Introduction
Maximum Leaf Out-branchings
Minimum Leaf Out-branchings
Fast P-Space Algorithm for Out-Branchings with at Least $k$ Inter

Algorithms for Out-Branchings in Digraphs

Gregory Gutin

School of Computer Science and Mathematics
Royal Holloway, University of London

Liverpool, 20th Sep 2018
Outline

1. Introduction
2. Maximum Leaf Out-branchings
3. Minimum Leaf Out-branchings
4. Fast P-Space Algorithm for Out-Branchings with at Least $k$ Internal Vertices
Outline

1 Introduction

2 Maximum Leaf Out-branchings

3 Minimum Leaf Out-branchings

4 Fast P-Space Algorithm for Out-Branchings with at Least $k$ Internal Vertices
Very Recent Book on Digraphs

Introduction
Maximum Leaf Out-branchings
Minimum Leaf Out-branchings
Fast P-Space Algorithm for Out-Branchings with at Least $k$ Internal Vertices
Out/In-Trees and Out/In-Branchings

A subgraph $T^+ (T^-)$ of a digraph $D$ is an out-tree (in-tree) if $T$ is an oriented tree with only one vertex $s$ of in-degree (out-degree) 0 (root).
Out/In-Trees and Out/In-Branchings

- A subgraph $T^+$ ($T^-$) of a digraph $D$ is an out-tree (in-tree) if $T$ is an oriented tree with only one vertex $s$ of in-degree (out-degree) $0$ (root).

- Vertices of $T^+$ ($T^-$) of out-degree (in-degree) $0$ are leaves; non-leaves = internal vertices.
Out/In-Trees and Out/In-Branchings

- A subgraph \( T^+ (T^-) \) of a digraph \( D \) is an out-tree (in-tree) if \( T \) is an oriented tree with only one vertex \( s \) of in-degree (out-degree) 0 (root).

- Vertices of \( T^+ (T^-) \) of out-degree (in-degree) 0 are leaves; non-leaves = internal vertices.

- out-branching = spanning out-tree; in-branching = spanning in-tree
Out/In-Trees and Out/In-Branchings

- A subgraph $T^+ (T^-)$ of a digraph $D$ is an out-tree (in-tree) if $T$ is an oriented tree with only one vertex $s$ of in-degree (out-degree) 0 (root).

- Vertices of $T^+ (T^-)$ of out-degree (in-degree) 0 are leaves; non-leaves = internal vertices.

- out-branching = spanning out-tree; in-branching = spanning in-tree

- A digraph $D$ has an out-branching (in-branching) iff $D$ has only one initial (terminal) strongly connected component.
Example

Figure 1: A digraph $D$ and its out-branchings with minimum and maximum number of leaves ($Q$ and $R$, respectively).
Some Well-Known Results

- A digraph $D$ has an out-branching iff $D$ has only one initial strongly-connected component. (folklore)
Some Well-Known Results

- A digraph $D$ has an out-branching iff $D$ has only one initial strongly-connected component. (folklore)

- **Matrix Tree Theorem.** For a digraph $D = ([n], A)$, Kirchoff matrix $K = [K_{ij}]$: $K_{ij} := -x_{ij}$ if $i \neq j$ and $ij \in A$, and $\sum_{\ell \in A} x_{\ell}$ if $i = j$. $K_r$ is Kirchoff matrix minus $r$'th row and column. $B_r$ is the set of out-branchings rooted at $r$. Then $\det(K_r) = \sum_{B \in B_r} \prod_{ij \in A(B)} x_{ij}$. 
Some Well-Known Results

- A digraph $D$ has an out-branching iff $D$ has only one initial strongly-connected component. (folklore)

- **Matrix Tree Theorem.** For a digraph $D = ([n], A)$, Kirchhoff matrix $K = [K_{ij}]$: $K_{ij} := -x_{ij}$ if $i \neq j$ and $ij \in A$, and $\sum_{\ell i \in A} x_{\ell i}$ if $i = j$. $K_r$ is Kirchhoff matrix minus $r$’th row and column. $B_r$ is the set of out-branchings rooted at $r$. Then $\det(K_r) = \sum_{B \in B_r} \prod_{ij \in A(B)} x_{ij}$.

- A min weight out-branching in polynomial time: intersection of two matroids, an $O(n(n + m))$-time algorithm (Edmonds, 1967).
Problems with Extremal Number of Leaves

- Find an out-branching with \( \min \) number of leaves, \( \ell_{\min}(D) \), or find an out-branching with \( \max \) number of internal vertices, \( iv_{\max}(D) \).
Problems with Extremal Number of Leaves

- Find an out-branching with min number of leaves, \( \ell_{\text{min}}(D) \), or find an out-branching with max number of internal vertices, \( iv_{\text{max}}(D) \).

- If \( \ell_{\text{min}}(D) = 1 \), \( D \) has a Hamilton dipath.
Problems with Extremal Number of Leaves

- Find an out-branching with min number of leaves, \( \ell_{\text{min}}(D) \), or find an out-branching with max number of internal vertices, \( iv_{\text{max}}(D) \).

- If \( \ell_{\text{min}}(D) = 1 \), \( D \) has a Hamilton dipath.

- Find an out-branching with max number of leaves, \( \ell_{\text{max}}(D) \).
Outline

1 Introduction

2 Maximum Leaf Out-branchings

3 Minimum Leaf Out-branchings

4 Fast P-Space Algorithm for Out-Branchings with at Least $k$ Internal Vertices
Finding a max leaf out-branching is NP-hard even for acyclic digraphs. [Alon, Fomin, Gutin, Krivelevich, Saurabh, 2007]
\textbf{\textit{k-Leaf-Out-Branching problem}}

- Finding a max leaf out-branching is NP-hard even for acyclic digraphs. [Alon, Fomin, Gutin, Krivelevich, Saurabh, 2007]

- \textit{k-Leaf-Out-Branching}: given $D$ check whether $\ell_{\max}(D) \geq k$. Is \textit{k-Leaf-Out-Branching} FPT? [M. Fellows, 2005]
**k-Leaf-Out-Branching problem**

- Finding a max leaf out-branching is NP-hard even for acyclic digraphs. [Alon, Fomin, Gutin, Krivelevich, Saurabh, 2007]

- **k-Leaf-Out-Branching**: given $D$ check whether $\ell_{\text{max}}(D) \geq k$. Is k-Leaf-Out-Branching FPT? [M. Fellows, 2005]

- Alon et al. (2007): an $O^*(2^{O(k \log^2 k)})$-time algorithm for strong digraphs and a $O^*(2^{O(k \log k)})$-time algorithm for acyclic digraphs.
**Introduction**

**Maximum Leaf Out-branchings**

**Minimum Leaf Out-branchings**

Fast P-Space Algorithm for Out-Branchings with at Least $k$ Internal Vertices

---

**$k$-Leaf-Out-Branching problem**

- Finding a max leaf out-branching is NP-hard even for acyclic digraphs. [Alon, Fomin, Gutin, Krivelevich, Saurabh, 2007]

- **$k$-Leaf-Out-Branching**: given $D$ check whether $\ell_{\text{max}}(D) \geq k$. Is $k$-Leaf-Out-Branching FPT? [M. Fellows, 2005]

- Alon et al. (2007): an $O^*(2^{O(k \log^2 k)})$-time algorithm for strong digraphs and a $O^*(2^{O(k \log k)})$-time algorithm for acyclic digraphs.

- Bonsma and Dorn (2008): an $O^*(2^{O(k \log k)})$-time algorithm.
Faster Algorithms

Faster Algorithms


Simple: at each iteration the algorithm either declares a leaf $v$ of the current out-tree $T$ leaf of the out-branching or adds all children of $v$ to $T$.
Faster Algorithms


  Simple: at each iteration the algorithm either declares a leaf $v$ of the current out-tree $T$ leaf of the out-branching or adds all children of $v$ to $T$.

- Daligault, Gutin, Kim and Yeo (2010): an $O^*(3.72^k)$-time algorithm (currently fastest).
Kernels

- Binkele-Raible, Fernau, Fomin, Lokshtanov, Saurabh and Villanger (2012): no polynomial kernel for k-Leaf-Out-Branching (for arbitrary digraphs) unless $\text{coNP} \subseteq \text{NP}/\text{poly}$, which is highly unlikely.
Kernels

- Binkele-Raible, Fernau, Fomin, Lokshtanov, Saurabh and Villanger (2012): no polynomial kernel for $k$-Leaf-Out-Branching (for arbitrary digraphs) unless $\text{coNP} \subseteq \text{NP/poly}$, which is highly unlikely.

- Daligault, Gutin, Kim and Yeo (2010): an $O(k)$-vertex kernel for acyclic digraphs.
Kernels

- Binkele-Raible, Fernau, Fomin, Lokshtanov, Saurabh and Villanger (2012): no polynomial kernel for $k$-Leaf-Out-Branching (for arbitrary digraphs) unless $\text{coNP} \subseteq \text{NP}/\text{poly}$, which is highly unlikely.

- Daligault, Gutin, Kim and Yeo (2010): an $O(k)$-vertex kernel for acyclic digraphs.

- Binkele-Raible et al. (2012): an $O(k^3)$-vertex kernel for Rooted $k$-Leaf-Out-Branching. Thus, a Turing polynomial kernel exists.
Kernels

- Binkele-Raible, Fernau, Fomin, Lokshtanov, Saurabh and Villanger (2012): no polynomial kernel for $k$-Leaf-Out-Branching (for arbitrary digraphs) unless $\text{coNP} \subseteq \text{NP/poly}$, which is highly unlikely.

- Daligault, Gutin, Kim and Yeo (2010): an $O(k)$-vertex kernel for acyclic digraphs.

- Binkele-Raible et al. (2012): an $O(k^3)$-vertex kernel for Rooted $k$-Leaf-Out-Branching. Thus, a Turing polynomial kernel exists.

- Still open: Is there an $O(k)$-vertex kernel for Rooted $k$-Leaf-Out-Branching?
Outline

1. Introduction

2. Maximum Leaf Out-branchings

3. Minimum Leaf Out-branchings

4. Fast P-Space Algorithm for Out-Branchings with at Least $k$ Internal Vertices
**MinLeaf for DAGs**

**Definition**

MinLeaf: For a digraph $D$ find an out-branching with $\ell_{\min}(D)$ leaves.
MinLeaf for DAGs

Definition

MinLeaf: For a digraph $D$ find an out-branching with $\ell_{\text{min}}(D)$ leaves.

- US patent of Demers and Downing, 2000, for database search. Reduced to MinLeaf in directed acyclic graphs (DAGs). A heuristic suggested.
MinLeaf for DAGs

Definition

MinLeaf: For a digraph $D$ find an out-branching with $\ell_{\min}(D)$ leaves.

- US patent of Demers and Downing, 2000, for database search. Reduced to MinLeaf in directed acyclic graphs (DAGs). A heuristic suggested.

- Gutin, Razgon and Kim, 2009: a polytime algorithm for MinLeaf on DAGs.
Fixed-Parameter Tractability: a Generalization of P

Definition

A parameterized problem $\Pi$ can be considered as a set of pairs $(I, k)$ where $I$ is the problem instance and $k$ is the parameter.
Fixed-Parameter Tractability: a Generalization of P

**Definition**

A parameterized problem $\Pi$ can be considered as a set of pairs $(l, k)$ where $l$ is the problem instance and $k$ is the parameter.

**Definition**

$\Pi$ is fixed-parameter tractable (FPT) if membership of $(l, k)$ in $\Pi$ can be decided in time $O(f(k)|l|^{O(1)}) = O^*(f(k))$, where $f(k)$ is a computable function.
**Fixed-Parameter Tractability: a Generalization of P**

**Definition**

A parameterized problem $\Pi$ can be considered as a set of pairs $(I, k)$ where $I$ is the problem instance and $k$ is the parameter.

**Definition**

$\Pi$ is fixed-parameter tractable (FPT) if membership of $(I, k)$ in $\Pi$ can be decided in time $O(f(k)|I|^{O(1)}) = O^*(f(k))$, where $f(k)$ is a computable function.

**Definition**

A kernelization is a polytime reduction $(I, k) \mapsto (I', k')$ from a parameterized problem $\Pi$ to itself such that $(I, k) \in \Pi$ iff $(I', k') \in \Pi$ with $k' + |I'| \leq h(k)$ for a fixed function $h$; $h(k)$ is the size of the kernel. A kernel is polynomial if $h(k)$ is a polynomial.
FPT Result and Kernel

- For any fixed $k$, deciding if $\ell_{\text{min}}(D) \leq k$ is NP-hard.
FPT Result and Kernel

- For any fixed $k$, deciding if $\ell_{\text{min}}(D) \leq k$ is NP-hard.

- Let $k$ be a parameter and $iv(D) = |V(D)| - \ell(D)$. Deciding if $iv_{\text{max}}(D) \geq k$ is FPT: there is an $O(k^2)$-vertex kernel and an $O^*(2^{O(k \log k)})$-algorithm for deciding if $iv_{\text{max}}(D) \geq k$. [Gutin, Razgon and Kim, 2009]
For any fixed $k$, deciding if $\ell_{\min}(D) \leq k$ is NP-hard.

Let $k$ be a parameter and $iv(D) = |V(D)| - \ell(D)$. Deciding if $iv_{\max}(D) \geq k$ is FPT: there is an $O(k^2)$-vertex kernel and an $O^*(2^{O(k \log k)})$-algorithm for deciding if $iv_{\max}(D) \geq k$. [Gutin, Razgon and Kim, 2009]

Still open: Is there an $O(k)$-vertex kernel? There is a $O(k)$-vertex kernel for acyclic [Gutin, Razgon and Kim, 2009] and symmetric [Fomin et al., 2013] digraphs.
Faster Deterministic and Randomized Algorithms

- $O^*(55.8^k)$ [det, Cohen et al. 2010], $O^*(4^k)$ [random, Daligault, 2011],
- $O^*(16^k(1+o(1)))$ [det, Fomin et al., 2012], $O^*(6.855^k)$ [det, Shachnai and Zehavi, 2015],
- $O^*(5.139^k)$ [det, Zehavi, 2016], $O^*(3.617^k)$ [random, Zehavi, 2015],
- $O^*(2^k)$ [random, Björklund, Kaski and Koutis, 2017],
- $O^*(3.41^k)$ [det, Gutin, Reidl, Wahlström and Zehavi, 2018].
Outline

1. Introduction
2. Maximum Leaf Out-branchings
3. Minimum Leaf Out-branchings
4. Fast P-Space Algorithm for Out-Branchings with at Least \( k \) Internal Vertices
Algorithm 1

$O^*(3.86^k)$-time and $O^*(1)$-space deterministic algorithm for deciding an out-branching with at least $k$ internal vertices [Gutin, Reidl, Wahlström and Zehavi, JCSS 95(1), 2018].
Algorithm 1

- $O^*(3.86^k)$-time and $O^*(1)$-space deterministic algorithm for deciding an out-branching with at least $k$ internal vertices [Gutin, Reidl, Wahlström and Zehavi, JCSS 95(1), 2018].

- Matrix Tree Theorem: $\det(K_r) = \sum_{B \in B_r} \prod_{ij \in A(B)} x_{ij}$. 

Gregory Gutin Branchings in Digraphs
Algorithm 1

- $O^*(3.86^k)$-time and $O^*(1)$-space deterministic algorithm for deciding an out-branching with at least $k$ internal vertices [Gutin, Reidl, Wahlström and Zehavi, JCSS 95(1), 2018].

- Matrix Tree Theorem: $\det(K_\bar{r}) = \sum_{B \in \mathcal{B}_r} \prod_{ij \in A(B)} x_{ij}$.

- Fix $r$. Set $x_{ij} = x_i$. Let $\mathcal{B}_{r,k} = \{ B \in \mathcal{B}_r : iv(B) \geq k \}$. $\exists B \in \mathcal{B}_{r,k}$ iff $\det(K_\bar{r})$ has a monomial with at least $k$ distinct $x_i$'s.
**Algorithm 1**

- \( O^*(3.86^k) \)-time and \( O^*(1) \)-space deterministic algorithm for deciding an out-branching with at least \( k \) internal vertices [Gutin, Reidl, Wahlström and Zehavi, JCSS 95(1), 2018].

- **Matrix Tree Theorem**: 
  \[
  \det(K_r) = \sum_{B \in \mathcal{B}_r} \prod_{ij \in A(B)} x_{ij}. 
  \]

- Fix \( r \). Set \( x_{ij} = x_i \). Let \( \mathcal{B}_{r,k} = \{ B \in \mathcal{B}_r : \text{iv}(B) \geq k \} \).
  \( \exists B \in \mathcal{B}_{r,k} \) iff \( \det(K_r) \) has a monomial with at least \( k \) distinct \( x_i \)'s.

- To check it efficiently, we use efficient color coding and monomial sieving. \( k \)-coloring: \( \{x_1, \ldots, x_n\} \rightarrow \{y_1, \ldots, y_k\} \).
Algorithm 2

- **M-Lemma:** (i) Let $T$ be an out-tree s.t. $iv(T) \geq k$. Then $T$ has a matching of size $\geq k/2$; (ii) Let $M$ be a matching in $D$. In $O^*(1)$ time, we can find an out-branching $B$ of $D$ s.t. every arc of $M$ has at least one vertex as an internal vertex in $B$. 
Algorithm 2

- **M-Lemma**: (i) Let $T$ be an out-tree s.t. $iv(T) \geq k$. Then $T$ has a matching of size $\geq k/2$; (ii) Let $M$ be a matching in $D$. In $O^*(1)$ time, we can find an out-branching $B$ of $D$ s.t. every arc of $M$ has at least one vertex as an internal vertex in $B$.

- Let $M$ be a maximum matching in $D$ of size $t$. By M-Lemma, $k/2 \leq t \leq k$. 


Algorithm 2

- **M-Lemma**: (i) Let $T$ be an out-tree s.t. $iv(T) \geq k$. Then $T$ has a matching of size $\geq k/2$; (ii) Let $M$ be a matching in $D$. In $O^*(1)$ time, we can find an out-branching $B$ of $D$ s.t. every arc of $M$ has at least one vertex as an internal vertex in $B$.

- Let $M$ be a maximum matching in $D$ of size $t$. By M-Lemma, $k/2 \leq t \leq k$.

- For every $c \in \{0, 1, \ldots, k\}$ consider all sets $M'$ of $c$ arcs in $M$ in which both vertices are leaves in some $B \in B_{r,k}$. For every such vertex $i$, $x_i$ gets its own $y_j$. 

---

Gregory Gutin  
Branchings in Digraphs
Algorithm 2

- **M-Lemma**: (i) Let $T$ be an out-tree s.t. $iv(T) \geq k$. Then $T$ has a matching of size $\geq k/2$; (ii) Let $M$ be a matching in $D$. In $O^*(1)$ time, we can find an out-branching $B$ of $D$ s.t. every arc of $M$ has at least one vertex as an internal vertex in $B$.

- Let $M$ be a maximum matching in $D$ of size $t$. By M-Lemma, $k/2 \leq t \leq k$.

- For every $c \in \{0,1,\ldots,k\}$ consider all sets $M'$ of $c$ arcs in $M$ in which both vertices are leaves in some $B \in \mathcal{B}_{r,k}$. For every such vertex $i$, $x_i$ gets its own $y_j$.

- For $ip \in M \setminus M'$, $x_i, x_p$ get one $y_j$. 
Algorithm 3

- Every other $x_i$ gets a random $y_j$ out of the remaining $k - t - c$ ones. Derandomization via a perfect hash family.
Algorithm 3

- Every other $x_i$ gets a random $y_j$ out of the remaining $k - t - c$ ones. Derandomization via a perfect hash family.

- Sieving Lemma allows to decide if $\det(K_{\bar{F}}(y_1, \ldots, y_k))$ has a monomial with all $y_1, \ldots, y_k$ in time $O^*(2^k)$. 

\[ f(k) = O^*(3^{0.857k}) \]
Algorithm 3

- Every other $x_i$ gets a random $y_j$ out of the remaining $k - t - c$ ones. Derandomization via a perfect hash family.

- Sieving Lemma allows to decide if $\det(K_{\bar{r}}(y_1, \ldots, y_k))$ has a monomial with all $y_1, \ldots, y_k$ in time $O^*(2^k)$.

- Exp. part of runtime $f(k) = \sum_{c=0}^{k-t} \binom{t}{c} e^{(k-t-c)(1+o(1))} 2^k$. 

Gregory Gutin  Branchings in Digraphs
Algorithm 3

- Every other $x_i$ gets a random $y_j$ out of the remaining $k - t - c$ ones. Derandomization via a perfect hash family.

- Sieving Lemma allows to decide if $\det(K_{\bar{r}}(y_1, \ldots, y_k))$ has a monomial with all $y_1, \ldots, y_k$ in time $O^*(2^k)$.

- Exp. part of runtime $f(k) = \sum_{c=0}^{k-t} \binom{t}{c} e^{(k-t-c)(1+o(1))} 2^k$.

- $f(k) = O^*(3.857^k)$. 

Gregory Gutin
Branchings in Digraphs
Questions

- Questions?
- Comments?