Integer Linear Programming

Input: An integer linear program with k variables.

Parameter: k

Question: Does this ILP have a solution?

This Problem is fixed parameter tractable.

The running time is $f(k)n^{O(1)}$, but the $f(k)$ are painfully large.

Proof: very involved...

Feedback Vertex Set

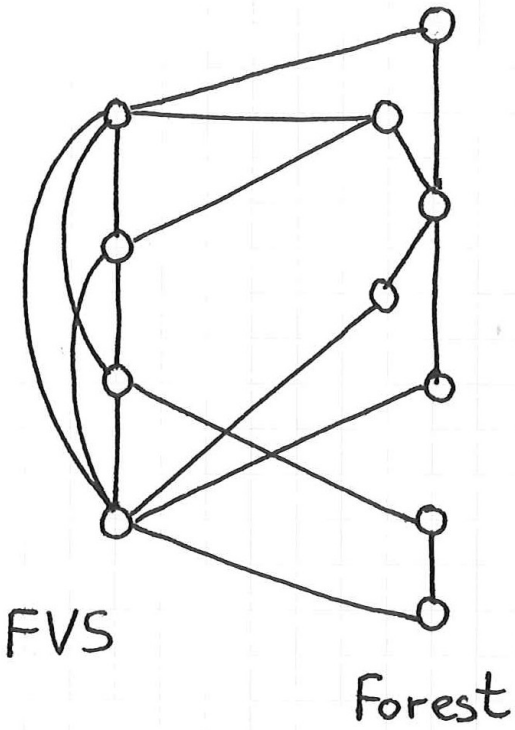
Input: A graph G and a number k

Parameter: k

Question: Are there $\leq k$ nodes whose removal makes G acyclic?

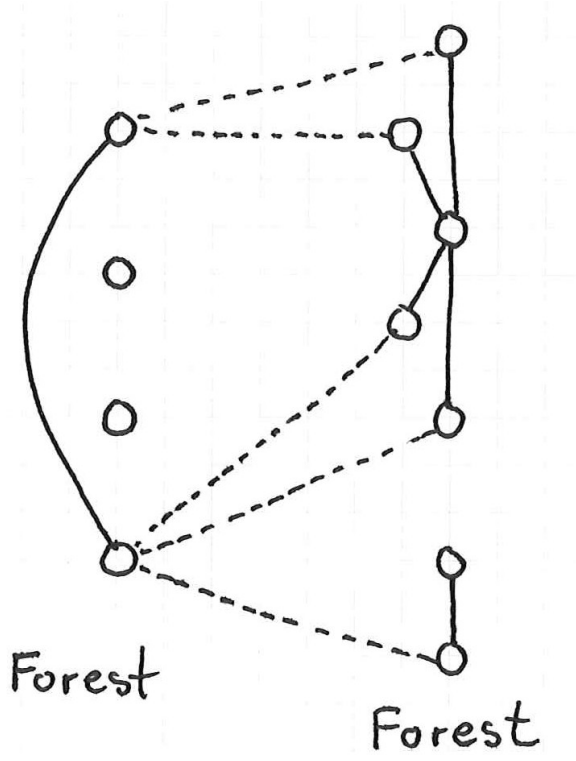
Is FVS fixed parameter tractable?

Iterative Compression



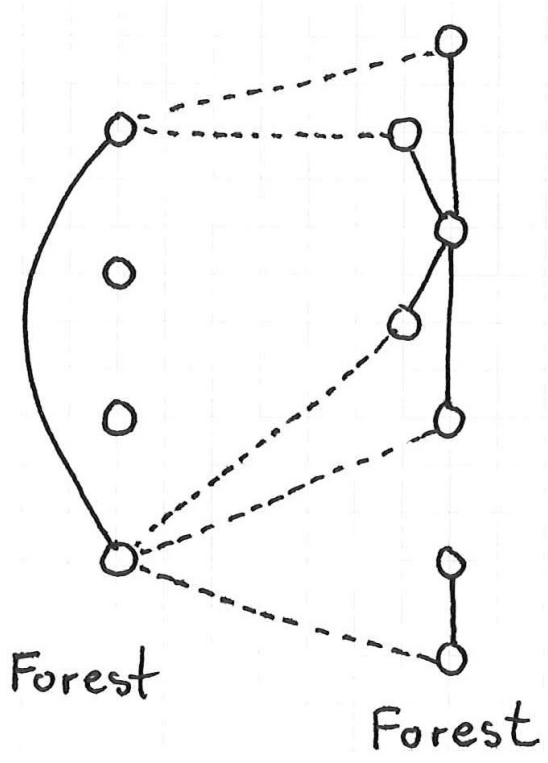
Assume we already know a FVS of size k.

Does this help to find a FVS of size $k-1$?



Step 1: Find a subset of the FVS to keep

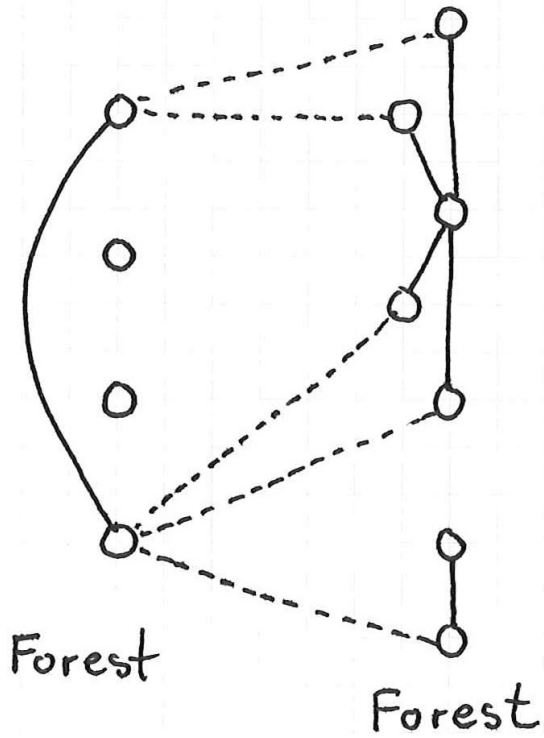
Plan: Add vertices from the forest to this FVS



Step 2: Apply reduction rules

Contract components of the FVS into one vertex

Contract degree-2 vertices in the forest



Step 3: Branching algorithm

If a node in the forest has two neighbors in the FVS:

a) put it into the FVS

b) contract its neighborhood

Running time

Size of the branching tree 2^k

Total size of all branching trees:

$$\sum_{j=0}^{k} \binom{k}{j} 2^k = 4^k$$

Total running time $4^k n^{O(1)}$

Overview

Depth-First Search Trees

Input: A graph G and a number k

Parameter: k

Question: Is there a path of length k in G?

Construct a depth-first search tree.

What is the helpful property of a DFS tree?

Depth-First Search Trees

Input: A graph G and a number k

Parameter: k

Question: Is there a path of length k in G?

Construct a depth-first search tree.

What is the helpful property of a DFS tree?

A Simple Theorem

Theorem

Let G be a graph and k a number.

Then it takes only polynomial time to find one of these:

1. *A cycle of length at least k*
2. *A tree decomposition of treewidth at most k*

Proof

$k+1$ cops slowly traverse the DFS tree

A Simple Theorem

Theorem

Let G be a graph and k a number.

Then it takes only polynomial time to find one of these:

1. *A cycle of length at least k*
2. *A tree decomposition of treewidth at most k*

Proof

$k + 1$ cops slowly traverse the DFS tree

Long Paths

The theorem allows us to find paths of length k easily:

1. If we find a cycle longer than k, there obviously is a path of length k as well
2. Otherwise we use the tree decomposition and Courcelle's theorem:
   $\exists x_1 \ldots \exists x_{k+1}(inc(x_1, x_2) \wedge \cdots \wedge inc(x_k, x_{k+1}) \wedge x_1 \neq x_2 \ldots)$

Long Paths

The theorem allows us to find paths of length k easily:

1. If we find a cycle longer than k, there obviously is a path of length k as well
2. Otherwise we use the tree decomposition and Courcelle's theorem:
   $\exists x_1 \ldots \exists x_{k+1}(inc(x_1, x_2) \wedge \cdots \wedge inc(x_k, x_{k+1}) \wedge x_1 \neq x_2 \ldots)$

Long Paths

The theorem allows us to find paths of length k easily:

1. If we find a cycle longer than k, there obviously is a path of length k as well
2. Otherwise we use the tree decomposition and Courcelle's theorem:
   $\exists x_1 \ldots \exists x_{k+1}(inc(x_1, x_2) \wedge \cdots \wedge inc(x_k, x_{k+1}) \wedge x_1 \neq x_2 \ldots)$

Question: Can we solve Vertex Cover this way?

A Complicated Theorem

Theorem (Bodlaender)

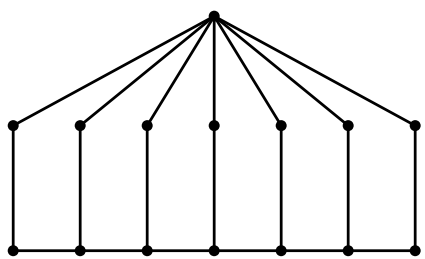
Let G be a graph and k, l some numbers.

It takes $f(k,l)|G|$ steps to find one of these:

1. *A subdivision of the $2 \times k$ grid*
2. *A subdivision of the l-circus graph*
3. *A tree decompositon of G of treewidth $2(k-1)^2(l-1)+1$.*

Proof

Again, using a DFS tree.



7-circus graph

A Complicated Theorem

Theorem (Bodlaender)

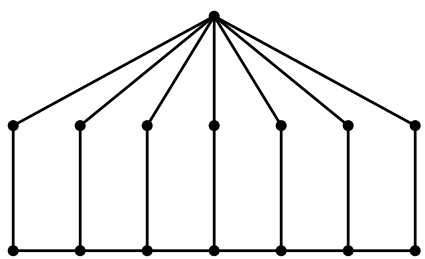
Let G be a graph and k, l some numbers.

It takes $f(k,l)|G|$ steps to find one of these:

1. *A subdivision of the $2 \times k$ grid*
2. *A subdivision of the l-circus graph*
3. *A tree decompositon of G of treewidth $2(k-1)^2(l-1)+1$.*

Proof

Again, using a DFS tree.



7-circus graph

A Complicated Theorem

Theorem (Bodlaender)

Let G be a graph and k, l some numbers.

It takes $f(k,l)|G|$ steps to find one of these:

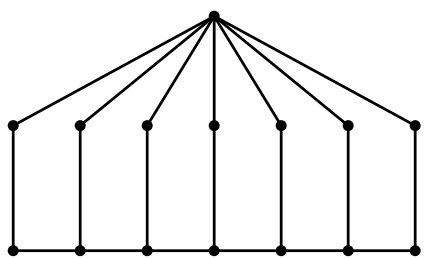
1. *A subdivision of the $2 \times k$ grid*
2. *A subdivision*
3. *A tree decom*

Question: Can we solve Dominating Set this way?

$-1)+1$.

Proof

Again, using a DFS tree.



7-circus graph

A Complicated Theorem

Theorem (Bodlaender)

Let G be a graph and k, l some numbers.

It takes $f(k,l)|G|$ steps to find one of these:

1. *A subdivision of the $2 \times k$ grid*
2. *A subdivision*
3. *A tree decom*

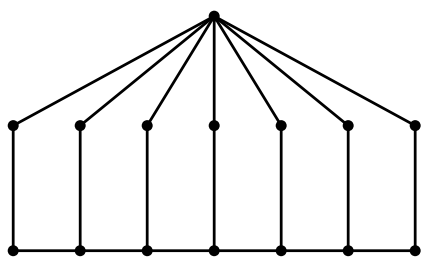
$-1)+1$.

Question: Can we solve Dominating Set this way?

Proof

Again, using

Why did it work for planar graphs?



7-circus graph

Application 1

Max-Leaf-Spanning-Tree:

Input: A graph G and a number k

Parameter: k

Question: Does G have a spanning tree with at least k leaves?

Both the $2 \times k$ grid and the k-circus graph contain a tree with k leaves.

That is, Max-Leaf-Spanning-Tree is fixed parameter tractable.

Application 1

Max-Leaf-Spanning-Tree:

Input: A graph G and a number k

Param

Does the following statement hold?

Questi

If a graph contains a tree with k leaves, then it also contains a spanning tree with at least k leaves.

ast k leaves?

Both the $2 \times k$ grid and the k-circus graph contain a tree with k leaves.

That is, Max-Leaf-Spanning-Tree is fixed parameter tractable.

Application 2

Feedback Vertex Set:

Input: A graph G and a number k

Parameter: k

Question: Are there $\leq k$ nodes whose removal makes G acyclic?

Theorem

Feedback Vertex Set is fixed parameter tractable.

Application 2

Feedback Vertex Set:

Input: A graph G and a number k

Parameter: k

Question: Are there $\leq k$ nodes whose removal makes G acyclic?

Theorem

Feedback Vertex Set is fixed parameter tractable.

Feedback Vertex Set

Theorem

Feedback Vertex Set is fixed parameter tractable.

Proof

Apply Bodlaender's theorem.

1. Small tree decomposition: Courcelle
2. $2 \times 3k$ grid: No
3. $4k$-circus graph: Remove the tip and check for a FVS of size $k-1$.

Feedback Vertex Set

Theorem

Feedback Vertex Set is fixed parameter tractable.

Proof

Apply Bodlaender's theorem.

1. Small tree decomposition: Courcelle
2. $2 \times 3k$ grid: No
3. $4k$-circus graph: Remove the tip and check for a FVS of size $k-1$.

Feedback Vertex Set

Theorem

Feedback Vertex Set is fixed parameter tractable.

Proof

Apply Bodlaender's theorem.

1. Small tree decomposition: Courcelle
2. $2 \times 3k$ grid: No
3. $4k$-circus graph: Remove the tip and check for a FVS of size $k-1$.

Feedback Vertex Set

Theorem

Feedback Vertex Set is fixed parameter tractable.

Proof

Apply Bodlaender's theorem.

1. Small tree decomposition: Courcelle
2. $2 \times 3k$ grid: No
3. $4k$-circus graph: Remove the tip and check for a FVS of size $k-1$.

Feedback Vertex Set

Theorem

Feedback Vertex Set is fixed parameter tractable.

Proof

Apply Bodlaender's theorem.

1. Small tree decomposition: Courcelle
2. $2 \times 3k$ grid: No
3. $4k$-circus graph: Remove the tip and check for a FVS of size $k-1$.

Feedback Vertex Set

Theorem

Feedback Vertex Set is fixed parameter tractable.

Proof

Apply Bodlaender's theorem.

1. Small tree decomposition: Courcelle
2. $2 \times 3k$ grid: No
3. $4k$-circus graph: Remove the tip and check for a FVS of size $k - 1$.

Overview

Parameterized Complexity Theory

Classical complexity theory:

- Complexity classes P, NP, etc.
- Languages $L \in P$, $L \subseteq \Sigma^*$
- Framework insufficient for parameterized problems

Parameterized Complexity Theory

Definition

A parameterized problem over the alphabet Σ is a set of pairs (w, k), where $w \in \Sigma^*$ and $k \in \mathbf{N}$.

It is not allowed that there exists w and $k \neq k'$ with $(w, k) \in L$ and $(w, k') \in L$, if L is a parameterized problem.

The second condition states that k is a function of w.

Parameterized Complexity Theoery

We like to state parameterized problems as follows:

Input: A graph G and a number k

Parameter: k

Question: Does G contain a clique of size k as a subgraph?

Parameterized Complexity Theory

The parameter can be some arbitraty number, if it can be easily computed from the input.

Input: A graph G and a number k

Parameter: The diameter of G

Question: Does G contain a clique of size k as a subgraph?

Here it is easy to compute $(G, \Delta(G))$ from G in order to get formally a parameterized problem.

Parameterized Complexity Theory

One goal of complexity theory is to categorize problems into easy and hard ones.

For this purpose P and NP are best known.

Others are:

- NC and L
- AC^0 and NC^1
- $EXPTIME$ and $EXPSPACE$
- etc. etc.

Parameterized Complexity Theory

In parameterized complexity theory the easy problems can be found in the class FPT.

Definition

The class FPT contains all parameterized problems that are fixed parameter tractable.

Formally: $L \in FPT$, if there is an algorithm solving $(w, k) \in L$ in at most $f(k)|w|^c$ steps, where c is a constant and $f : \mathbf{N} \to \mathbf{N}$ an arbitrary function.

Parameterized Complexity Theory

A fundamental concept in complexity theory are reductions.

Important example: polynomial time many-one reductions:

$g : \Sigma^* \to \Sigma^*$ reduces the problem L_1 to L_2, if

1. $w \in L_1 \iff g(w) \in L_2$.
2. $g(w)$ can be computed in $|w|^{O(1)}$ steps.

Important property: If L_1 can be reduced to L_2 and $L_1 \notin P$, then $L_2 \notin P$.

"I can't find an efficient algorithm, but neither can all these famous people."

Important property: If L_1 can be reduced to L_2 and $L_1 \notin P$, then $L_2 \notin P$.

Parameterized Complexity Theory

Question: Is this reduction useful for parameterized problems?

1. $w \in L_1 \iff g(w) \in L_2$.
2. $g(w)$ can be computed in $|w|^{O(1)}$ steps.

Does the corresping property hold: It L_1 can be reduced to L_2 and $L_1 \notin FPT$, then $L_2 \notin FPT$.

Parameterized Complexity Theory

That corresponding property does not hold:

We can map (w, k) to $(w, |w|)$!

If we reduce a problem to itself like this, we have $f(|w|)|w|^c$ steps instead of $f(|w|)|w|^c$ steps to compute a solution.

It that way we can solve every computable problem.

A polynomial time reduction is not fine grained enough.

Parameterized Reductions

Definition

A parameterized problem $L_1 \subseteq \Sigma^*$ can be reduced to $L_2 \subseteq \Gamma^*$ by a parameterized reduction if

- $r, s \colon \mathbf{N} \to \mathbf{N}$ are computable functions,
- there is a function $g \colon \Sigma^* \times \mathbf{N} \to \Gamma^*$, $(w, k) \mapsto (w', k')$, that can be computed in $r(k)|w|^{O(1)}$ steps and $k' = s(k)$,
- $(w, k) \in L_1$ if and only if $g(w, k) \in L_2$.

Parameterized Reductions

Theorem

If $L_1 \notin FPT$ and there is a parameterized reduction from L_1 to L_2, then $L_2 \notin FPT$.

Proof

Assume $L_2 \in FPT$. We can computed $(w', k') = g(w, k)$ in $r(k)|w|^c$ steps such that $k' = s(k)$ and $|w'| \leq r(k)|w|^c$.

Then test whether $(w', k') \in L_2$ taking $f'(k')|w'|^d \leq f'(s(k))r(k)^d|w|^{cd}$ steps.

Because $(w, k) \in L_1 \iff (w', k') \in L_2$, we answered whether $(w, k) \in L_1$ holds and therefore $L_1 \in FPT$.