Fixed-Parameter Algorithms, IA166

Sebastian Ordyniak

Faculty of Informatics
Masaryk University Brno

Spring Semester 2013

Outline

Iterative Compression

Motivation

- an easy but surprisingly powerful trick
- Most useful for deletion problems, i.e., delete k things to achieve some property
- like color coding, iterative compression comes for free;

A Simple Example: Vertex Cover

k-VERTEX COVER	**Parameter:** k

Input: A graph G and an integer k.
Question: Does G have a vertex cover of size at most k?

Idea: Reduce the problem to an easier compression version of the problem.

k-VERTEX COVER COMPRESSION	**Parameter:** k

Input: A graph G, an integer k, and a **vertex cover** C **of size at most** $k+1$.
Question: Does G have a vertex cover of size at most k?

A Simple Example: Vertex Cover

Idea: Reduce the problem to an easier compression version of the problem.

k-VERTEX COVER COMPRESSION	**Parameter:** k

Input: A graph G, an integer k, and a **vertex cover** C **of size at most** $k+1$.
Question: Does G have a vertex cover of size at most k?

There are 2 questions remaining:

- How to solve the compression problem?
- How to reduce k-VERTEX COVER to the compression problem?

A Simple Example: Vertex Cover

k-VERTEX COVER COMPRESSION — **Parameter: k**

Input: A graph G, an integer k, and a vertex cover C of size at most $k+1$.
Question: Does G have a vertex cover of size at most k?

An Algorithm for k-VERTEX COVER COMPRESSION

For every $C_{\text{KEEP}} \subseteq C$ *do*
  $C_{\text{REM}} := C \setminus C_{\text{KEEP}}$;
  If $G[C_{\text{REM}}]$ *contains no edges then*
    $C_{\text{NEW}} := N[C_{\text{REM}}] \setminus C$;
    If $|C_{\text{NEW}}| + |C_{\text{KEEP}}| \leq k$ *then*
      return $C_{\text{NEW}} \cup C_{\text{KEEP}}$;
return NO;

A Simple Example: Vertex Cover

k-VERTEX COVER COMPRESSION **Parameter: k**

Input: A graph G, an integer k, and a vertex cover C of size at most $k+1$.
Question: Does G have a vertex cover of size at most k?

Proof of Correctness of the Algorithm (sketch):

It is straightforward to check that if the algorithm returns a set then this set is a vertex cover of G of size at most k.
On the other hand any vertex cover C' of size at most k must contain $N[C \setminus C'] \setminus C$ and hence if a solution exists it is found by the algorithm!

A Simple Example: Vertex Cover

Theorem

k-VERTEX COVER COMPRESSION can be solved in time $O(2^k n^{O(1)})$.

How can we use the compression problem to solve vertex cover?

Start with the empty graph and add vertices one by one

A Simple Example: Vertex Cover

An Algorithm for k-Vertex Cover

$C := \emptyset$;
$V := \emptyset$;
For every $v \in V(G)$ do
  $V := V \cup \{v\}$;
  if C is not a vertex cover for $G[V]$ then
    $C := C \cup \{v\}$;
  if $|C| > k$ then;
    if k-VCC$(G[V], C, k)$ is a NO*-instance then*
      return No*;*
    $C := k$-VCC$(G[V], C, k)$;
return Yes*;*

A Simple Example: Vertex Cover

An Algorithm for k-VERTEX COVER

k-VERTEX COVER can be solved in time $O(2^k n^{O(1)})$.

General Iterative Compression

We can use the approach for any minimization problem on instances G that have an integer objective value and where we can construct a sequence $G_1, \ldots, G_n$ of polynomial length with $G_n = G$ and:

(1) A k-solution for G_1 exists and can be found in polynomial time.

(2) If G_i has a k-solution then G_{i+1} has a $k+1$-solution, which can be found in polynomial time.

(3) If G_i has no k-solution then G has no k-solution.

(4) If a $(k+1)$-solution S for G_{i+1} is given, then there is an FPT algorithm for parameter k that decides whether G_{i+1} has a k-solution (The compression step).

General Iterative Compression

For problems satisfying the properties of the previous slide, the following algorithm is an FPT-algorithm for parameter k that decides whether a k-solution exists for G:

The General Algorithm for Iterative Compression

Let S_1 *be a* k*-solution for* G_1*;*
For $i = 1$ *to* $i = n - 1$ *do;*
  Use S_i *to construct a* $(k+1)$*-solution* S_{i+1} *for* G_{i+1}*;*
  if $\mathrm{COMP}(G_{i+1}, S_{i+1}, k)$ *is a* NO*-instance then*
    return NO*;*
  $S_{i+1} := \mathrm{COMP}(G_{i+1}, S_{i+1}, k)$*;*
return YES*;*

Example: Graph Bipartisation

k-GRAPH BIPARTISATION **Parameter: k**

Input: A graph G and an integer k.
Question: Is there an $S \subseteq V(G)$ with $|S| \leq k$ such that $G \setminus S$ is bipartite?

- Standard example for the use of iterative compression.
- Very hard to tackle without iterative compression.

Example: Graph Bipartisation

Using the sequence $G_i := G[v_1, \dots, v_i]$ for an arbitrary ordering $v_1, \dots, v_{|V(G)|}$ of the vertices of G we obtain:

(1) $S_1 := \emptyset$ is a bipartization of G_1.

(2) If S_i is a k-bipartization for G_i then $S_{i+1} := S_i \cup \{v_{i+1}\}$ is a $(k+1)$-bipartization for G_{i+1}.

(3) If G_i has no k-bipartization then G has no k-bipartization.

(4) FPT-algorithm for compression version????

Hence, we only need to find an FPT-algorithm for the compression version of the problem!

Example: Graph Bipartisation

Using the sequence $G_i := G[v_1, \ldots, v_i]$ for an arbitrary ordering $v_1, \ldots, v_{|V(G)|}$ of the vertices of G we obtain:

(1) $S_1 := \emptyset$ is a bipartization of G_1.
(2) If S_i is a k-bipartization for G_i then $S_{i+1} := S_i \cup \{v_{i+1}\}$ is a $(k+1)$-bipartization for G_{i+1}.
(3) If G_i has no k-bipartization then G has no k-bipartization.
(4) FPT-algorithm for compression version????

Hence, we only need to find an FPT-algorithm for the compression version of the problem!

Example: Graph Bipartisation

k-GRAPH BIPARTISATION COMPRESSION **Parameter: k**

Input: A graph G, an integer k, and a $S \subseteq V(G)$ with $|S| \leq k+1$ s.t. $G \setminus S$ is bipartite.
Question: Is there an $S' \subseteq V(G)$ with $|S'| \leq k$ such that $G \setminus S'$ is bipartite?

Question

How to solve this problem?

Example: Graph Bipartisation

Answer

- Guess the intersection S_{KEEP} of an optimal solution with S.
- Then the vertices in $S_{\text{REM}} := S \setminus S_{\text{KEEP}}$ are not part of an optimal solution.
- Guess a bipartition $\{A, B\}$ of the vertices in S_{REM} (s.t. A and B are independent sets in G).
- Then the graph $G \setminus S_{\text{KEEP}}$ has a small bipartization (that uses no vertices from S_{REM}) if and only if the graph $G \setminus S$ has a small bipartization where the neighbors the neighbors of A in G and the neighbors of B in G are in different parts of the bipartization.

Example: Graph Bipartisation

Hence, after guessing the at most 3^k partitions of S we are left with the following problem:

k-$\{A, B\}$-Graph Bipartisation **Parameter: k**

Input: A bipartite graph G, an integer k, and 2 independent vertex sets A and B.
Question: Is there a $S \subseteq V(G)$ with $|S'| \leq k$ such that $G \setminus S$ has a bipartization such that the vertices in $A \setminus S'$ and $B \setminus S'$ are in different parts?

Question

How to solve this problem?

Example: Graph Bipartisation

Answer

- Find an arbitrary bipartization $\{A_0, B_0\}$ of G.
- Then the vertices in $C := (A_0 \cap B) \cup (B_0 \cap A)$ have to change, while the vertices in $R := (A_0 \cap A) \cup (B_0 \cap B)$ should remain in the same part.
- Observation: There is a set $S \subseteq V(G)$ such that $G \setminus S$ has the required bipartization if and only if S separates C and R, i.e., no component of $G \setminus S$ contains vertices from both $C \setminus S$ and $R \setminus S$.

Example: Graph Bipartisation

Observation

There is a set $S \subseteq V(G)$ such that $G \setminus S$ has the required bipartization if and only if S separates C and R, i.e., no component of $G \setminus S$ contains vertices from both $C \setminus S$ and $R \setminus S$.

Proof (sketch):

$\rightarrow$ In a bipartitation of $G \setminus S$ every vertex either changed parts or stays in the same part. Adjacent vertices have to do the same. Hence, every component of $G \setminus S$ either changed or remained in the same part.
$\leftarrow$ Flip the parts for all vertices in components of $G \setminus S$ containing vertices from C. Hence, no vertex from R is flipped.

Example: Graph Bipartisation

Using max-flow min-cut techniques we can check whether there is such a set S that separates C and R in time $O(k|E(G)|)$.

Theorem

k-Graph Bipartization Compression can be solved in time $O(3^k n^{O(1)})$.

And using our iterative compression framework, we obtain:

Theorem

k-Graph Bipartization can be solved in time $O(3^k n^{O(1)})$.

Example: Feedback Vertex Set

k-FEEDBACK VERTEX SET **Parameter: k**

Input: A graph G and an integer k.
Question: Is there a set $S \subseteq V(G)$ with $|S| \leq k$ and $G \setminus S$ is a tree?

Example: Feedback Vertex Set

Using the sequence $G_i := G[v_1, \ldots, v_i]$ for an arbitrary ordering $v_1, \ldots, v_{|V(G)|}$ of the vertices of G we obtain:

(1) $S_1 := \emptyset$ is a k-FVS for G_1.

(2) If S_i is a k-FVS for G_i then $S_{i+1} := S_i \cup \{v_{i+1}\}$ is a $(k+1)$-FVS for G_{i+1}.

(3) If G_i has no k-FVS then G has no k-FVS.

(4) FPT-algorithm for compression version????

Hence, we only need to find an FPT-algorithm for the compression version of the problem!

Example: Feedback Vertex Set

Using the sequence $G_i := G[v_1, \ldots, v_i]$ for an arbitrary ordering $v_1, \ldots, v_{|V(G)|}$ of the vertices of G we obtain:

(1) $S_1 := \emptyset$ is a k-FVS for G_1.

(2) If S_i is a k-FVS for G_i then $S_{i+1} := S_i \cup \{v_{i+1}\}$ is a $(k+1)$-FVS for G_{i+1}.

(3) If G_i has no k-FVS then G has no k-FVS.

(4) FPT-algorithm for compression version????

Hence, we only need to find an FPT-algorithm for the compression version of the problem!

Example: Feedback Vertex Set

k-FEEDBACK VERTEX SET COMPRESSION **Parameter: k**

Input: A graph G , an integer k, and a $k+1$-FVS of G.
Question: Is there a k-FVS for G?

Question

How to solve this problem?

Example: Feedback Vertex Set

Answer

Again we guess the intersection of S with an optimal solution. There are 2^{k+1} such guesses and for each guess we have to solve an instance of k-FEEDBACK VERTEX SET DISJOINT defined below.

k-S-DISJOINT FEEDBACK VERTEX SET **Parameter:** $|S|$

Input: A graph G , and a FVS S of G.
Question: Is there a $|S| - 1$-FVS for G that is disjoint from S?

Example: Feedback Vertex Set

k-S-Disjoint Feedback Vertex Set **Parameter: $|S|$**

Input: A graph G , and a FVS S of G.
Question: Is there a $|S| - 1$-FVS for G that is disjoint from S?

Goal

We want to solve the above problem using kernelization.

Example: Feedback Vertex Set

Degree 1 rule

If $v \in V(G) \setminus S$ has degree 1 in G. Then G has a k-S-DFVS iff $G \setminus \{v\}$ has a k-S-DFVS.

Example: Feedback Vertex Set

Degree 2 rule

If $v \in V(G) \setminus S$ has 2 neighbors u, w in G and $w \notin S$. Then either

(1) $\{v, w\} \in E(G)$ and G has a k-S-DFVS iff $G \setminus \{w\}$ has a $k-1$-S-DFVS, or

(2) $\{v, w\} \notin E(G)$ and G has a k-S-DFVS iff G' has a k-S-DFVS, where G' is obtained by contracting $\{v, w\}$.

Hence, in a reduced instance (G, S, k) of k-DFVS, all vertices in $V(G) \setminus S$ have degree at least 3, or only neighbors in S.

Example: Feedback Vertex Set

Goal

Let (G, S, k) be a reduced k-S-DFVS instance and suppose a k-S-DFVS S' exists. We want to prove an upper bound on $|V(G)|$.

Let $T := G \setminus S$. Then T is a forest.

- Let H be the vertices in T with degree at least 3 in T.
- Let L be the vertices in T with degree 1 in T.
- Let R be the vertices in T with degree 2 in T.
- Let Z be the vertices in T with degree 0 in T.

Let $H' := H \cap S'$, $L' := L \cap S'$, $R' := R \cap S'$ and $Z' := Z \cap S'$.

Example: Feedback Vertex Set

Because $|E(T)| = \frac{1}{2}\sum_{v\in V(T)} d_T(v)$, we have:

$$|E(T)| = \frac{1}{2}|L| + |R| + \frac{1}{2}\sum_{v\in H} d_T(v)$$

Furthermore, the number of edges removed by deleting $S' = H' \cup L' \cup R'$ is at most:

$$|L'| + 2|R'| + \sum_{v\in H'} d_T(v)$$

Proposition

Let T be a forest with c components, where the set H (L) contains the vertices of degree at least 3 (exactly 1), respectively. Then $\sum_{v\in H}(d_T(v) - 2) = |L| - 2c$.

Example: Feedback Vertex Set

Combining the previous (in)equalities, we obtain:

$$
\begin{gathered}
|E(T \setminus S')| \geq \\
\tfrac{1}{2}|L| + |R| + \tfrac{1}{2}\sum_{v \in H} d_T(v) - |L'| - 2|R'| - \sum_{v \in H'} d_T(v) \geq \\
\sum_{v \in H}(d_T(v) - 1) + |R| - |L'| - 2|R'| - \sum_{v \in H'} d_T(v) = \\
\sum_{v \in H \setminus H'} d_T(v) - |H| + |R| - |L'| - 2|R'| \geq \\
3|H \setminus H'| - |H| + |R| - |L'| - 2|R'| = \\
2|H| - 3|H'| + |R| - 2|R'| - |L'|
\end{gathered}
$$

Example: Feedback Vertex Set

Because G is reduced the vertices in L, R, and Z have at least 2, 1, or 2 neighbors in S, respectively. Hence:

$$|E(G \setminus S')| \geq |E(T \setminus S')| + 2|L \setminus L'| + |R \setminus R'| + 2|Z \setminus Z'| \geq$$
$$2|H| - 3|H'| + |R| - 2|R'| - |L'| + 2|L| - 2|L'| + |R| - |R'| +$$
$$2|Z| - 2|Z'| =$$
$$2|H| - 3|H'| + 2|R| - 3|R'| + 2|L| - 3|L'| + 2|Z| - 2|Z'|$$

Because S' is FVS $G \setminus S'$ is a forest and hence:

$$|E(G \setminus S')| \leq |V(G \setminus S')| - 1 = |S| + |H| + |L| + |R| + |Z| - |S'| - 1$$

Example: Feedback Vertex Set

Combining these bounds gives:

$$2|H| - 3|H'| + 2|R| - 3|R'| + 2|L| - 3|L'| + 2|Z| - 2|Z'| \leq$$
$$|E(G \setminus S')| \leq$$
$$|S| + |H| + |L| + |R| + |Z| - |S'| - 1 \leftrightarrow$$
$$|H| + |L| + |R| + |Z| \leq 2|S'| + |S| - 1 \leftrightarrow$$
$$|V(G) \setminus S| \leq 3k$$

Hence, the reduced graph G has at most $4k + 1$ vertices!

Example: Feedback Vertex Set

Theorem

Let G be a graph that has a FVS S with $|S| = k + 1$. Then it can be decided whether G has a k-S-DFVS in time $n^{O(1)} + O^*(6.75^k)$.

Algorithm

(1) If $G[S]$ contains a cycle, return NO.
(2) Apply the degree 1 and 2 reduction rules until a reduced instance (G', S, k') is obtained (This needs time $n^{O(1)}$).
(3) If $|V(G') \setminus S| > 3k$, return NO.
(4) test all subsets $S' \subseteq V(G) \setminus S$ with $|S'| = k'$. If one of them is a FVS return YES, otherwise return NO.

Example: Feedback Vertex Set

Algorithm

(1) If $G[S]$ contains a cycle, return No.

(2) Apply the degree 1 and 2 reduction rules until a reduced instance (G', S, k') is obtained (This needs time $n^{O(1)}$).

(3) If $|V(G') \setminus S| > 3k$, return No.

(4) test all subsets $S' \subseteq V(G) \setminus S$ with $|S'| = k'$. If one of them is a FVS return Yes, otherwise return No.

The correctness of the algorithm follows from the previous slides. What about the complexity bound?

Example: Feedback Vertex Set

Theorem (Stirling's approximation)

$\lim_{n\to\infty} \frac{n!}{\sqrt{2\pi n}(\frac{n}{e})^n} = 1$

Corollary

$n! \in \Theta(\sqrt{n}(\frac{n}{e})^n)$

Recall

- $f(n) \in \Omega(g(n))$ means: there is c and N such that for every $n > N$ it holds that $f(n) \geq cg(n)$.
- $f(n) \in \Theta(g(n))$ means: $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$.

Clearly, if $f_1(n) \in \Theta(g_1(n))$ and $f_2(n) \in \Theta(g_2(n))$, then $f_1(n)f_2(n) \in \Theta(g_1(n)g_2(n))$ and $f_1(n)/f_2(n) \in \Theta(g_1(n)/g_2(n))$.

Example: Feedback Vertex Set

Applying the reduction rules takes time $n^{O(1)}$.
Let the reduced instance G' have n' vertices not in S. If $n' \leq 3k'$ (and $k \geq 2$), then the number of sets tested is:

$$\binom{n'}{k'} \leq \binom{3k'}{k'} \leq \binom{3k}{k} = \frac{(3k)!}{(2k)!k!} \in$$
$$\Theta\left(\frac{\sqrt{3k}(3k)^{3k}e^{-3k}}{\sqrt{2k}(2k)^{2k}e^{-2k}\sqrt{k}k^{k}e^{-k}}\right) \in$$
$$O\left(\frac{3^{3k}}{2^{2k}}\right) = O\left(\left(\frac{3^3}{2^2}\right)^k\right) = O(6.75^k)$$

Testing whether a set S' is a FVS can be done in polynomial time $k^{O(1)}$, hence the total time complexity is $n^{O(1)} + O(6.75^k)k^{O(1)}$.

Example: Feedback Vertex Set

On the last slide we had a function $f(k) \in \Theta(6.75^k k^c)$ for some constant c.

- Observe that: $f(k) \in O((6.75 + \epsilon)^k)$, but $f(k) \notin O(6.75^k)$.
- So the polynomial factor k^c seems irrelevant, but still we may not just omit it using the O-notation.
- To get around this annoying situation the O^* notation is defined less precise as the O-notation and one can state $6.75^k k^c \in O^*(6.75^k)$.

Example: Feedback Vertex Set

Hence, the overall complexity for the compression problem is $n^{O(1)} + O^*(6.75^k)$. Because we have to make $O(2^k)$ guesses to reduce to the compression problem, we obtain:

Theorem

k-FVS can be decided in time $n^{O(1)}O^*(13.5^k)$.