6. Parameterized branching algorithms
COMP6741: Parameterized and Exact Computation

Serge Gaspers
Semester 2, 2017

Contents
1 Running time analysis 1
2 Feedback Vertex Set 2
3 Maximum Leaf Spanning Tree 3
4 Further Reading 5

1 Running time analysis

Search trees

Recall: A search tree models the recursive calls of an algorithm. For a \( b \)-way branching where the parameter \( k \) decreases by \( a \) at each recursive call, the number of nodes is at most \( b^{k/a} \cdot (k/a + 1) \).

\[
\begin{array}{c}
  k \\
  \downarrow \\
  k - a \\
  \downarrow \\
  k - 2a \\
  \downarrow \\
  \vdots \\
  \downarrow \\
  \leq b^{k/a}
\end{array}
\]

\[ k - a \\
\downarrow \\
k - 2a \\
\downarrow \\
k - 2a \\
\downarrow \\
k - 2a \\
\downarrow \\
\leq k/a + 1
\]

If \( k/a \) and \( b \) are upper bounded by a function of \( k \), and the time spent at each node is FPT (typically, polynomial), then we get an FPT running time.

Recall: Measure Based Analysis

For more precise running time upper bounds:

Lemma 1 (Measure Analysis Lemma). Let

- \( A \) be a branching algorithm
- \( c \geq 0 \) be a constant, and
- \( \mu(\cdot), \eta(\cdot) \) be two measures for the instances of \( A \),

such that on input \( I \), \( A \) calls itself recursively on instances \( I_1, \ldots, I_k \), but, besides the recursive calls, uses time \( O((\eta(I))^c) \), such that

\[
(\forall i) \quad \eta(I_i) \leq \eta(I) - 1, \quad \text{and} \quad 2^{\mu(I_1)} + \ldots + 2^{\mu(I_k)} \leq 2^{\mu(I)}.
\]

Then \( A \) solves any instance \( I \) in time \( O(\eta(I)^{c+1}) \cdot 2^{\mu(I)} \).
2 Feedback Vertex Set

A *feedback vertex set* of a multigraph $G = (V, E)$ is a set of vertices $S \subseteq V$ such that $G - S$ is acyclic.

<table>
<thead>
<tr>
<th>FEEDBACK VERTEX SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input: Multigraph $G = (V, E)$, integer $k$</td>
</tr>
<tr>
<td>Parameter: $k$</td>
</tr>
<tr>
<td>Question: Does $G$ have a feedback vertex set of size at most $k$?</td>
</tr>
</tbody>
</table>

**Simplification Rules**

We apply the first *applicable* simplification rule.

(Loop)
If $G$ has a loop $vv \in E$, then set $G \leftarrow G - v$ and $k \leftarrow k - 1$.

(Multiedge)
If $E$ contains an edge $uv$ more than twice, remove all but two copies of $uv$.

(Degree-1)
If $\exists v \in V$ with $d_G(v) \leq 1$, then set $G \leftarrow G - v$.

(Budget-exceeded)
If $k < 0$, then return No.

(Degree-2)
If $\exists v \in V$ with $d_G(v) = 2$, then denote $N_G(v) = \{u, w\}$ and set $G \leftarrow G' = (V \setminus \{v\}, (E \setminus \{vu, vw\}) \cup \{uw\})$.

Lemma 2. (Degree-2) is sound.

*Proof.* Suppose $S$ is a feedback vertex set of $G$ of size at most $k$. Let

$$S' = \begin{cases}  
S & \text{if } v \notin S \\
(S \setminus \{v\}) \cup \{u\} & \text{if } v \in S. 
\end{cases}$$

Now, $|S'| \leq k$ and $S'$ is a feedback vertex set of $G'$ since every cycle in $G'$ corresponds to a cycle in $G$, with, possibly, the edge $uw$ replaced by the path $(u, v, w)$.

Suppose $S'$ is a feedback vertex set of $G'$ of size at most $k$. Then, $S'$ is also a feedback vertex set of $G$. \(\square\)

**Remaining issues**

- A select–discard branching decreases $k$ in only one branch
- One could branch on all the vertices of a cycle, but the length of a shortest cycle might not be bounded by any function of $k$

Idea:

- An acyclic graph has average degree $< 2$
- After applying simplification rules, $G$ has average degree $\geq 3$
- The selected feedback vertex set needs to be incident to many edges
- Does a feedback vertex set of size at most $k$ contain at least one vertex among the $f(k)$ vertices of highest degree?

---

1 A simplification rule is *applicable* if it modifies the instance.
The fvs needs to be incident to many edges

**Lemma 3.** If \( S \) is a feedback vertex set of \( G = (V, E) \), then
\[
\sum_{v \in S} (d_G(v) - 1) \geq |E| - |V| + 1
\]

*Proof.* Since \( F = G - S \) is acyclic, \( |E(F)| \leq |V| - |S| - 1 \). Since every edge in \( E \setminus E(F) \) is incident with a vertex of \( S \), we have
\[
|E| = |E| - |E(F)| + |E(F)| \\
\leq \left( \sum_{v \in S} d_G(v) \right) + (|V| - |S| - 1) \\
= \left( \sum_{v \in S} (d_G(v) - 1) \right) + |V| - 1.
\]

□

The fvs needs to contain a high-degree vertex

**Lemma 4.** Let \( G \) be a graph with minimum degree at least 3 and let \( H \) denote a set of \( 3k \) vertices of highest degree in \( G \). Every feedback vertex set of \( G \) of size at most \( k \) contains at least one vertex of \( H \).

*Proof.* Suppose not. Let \( S \) be a feedback vertex set with \( |S| \leq k \) and \( S \cap H = \emptyset \). Then,
\[
2|E| - |V| = \sum_{v \in V} (d_G(v) - 1) \\
= \sum_{v \in H} (d_G(v) - 1) + \sum_{v \in V \setminus H} (d_G(v) - 1) \\
\geq 3 \cdot (\sum_{v \in S} (d_G(v) - 1)) + \sum_{v \in S} (d_G(v) - 1) \\
\geq 4 \cdot (|E| - |V| + 1) \\
\Leftrightarrow 3|V| \geq 2|E| + 4.
\]

But this contradicts the fact that every vertex of \( G \) has degree at least 3. □

**Algorithm for Feedback Vertex Set**

**Theorem 5.** Feedback Vertex Set can be solved in \( O^*((3k)^k) \) time.

*Proof (sketch).*
- Exhaustively apply the simplification rules.
- The branching rule computes \( H \) of size \( 3k \), and branches into subproblems \( (G - v, k - 1) \) for each \( v \in H \).

Current best: \( O^*(3.619^k) \) [Kociumaka, Pilipczuk, 2014]

### 3 Maximum Leaf Spanning Tree

A *leaf* of a tree is a vertex with degree 1. A *spanning tree* in a graph \( G = (V, E) \) is a subgraph of \( G \) that is a tree and has \( |V| \) vertices.

<table>
<thead>
<tr>
<th>Maximum Leaf Spanning Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> connected graph ( G ), integer ( k )</td>
</tr>
<tr>
<td><strong>Parameter:</strong> ( k )</td>
</tr>
<tr>
<td><strong>Question:</strong> Does ( G ) have a spanning tree with at least ( k ) leaves?</td>
</tr>
</tbody>
</table>
Property

A \textit{k-leaf tree} in \( G \) is a subgraph of \( G \) that is a tree with at least \( k \) leaves. A \textit{k-leaf spanning tree} in \( G \) is a spanning tree in \( G \) with at least \( k \) leaves.

Lemma 6. \textit{Let} \( G = (V, E) \) \textit{be a connected graph.} \( G \) has a \textit{k-leaf tree} \( \iff \) \( G \) has a \textit{k-leaf spanning tree.} \]

Proof. \((\Rightarrow)\): \textit{trivial} 
\((\Leftarrow)\): Let \( T \) be a \( k \)-leaf tree in \( G \). By induction on \( x := |V| - |V(T)| \), we will show that \( T \) can be extended to a \( k \)-leaf spanning tree in \( G \).
Base case: \( x = 0 \). 
Induction: \( x > 0 \), and assume the claim is true for all \( x' < x \). Choose \( uv \in E \) such that \( u \in V(T) \) and \( v \notin V(T) \). 
Since \( T' := (V(T) \cup \{v\}, E(T) \cup \{uv\}) \) has \( \geq k \) leaves and \( < x \) external vertices, it can be extended to a \( k \)-leaf spanning tree in \( G \) by the induction hypothesis. \( \square \)

Strategy

- The branching algorithm will check whether \( G \) has a \( k \)-leaf tree.
- A tree with \( \geq 3 \) vertices has at least one \textit{internal} (= non-leaf) vertex.
- "Guess" an internal vertex \( r \), i.e., do a \(|V|\)-way branching fixing an initial internal vertex \( r \).
- In any branch, the algorithm has computed
  - \( T \) – a tree in \( G \)
  - \( I \) – the internal vertices of \( T \), with \( r \in I \)
  - \( B \) – a subset of the leaves of \( T \) where \( T \) may be extended: the boundary set
  - \( L \) – the remaining leaves of \( T \)
  - \( X \) – the external vertices \( V \setminus V(T) \)
- The question is whether \( T \) can be extended to a \( k \)-leaf tree where all the vertices in \( L \) are leaves.

Simplification Rules

Apply the first applicable simplification rule:

(Halt-Yes)
If \( |L| + |B| \geq k \), then return Yes.

(Halt-No)
If \( |L| = 0 \), then return No.

(Non-extendable)
If \( \exists v \in B \) with \( N_G(v) \cap X = \emptyset \), then move \( v \) to \( L \).

Branching Lemma

Lemma 7 (Branching Lemma). \textit{Suppose} \( u \in B \) \textit{and there exists a} \( k \)-leaf tree \( T' \) \textit{extending} \( T \) \textit{where} \( u \) \textit{is an internal vertex.} \textit{Then, there exists a} \( k \)-leaf tree \( T'' \) \textit{extending} \( (V(T) \cup N_G(u), E(T) \cup \{uv : v \in N_G(u) \cap X\}) \).

Proof. Start from \( T'' \leftarrow T' \) \textit{and perform the following operation for each} \( v \in N_G(u) \cap X \).

If \( v \notin V(T') \), then add he vertex \( v \) and the edge \( uv \). Otherwise, add the edge \( uv \), creating a cycle \( C \) in \( T \) and remove the other edge of \( C \) incident to \( v \). This does not decrease the number of leaves, since it only increases the number of edges incident to \( u \), and \( u \) was already internal. \( \square \)

Follow Path Lemma

Lemma 8 (Follow Path Lemma). \textit{Suppose} \( u \in B \) \textit{and} \( |N_G(u) \cap X| = 1 \). \textit{Let} \( N_G(u) \cap X = \{v\} \). \textit{If there exists a} \( k \)-leaf tree \textit{extending} \( T \) \textit{where} \( u \) \textit{is internal, but no} \( k \)-leaf tree \textit{extending} \( T \) \textit{where} \( u \) \textit{is a leaf, then there exists a} \( k \)-leaf tree \textit{extending} \( T \) \textit{where both} \( u \) \textit{and} \( v \) \textit{are internal.}

Proof. Suppose not, and let \( T' \) be a \( k \)-leaf tree extending \( T \) where \( u \) is internal and \( v \) is a leaf. But then, \( T - v \) is a \( k \)-leaf tree as well. \( \square \)
Algorithm

- Apply simplification rules
- Select \( u \in B \). Branch into
  - \( u \in L \)
  - \( u \in I \). In this case, add \( X \cap N_G(u) \) to \( B \) (Branching Lemma). In the special case where \( |X \cap N_G(u)| = 1 \), denote \( \{v\} = X \cap N_G(u) \), make \( v \) internal, and add \( N_G(v) \cap X \) to \( B \), continuing the same way until reaching a vertex with at least 2 neighbors in \( X \) (Follow Path Lemma).

- In one branch, a vertex moves from \( B \) to \( L \); in the other branch, \(|B|\) increases by at least 1.

Running time analysis

- Measure \( \mu := 2k - 2|L| - |B| \geq 0 \).
- Branch where \( u \in L \):
  - \(|B|\) decreases by 1, \(|L|\) increases by 1
  - \( \mu \) decreases by 1
- Branch where \( u \in I \).
  - \( u \) moves from \( B \) to \( I \)
  - \( \geq 2 \) vertices move from \( X \) to \( B \)
  - \( \mu \) decreases by at least 1

- Binary search tree
- Height \( \leq \mu \leq 2k \)

Result for Maximum Leaf Spanning Tree

**Theorem 9** ([Kneis, Langer, Rossmanith, 2011]). **Maximum Leaf Spanning Tree can be solved in** \( O^*(4^k) \) **time.**

Current best: \( O^*(3.72^k) \) [Daligault, Gutin, Kim, Yeo, 2010]

4 Further Reading

