Iterative Compression for Solving Hard Network Problems

Hannes Moser

Institut für Informatik
Friedrich-Schiller-Universität Jena

Project ITKO (NI 369/5)
DFG Schwerpunktprogramm 1126 – Jahreskolloquium 2006

Parameterized Approach to Hard Problems

We want an exact algorithm, but that implies exponential runtime.

Parameterized Approach to Hard Problems

We want an exact algorithm, but that implies exponential runtime.

Parameterized approach

Try to confine the combinatorial explosion to a parameter k.

Parameterized Approach to Hard Problems

We want an exact algorithm, but that implies exponential runtime.

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 $O(f(k) \cdot n^{O(1)})$ time.

Parameterized Approach to Hard Problems

We want an exact algorithm, but that implies exponential runtime.

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 $O(f(k) \cdot n^{O(1)})$ time.

Example

Vertex Cover can be solved in time $O(1.2852^k + k|V|)$.
k: size of the vertex cover

Techniques to Show Fixed-Parameter Tractability

Established Techniques

- Kernelizations
- Depth-bounded search trees
- Dynamic Programming
- Tree Decompositions
- ...

Recent Approach

- Iterative compression

Iterative Compression

Core Idea

Inductive approach: Compute a solution for a problem instance using the information provided by a solution for a smaller instance.

Iterative Compression

Core Idea

Inductive approach: Compute a solution for a problem instance using the information provided by a solution for a smaller instance.

In terms of a minimization problem on graphs

Compute a solution X for a problem instance (G, k) using the information provided by a solution X' for a subinstance $(G - v, k)$.

Iterative Compression

Core Idea

Inductive approach: Compute a solution for a problem instance using the information provided by a solution for a smaller instance.

In terms of a minimization problem on graphs

Compute a solution X for a problem instance (G, k) using the information provided by a solution X' for a subinstance $(G - v, k)$.

Example for this talk

FEEDBACK VERTEX SET IN TOURNAMENTS.

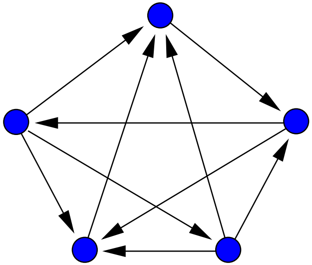
[DOM, GUO, HÜFFNER, NIEDERMEIER, AND TRUSS, CIAC 2006]

Feedback Vertex Set in Tournaments

Definition (Feedback Vertex Set in Tournaments)

Input: Tournament $G = (V, E)$, integer $k \geq 0$.

Output: Is there a subset $X \subseteq V$ of at most k vertices such that $G[V \setminus X]$ has no cycles?

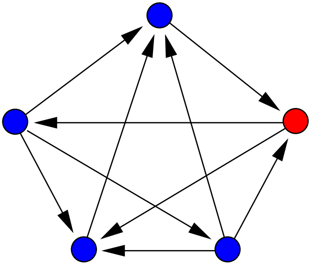


Feedback Vertex Set in Tournaments

Definition (Feedback Vertex Set in Tournaments)

Input: Tournament $G = (V, E)$, integer $k \geq 0$.

Output: Is there a subset $X \subseteq V$ of at most k vertices such that $G[V \setminus X]$ has no cycles?

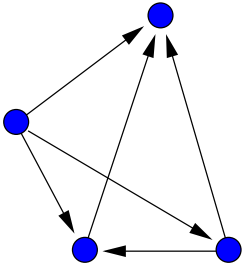


Feedback Vertex Set in Tournaments

Definition (Feedback Vertex Set in Tournaments)

Input: Tournament $G = (V, E)$, integer $k \geq 0$.

Output: Is there a subset $X \subseteq V$ of at most k vertices such that $G[V \setminus X]$ has no cycles?

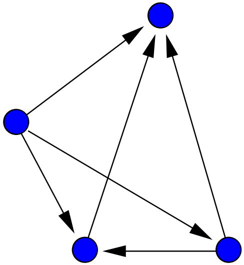


Feedback Vertex Set in Tournaments

Definition (Feedback Vertex Set in Tournaments)

Input: Tournament $G = (V, E)$, integer $k \geq 0$.

Output: Is there a subset $X \subseteq V$ of at most k vertices such that $G[V \setminus X]$ has no cycles?



Feedback Vertex Set in Tournaments is NP-complete

[Speckenmeyer, WG 1989]

Iterative Compression Framework

Idea

Use a *compression routine* iteratively: Given a solution of size $k+1$, compute a solution of size k.

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

Iterative Compression Framework

Idea

Use a *compression routine* iteratively: Given a solution of size $k+1$, compute a solution of size k.

[REED, SMITH, AND VETTA, Operations Research Letters 32, 2004]

Iterative Compression Framework for Feedback Vertex Set in Tournaments

1. Start with an empty graph G' and an empty solution X.
2. For each vertex v in G:
   2.1 Add v to G' and to X.
   2.2 Use the compression routine to compress X.
   2.3 If $|X| > k$, then return "NO".

Iterative Compression Framework

Idea

Use a *compression routine* iteratively: Given a solution of size $k+1$, compute a solution of size k.

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

Iterative Compression Framework for Feedback Vertex Set in Tournaments

1. Start with an empty graph G' and an empty solution X.
2. For each vertex v in G:
   2.1 Add v to G' and to X.
   2.2 Use the compression routine to compress X.
   2.3 If $|X| > k$, then return "NO".

Invariant during the loop:
X is a solution of size at most k for the current graph G'.

Compression Routine (1)

Task

Given a solution X of size $k+1$, compute a solution X' of size k.

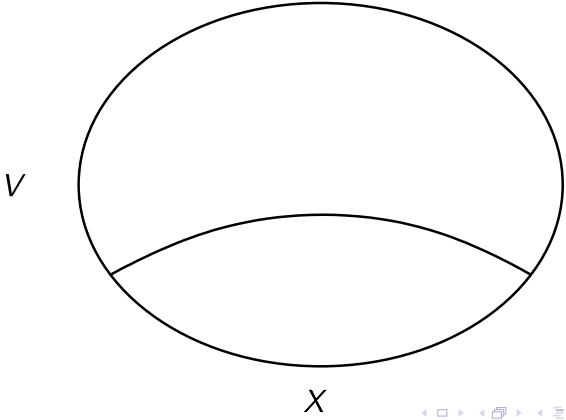
Compression Routine (1)

Task

Given a solution X of size $k+1$, compute a solution X' of size k.

Approach

Try all $2^{|X|}$ partitions of X into a part to exchange (S) and a part to keep ($X \setminus S$).



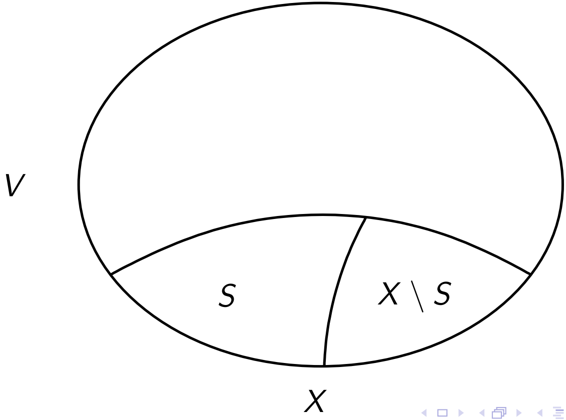
Compression Routine (1)

Task

Given a solution X of size $k+1$, compute a solution X' of size k.

Approach

Try all $2^{|X|}$ partitions of X into a part to exchange (S) and a part to keep ($X \setminus S$).



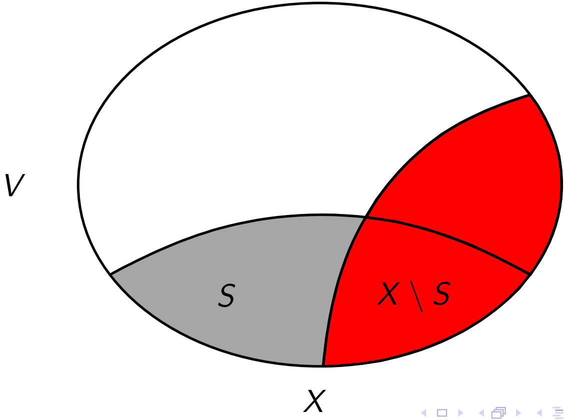
Compression Routine (1)

Task

Given a solution X of size $k+1$, compute a solution X' of size k.

Approach

Try all $2^{|X|}$ partitions of X into a part to exchange (S) and a part to keep ($X \setminus S$).



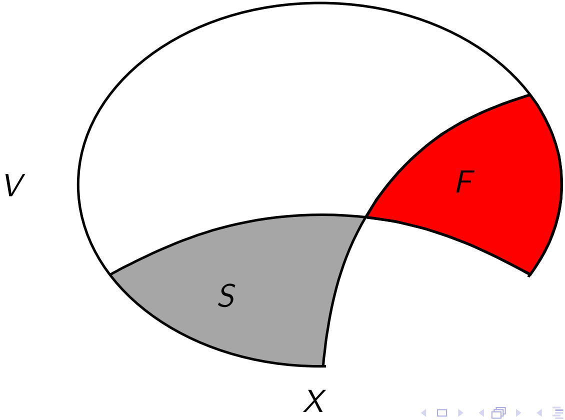
Compression Routine (1)

Task

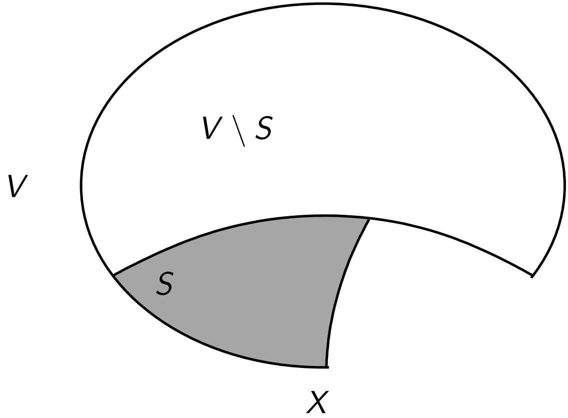
Given a solution X of size $k+1$, compute a solution X' of size k.

Approach

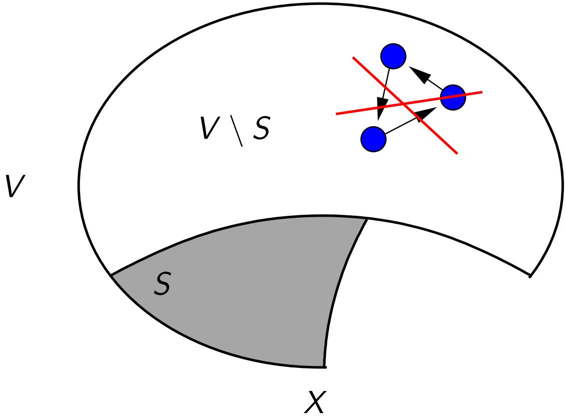
Try all $2^{|X|}$ partitions of X into a part to exchange (S) and a part to keep ($X \setminus S$).



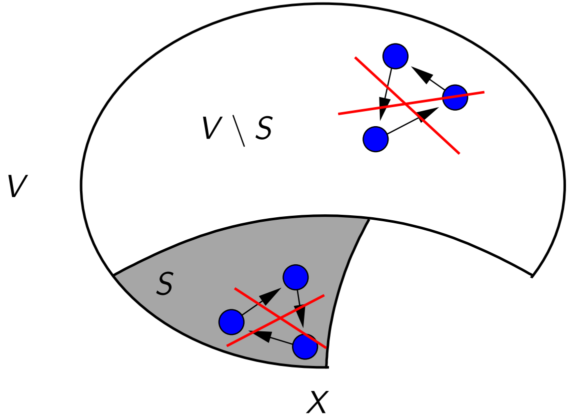
Compression Routine (2)



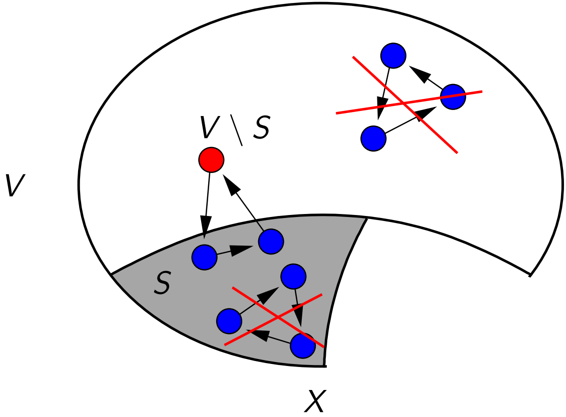
Compression Routine (2)



Compression Routine (2)



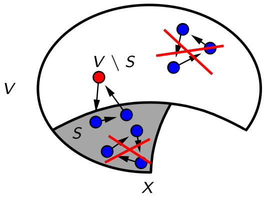
Compression Routine (2)



(Lemma: A tournament contains a cycle iff it contains a triangle.)

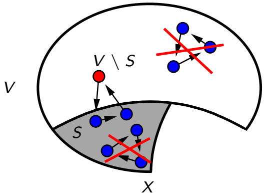
Topological Sorts

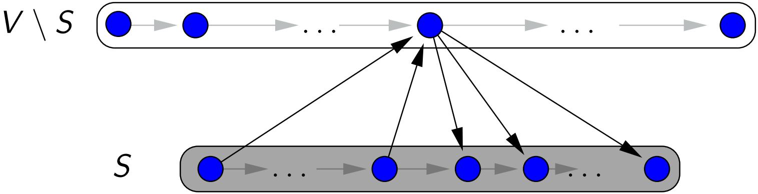
- $G[S]$ is cycle-free $\Rightarrow$
  S has topological sort $s_1, \ldots, s_{|S|}$.
- $G[V \setminus S]$ is cycle-free $\Rightarrow$
  $V \setminus S$ has topological sort.



Topological Sorts

- $G[S]$ is cycle-free $\Rightarrow$
  S has topological sort $s_1, \ldots, s_{|S|}$.
- $G[V \setminus S]$ is cycle-free $\Rightarrow$
  $V \setminus S$ has topological sort.



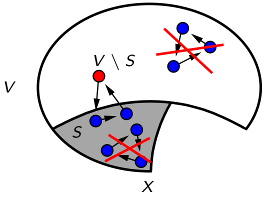


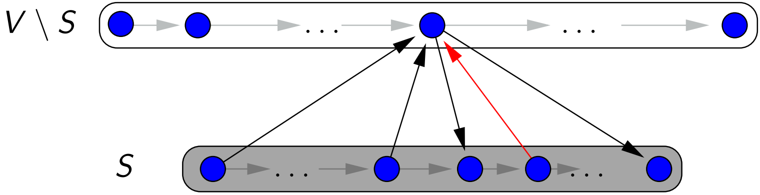
Goal: Insert a maximum subset of $V \setminus S$ into $s_1, \ldots, s_{|S|}$.

Observation: Every vertex v of $V \setminus S$ has a "natural position" $p(v)$ relative to the vertices of S.

Topological Sorts

- $G[S]$ is cycle-free $\Rightarrow$
  S has topological sort $s_1, \ldots, s_{|S|}$.
- $G[V \setminus S]$ is cycle-free $\Rightarrow$
  $V \setminus S$ has topological sort.

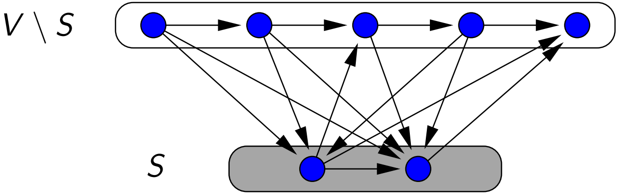




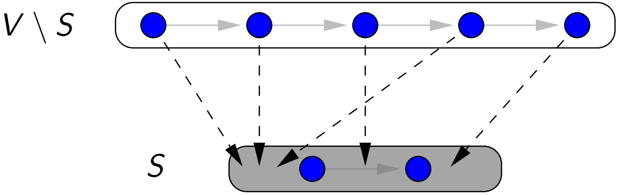
Goal: Insert a maximum subset of $V \setminus S$ into $s_1, \ldots, s_{|S|}$.

Observation: Every vertex v of $V \setminus S$ has a “natural position” $p(v)$ relative to the vertices of S.

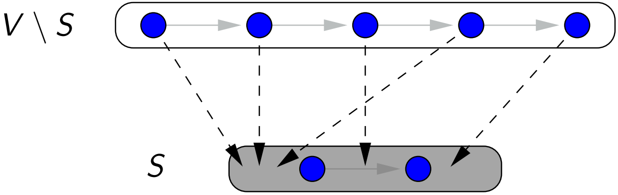
Longest Common Subsequence



Longest Common Subsequence



Longest Common Subsequence



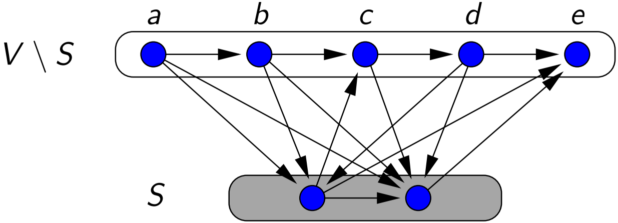
The vertices $(V \setminus S) \setminus F$ must be sortable in such a way that

- the topological sort of $V \setminus S$ is preserved, and
- the sort of $V \setminus S$ by "natural position" p is preserved.

$\Rightarrow$ Search for the longest common subsequence of both sorts.

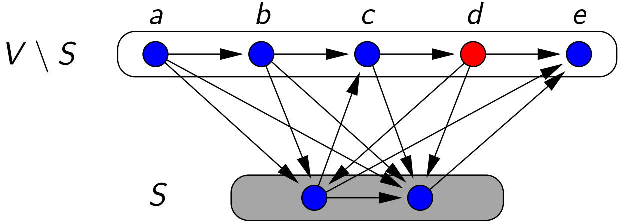
Example

- $V \setminus S$ sorted topologically: *abcde*
- $V \setminus S$ sorted by p: *abdce*
- A longest common subsequence is *abce*



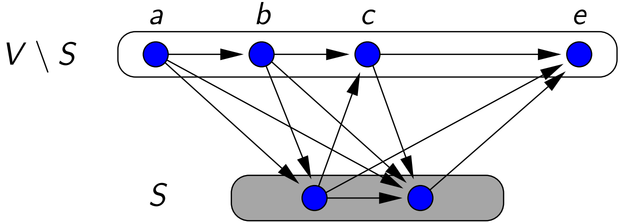
Example

- $V \setminus S$ sorted topologically: *abcde*
- $V \setminus S$ sorted by p: *abdce*
- A longest common subsequence is *abce*



Example

- $V \setminus S$ sorted topologically: *abcde*
- $V \setminus S$ sorted by p: *abdce*
- A longest common subsequence is *abce*



Overall Running Time

Time consumption:

- n iterations of the compression routine;
- 2^{k+1} partitions per iteration;
- time $O(n \cdot k)$ for destroying triangles, time $O(n \log n)$ for sorting vertices and finding the longest common subsequence.

Running time for solving FEEDBACK VERTEX SET IN TOURNAMENTS:

$$O(2^k \cdot n^2(\log n + k))$$

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, WADS 2005]
- Feedback Vertex Set $O(c^k m)$
  [Dehne, Fellows, Langston, Rosamond, and Stevens, COCOON 2005]
  [Guo, Gramm, Hüffner, Niedermeier, and Wernicke, WADS 2005]
- 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]

Discussion

Advantages

- The problem becomes easier: We improve a solution instead of computing an optimal solution directly.
- Optimize results from approximation algorithms and heuristics.

Discussion

Advantages

- The problem becomes easier: We improve a solution instead of computing an optimal solution directly.
- Optimize results from approximation algorithms and heuristics.

Drawbacks

- "Bottleneck" 2^k.
- The design of the compression routine still can be difficult.

Discussion

Advantages

- The problem becomes easier: We improve a solution instead of computing an optimal solution directly.
- Optimize results from approximation algorithms and heuristics.

Drawbacks

- "Bottleneck" 2^k.
- The design of the compression routine still can be difficult.

Future Work

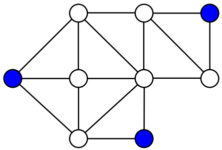
- Characterize problems amenable to solution compression.
- How can iterative compression be combined with other techniques?
- Apply to maximization problems.

A Candidate Problem for Future Research

Minimum Vertex Multiway Cut

Input: A graph $G = (V, E)$, a set of terminals $T \subseteq V$, and an integer $k \geq 0$.

Output: Is there a subset $X \subseteq V$ of at most k vertices such that no two vertices of T belong to the same connected component of $G[V \setminus X]$?

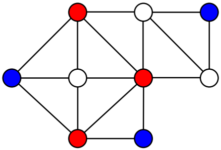


A Candidate Problem for Future Research

MINIMUM VERTEX MULTIWAY CUT

Input: A graph $G = (V, E)$, a set of terminals $T \subseteq V$, and an integer $k \geq 0$.

Output: Is there a subset $X \subseteq V$ of at most k vertices such that no two vertices of T belong to the same connected component of $G[V \setminus X]$?

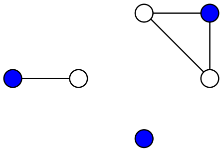


A Candidate Problem for Future Research

Minimum Vertex Multiway Cut

Input: A graph $G = (V, E)$, a set of terminals $T \subseteq V$, and an integer $k \geq 0$.

Output: Is there a subset $X \subseteq V$ of at most k vertices such that no two vertices of T belong to the same connected component of $G[V \setminus X]$?

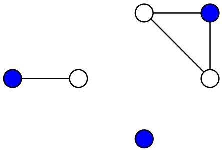


A Candidate Problem for Future Research

MINIMUM VERTEX MULTIWAY CUT

Input: A graph $G = (V, E)$, a set of terminals $T \subseteq V$, and an integer $k \geq 0$.

Output: Is there a subset $X \subseteq V$ of at most k vertices such that no two vertices of T belong to the same connected component of $G[V \setminus X]$?



Known Results

- NP-complete [CUNNINGHAM, DIMACS Series in DM and TCS, vol. 5, 1991]
- Fixed-parameter tractable $O(c^{k^3} \cdot \text{poly}(n))$ [MARX, IWPEC 2004]

Thank you!