a solution of size k, or prove that is does not exist.

[Reed, Smith, and Vetta, Operations Research Letters 32, 2004]

Example: Cluster Vertex Deletion (CVD)

Input: A graph $G = (V, E)$ and an integer $k > 0$.

Question: Is there a subset $S \subseteq V$ with $|S| \leq k$ such that every connected component of $G[V \setminus S]$ is a clique?

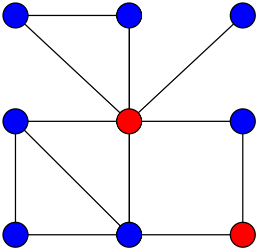


[Hüffner, PhD-thesis, 2007]

Example: Cluster Vertex Deletion (CVD)

Input: A graph $G = (V, E)$ and an integer $k > 0$.

Question: Is there a subset $S \subseteq V$ with $|S| \leq k$ such that every connected component of $G[V \setminus S]$ is a clique?

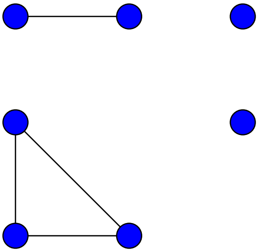


[Hüffner, PhD-thesis, 2007]

Example: Cluster Vertex Deletion (CVD)

Input: A graph $G = (V, E)$ and an integer $k > 0$.

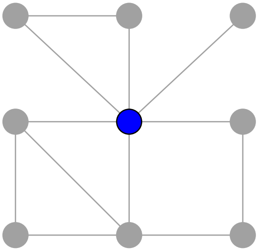
Question: Is there a subset $S \subseteq V$ with $|S| \leq k$ such that every connected component of $G[V \setminus S]$ is a clique?



[HÜFFNER, PhD-thesis, 2007]

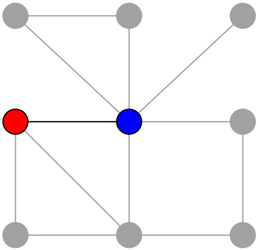
Iteration for Cluster Vertex Deletion

1. $V' := \emptyset$
2. $S := \emptyset$
3. While $G[V'] \neq G$
   3.1 Augment V' by adding a vertex v from $V \setminus V'$
   3.2 $S := S \cup \{v\}$
   3.3 $S := \text{CVD-COMPRESS}(G[V'], S)$
   3.4 If $|S| > k$ return “NO”
4. Return S



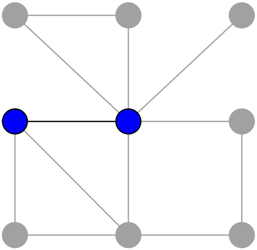
Iteration for Cluster Vertex Deletion

1. $V' := \emptyset$
2. $S := \emptyset$
3. While $G[V'] \neq G$
   3.1 Augment V' by adding a vertex v from $V \setminus V'$
   3.2 $S := S \cup \{v\}$
   3.3 $S := \text{CVD-Compress}(G[V'], S)$
   3.4 If $|S| > k$ return "NO"
4. Return S



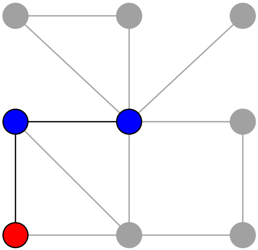
Iteration for Cluster Vertex Deletion

1. $V' := \emptyset$
2. $S := \emptyset$
3. While $G[V'] \neq G$
   3.1 Augment V' by adding a vertex v from $V \setminus V'$
   3.2 $S := S \cup \{v\}$
   3.3 $S := \textsc{CVD-Compress}(G[V'], S)$
   3.4 If $|S| > k$ return "NO"
4. Return S



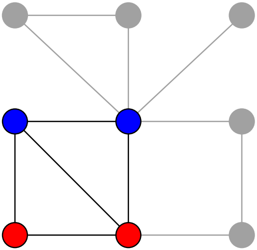
Iteration for Cluster Vertex Deletion

1. $V' := \emptyset$
2. $S := \emptyset$
3. While $G[V'] \neq G$
   3.1 Augment V' by adding a vertex v from $V \setminus V'$
   3.2 $S := S \cup \{v\}$
   3.3 $S := \textsc{CVD-Compress}(G[V'], S)$
   3.4 If $|S| > k$ return "NO"
4. Return S



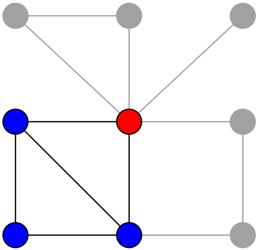
Iteration for Cluster Vertex Deletion

1. $V' := \emptyset$
2. $S := \emptyset$
3. While $G[V'] \neq G$
   3.1 Augment V' by adding a vertex v from $V \setminus V'$
   3.2 $S := S \cup \{v\}$
   3.3 $S := \textsc{CVD-Compress}(G[V'], S)$
   3.4 If $|S| > k$ return "NO"
4. Return S



Iteration for Cluster Vertex Deletion

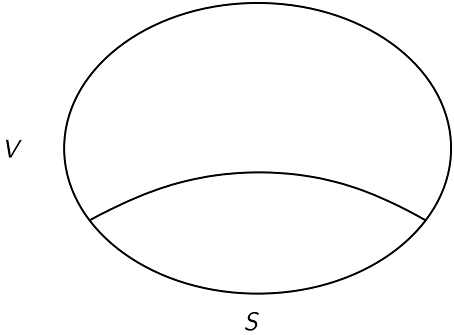
1. $V' := \emptyset$
2. $S := \emptyset$
3. While $G[V'] \neq G$
   3.1 Augment V' by adding a vertex v from $V \setminus V'$
   3.2 $S := S \cup \{v\}$
   3.3 $S := \text{CVD-Compress}(G[V'], S)$
   3.4 If $|S| > k$ return "NO"
4. Return S



Compression for Cluster Vertex Deletion

Approach

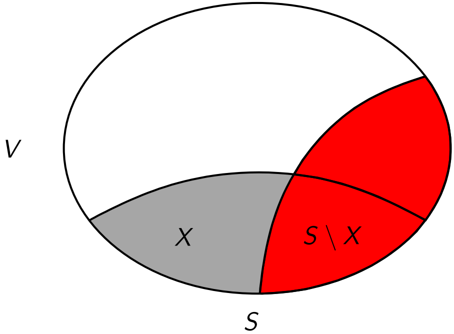
Try all $2^{|S|}$ partitions of S into a part to keep in the new solution and a part to exchange.



Compression for Cluster Vertex Deletion

Approach

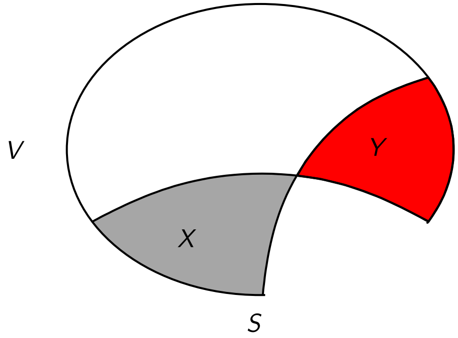
Try all $2^{|S|}$ partitions of S into a part to keep in the new solution and a part to exchange.



Compression for Cluster Vertex Deletion

Approach

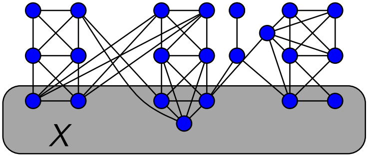
Try all $2^{|S|}$ partitions of S into a part to keep in the new solution and a part to exchange.



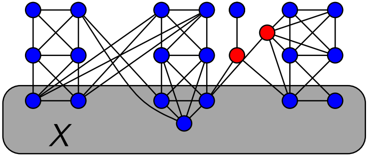
Simplified Problem

Given a solution X, compute a smaller *disjoint* solution Y.

Compression for Cluster Vertex Deletion

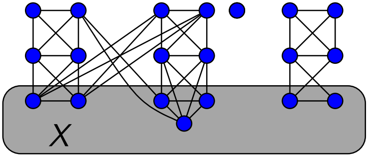


Compression for Cluster Vertex Deletion



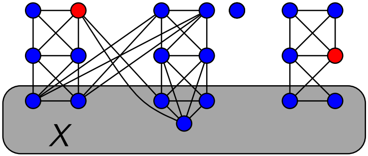
- Delete all vertices in $V \setminus X$ that are adjacent to more than one cluster in X.

Compression for Cluster Vertex Deletion



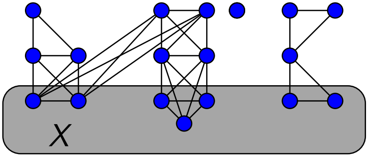
- Delete all vertices in $V \setminus X$ that are adjacent to more than one cluster in X.

Compression for Cluster Vertex Deletion



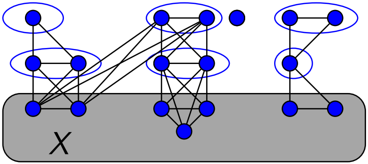
- Delete all vertices in $V \setminus X$ that are adjacent to more than one cluster in X.
- Delete vertices in $V \setminus X$ that are not adjacent to all vertices of a cluster in X.

Compression for Cluster Vertex Deletion



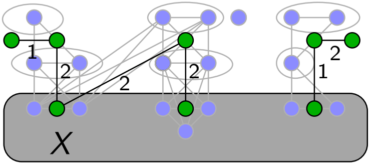
- Delete all vertices in $V \setminus X$ that are adjacent to more than one cluster in X.
- Delete vertices in $V \setminus X$ that are not adjacent to all vertices of a cluster in X.

Compression for Cluster Vertex Deletion



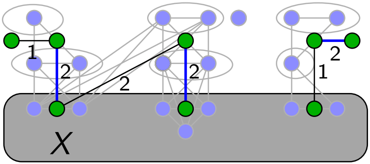
- Delete all vertices in $V \setminus X$ that are adjacent to more than one cluster in X.
- Delete vertices in $V \setminus X$ that are not adjacent to all vertices of a cluster in X.
- Classify the vertices in each cluster in $V \setminus X$ by their neighboring clusters in X.

Compression for Cluster Vertex Deletion



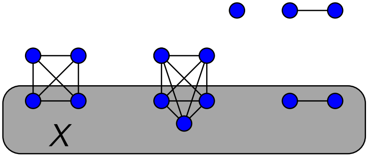
- Delete all vertices in $V \setminus X$ that are adjacent to more than one cluster in X.
- Delete vertices in $V \setminus X$ that are not adjacent to all vertices of a cluster in X.
- Classify the vertices in each cluster in $V \setminus X$ by their neighboring clusters in X.
- Generate a bipartite dependency graph; a maximum-weight matching represents an optimal solution.

Compression for Cluster Vertex Deletion



- Delete all vertices in $V \setminus X$ that are adjacent to more than one cluster in X.
- Delete vertices in $V \setminus X$ that are not adjacent to all vertices of a cluster in X.
- Classify the vertices in each cluster in $V \setminus X$ by their neighboring clusters in X.
- Generate a bipartite dependency graph; a maximum-weight matching represents an optimal solution.

Compression for Cluster Vertex Deletion



- Delete all vertices in $V \setminus X$ that are adjacent to more than one cluster in X.
- Delete vertices in $V \setminus X$ that are not adjacent to all vertices of a cluster in X.
- Classify the vertices in each cluster in $V \setminus X$ by their neighboring clusters in X.
- Generate a bipartite dependency graph; a maximum-weight matching represents an optimal solution.

Iterative Compression for Cluster Vertex Deletion - Analysis

- The iteration calls the compression up to n times.
- The compression tries all $O(2^k)$ partitions of a given solution.
- The remaining task to compute a *disjoint* solution can be performed in polynomial time.

Iterative Compression for Cluster Vertex Deletion - Analysis

- The iteration calls the compression up to n times.
- The compression tries all $O(2^k)$ partitions of a given solution.
- The remaining task to compute a *disjoint* solution can be performed in polynomial time.

Overall running time

- $O(2^k \cdot n^{O(1)})$
- Best known running time so far $O(2.08^k \cdot n^{O(1)})$ (via reduction to 3-Hitting Set, which can be solved by a rather involved algorithm [Wahlström, PhD-thesis, 2007])

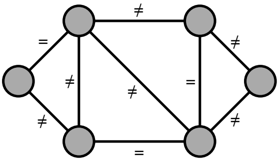
Applications of Iterative Compression

- Graph Bipartization $O(3^k kmn)$
  [Reed, Smith, and Vetta, Operations Research Letters 32, 2004]
- Edge Bipartization $O(2^k m^2)$
  [Guo, Gramm, Hüffner, Niedermeier, and Wernicke, JCSS 72, 2006]
- Feedback Vertex Set $O(c^k m)$
  [Dehne, Fellows, Langston, Rosamond, and Stevens, COCOON 2005]
  [Guo, Gramm, Hüffner, Niedermeier, and Wernicke, JCSS 72, 2006]
  [Chen, Fomin, Liu, Lu, and Villanger, WADS 2007]
- Feedback Vertex Set in Tournaments $O(2^k n^2(\lg n + k))$
  [Dom, Guo, Hüffner, Niedermeier, and Truss, CIAC 2006]
- Chordal Deletion
  [Marx, WG 2006]
- Implementation of Graph Bipartization $O(3^k mn)$
  [Hüffner, WEA 2005]
- Experimental results for Signed Graph Balancing
  [Hüffer, Betzler, and Niedermeier, WEA 2007]

Recent Application: Signed Graph Balancing

Definition

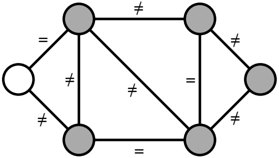
A graph with edges labeled by $=$ or $\neq$ (signed graph) is balanced if the vertices can be colored with two colors such that the relation on each edge holds.



Recent Application: Signed Graph Balancing

Definition

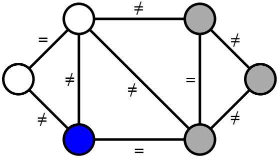
A graph with edges labeled by $=$ or $\neq$ (signed graph) is balanced if the vertices can be colored with two colors such that the relation on each edge holds.



Recent Application: Signed Graph Balancing

Definition

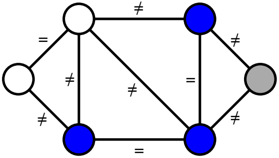
A graph with edges labeled by $=$ or $\neq$ (signed graph) is balanced if the vertices can be colored with two colors such that the relation on each edge holds.



Recent Application: Signed Graph Balancing

Definition

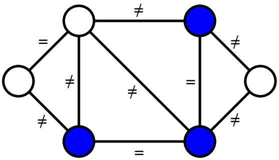
A graph with edges labeled by $=$ or $\neq$ (signed graph) is balanced if the vertices can be colored with two colors such that the relation on each edge holds.



Recent Application: Signed Graph Balancing

Definition

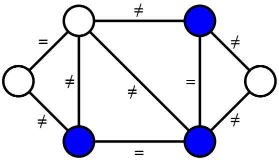
A graph with edges labeled by $=$ or $\neq$ (signed graph) is balanced if the vertices can be colored with two colors such that the relation on each edge holds.



Recent Application: Signed Graph Balancing

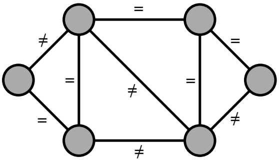
Definition

A graph with edges labeled by $=$ or $\neq$ (signed graph) is balanced if the vertices can be colored with two colors such that the relation on each edge holds.



Task

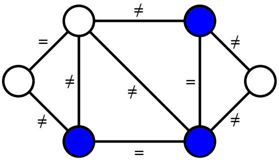
Find a minimum number of edges whose deletion makes the signed graph balanced.



Recent Application: Signed Graph Balancing

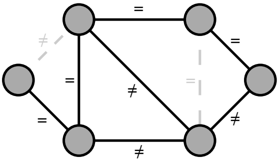
Definition

A graph with edges labeled by $=$ or $\neq$ (signed graph) is balanced if the vertices can be colored with two colors such that the relation on each edge holds.



Task

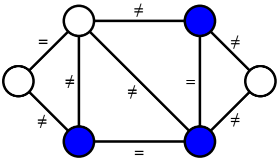
Find a minimum number of edges whose deletion makes the signed graph balanced.



Recent Application: Signed Graph Balancing

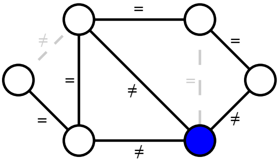
Definition

A graph with edges labeled by $=$ or $\neq$ (signed graph) is balanced if the vertices can be colored with two colors such that the relation on each edge holds.



Task

Find a minimum number of edges whose deletion makes the signed graph balanced.



Applications of Balanced Subgraph

- Balance in social networks
  [Harary, Mich. Math. J. 1953]
- Portfolio risk analysis
  [Harary et al., IMA J. Manag. Math. 2002]
- Minimum energy state of magnetic materials (spin glasses)
  [Kasteleyn, J. Math. Phys. 1963]
- Stability of fullerenes
  [Došlić&Vikičević, Discr. Appl. Math. 2007]
- Integrated circuit design
  [Chiang et al., IEEE Trans. CAD of IC&Sys. 2007]
- "Monotone subsystems" in biological networks
  [DasGupta et al., WEA 2006]

Signed Graph Balancing: Experimental Results

			Approximation [DasGupta et al., WEA 2006]			Exact alg. [Hüffner et al., WEA 2007]	
Data set	n	m	$k \geq$	$k \leq$	t [min]	k	t [min]
EGFR	330	855	196	219	7	210	108
Yeast	690	1082	0	43	77	41	1
Macr.	678	1582	218	383	44	374	1

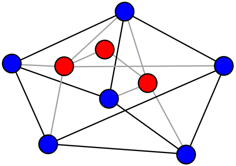
- A real-world network with 688 vertices and 2208 edges could not be solved.

When is Iterative Compression Applicable?

Two representative problems

Input: Graph $G = (V, E)$, parameter $k > 0$, integer constant $r \geq 0$.

r-Regular Deletion



Question: Is there a subset $S \subseteq V$, $|S| \leq k$, such that every vertex in $G[V \setminus S]$ has degree *exactly* r?

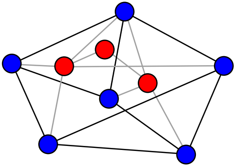
When is Iterative Compression Applicable?

Two representative problems

Input: Graph $G = (V, E)$, parameter $k > 0$, integer constant $r \geq 0$.

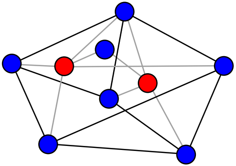
r-Regular Deletion

Question: Is there a subset $S \subseteq V$, $|S| \leq k$, such that every vertex in $G[V \setminus S]$ has degree *exactly* r?

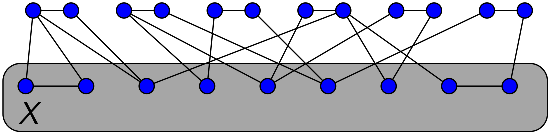


Maxdeg-r-Deletion

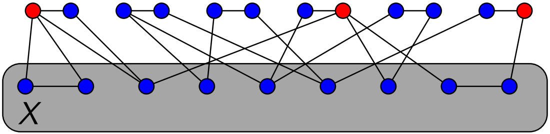
Question: Is there a subset $S \subseteq V$, $|S| \leq k$, such that every vertex in $G[V \setminus S]$ has degree *at most* r?



Compression for 1-Regular Deletion

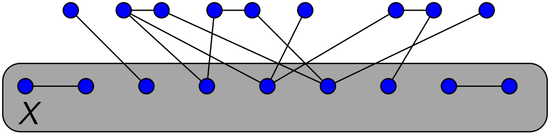


Compression for 1-Regular Deletion



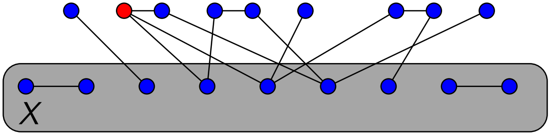
- For every edge in X remove all its neighbors in $V \setminus X$.

Compression for 1-Regular Deletion



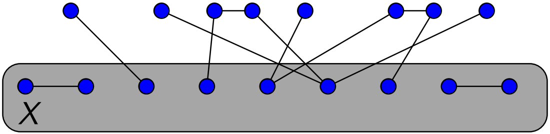
- For every edge in X remove all its neighbors in $V \setminus X$.

Compression for 1-Regular Deletion



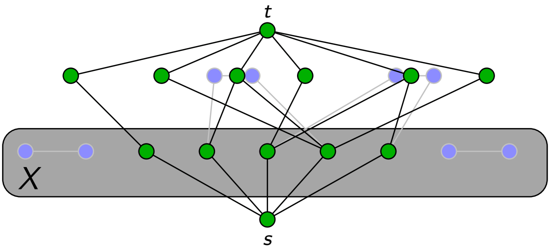
- For every edge in X remove all its neighbors in $V \setminus X$.
- Remove all vertices in $V \setminus X$ with more than one neighbor in X.

Compression for 1-Regular Deletion



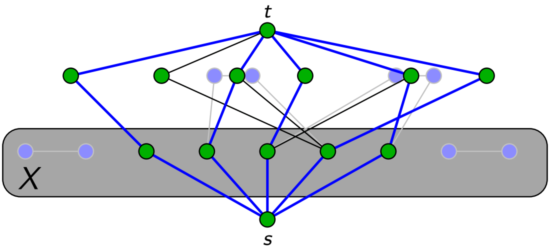
- For every edge in X remove all its neighbors in $V \setminus X$.
- Remove all vertices in $V \setminus X$ with more than one neighbor in X.

Compression for 1-Regular Deletion



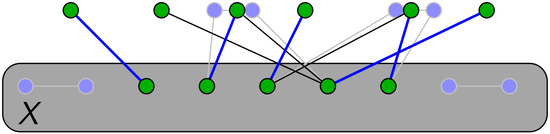
- For every edge in X remove all its neighbors in $V \setminus X$.
- Remove all vertices in $V \setminus X$ with more than one neighbor in X.
- Build a dependency graph; a max-flow solution represents an optimal solution.

Compression for 1-REGULAR DELETION



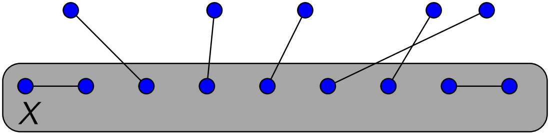
- For every edge in X remove all its neighbors in $V \setminus X$.
- Remove all vertices in $V \setminus X$ with more than one neighbor in X.
- Build a dependency graph; a max-flow solution represents an optimal solution.

Compression for 1-Regular Deletion



- For every edge in X remove all its neighbors in $V \setminus X$.
- Remove all vertices in $V \setminus X$ with more than one neighbor in X.
- Build a dependency graph; a max-flow solution represents an optimal solution.

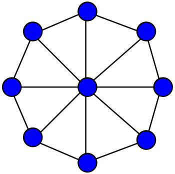
Compression for 1-Regular Deletion



- For every edge in X remove all its neighbors in $V \setminus X$.
- Remove all vertices in $V \setminus X$ with more than one neighbor in X.
- Build a dependency graph; a max-flow solution represents an optimal solution.

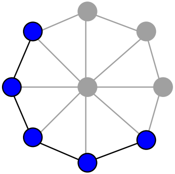
2-Regular Deletion

Iteration becomes difficult



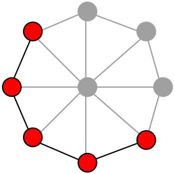
2-Regular Deletion

Iteration becomes difficult



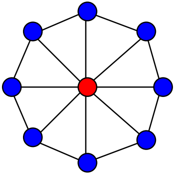
2-Regular Deletion

Iteration becomes difficult



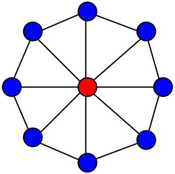
2-Regular Deletion

Iteration becomes difficult



2-REGULAR DELETION

Iteration becomes difficult



Compression becomes hard

Theorem

*Finding a smaller disjoint solution for 2-*REGULAR DELETION *is NP-hard.*

Proof approach: Reduction from 1-in-3-SAT.

Overview

	search tree	find disj. sol.	iterative compr.
1-Regular Del. maxdeg-1-del.	$O(3^k \cdot n^{O(1)})$	P (Maxflow) P (Matching)	$O(2^k \cdot n^{O(1)})$
2-Regular Del. maxdeg-2-del.	$O(4^k \cdot n^{O(1)})$	NP-hard ?	? $\surd$

Discussion

Advantages

- Problem simplification: Improve a solution instead of computing an optimal solution directly.
- Derivation of structural information.

Discussion

Advantages

- Problem simplification: Improve a solution instead of computing an optimal solution directly.
- Derivation of structural information.

Drawbacks

- "Bottleneck" 2^k (try all partitions).
- Design of compression routine challenging.

Discussion

Advantages

- Problem simplification: Improve a solution instead of computing an optimal solution directly.
- Derivation of structural information.

Drawbacks

- "Bottleneck" 2^k (try all partitions).
- Design of compression routine challenging.

Ongoing Work

- Characterize problems amenable to iterative compression.
- Combination with other techniques (e.g., approximation).

Thank you!