Maximal number of leaves in induced subtrees

Élise Vandomme
Joint work with A. Blondin Massé, J. de Carufel, A. Goupil, M. Lapointe & É. Nadeau

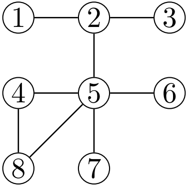
Université du Québec à Montréal

May 7, 2018
Optimization Days, HEC Montreal

Definition

For a graph $G = (V, E)$ and $n \geq 0$

- $\mathcal{T}_n$ = set of induced subtrees with n vertices
- $L_G(n) = \max\{\#\text{ leaves in } T | T \in \mathcal{T}_n\}$
- **Leaf sequence** of G: $L_G(n)_{n \in \{0,\ldots,|V|\}}$

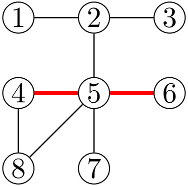


n	0	1	2	3	4	5	6	7	8
$L_G(n)$	0	0	2						

Definition

For a graph $G = (V, E)$ and $n \geq 0$

- $\mathcal{T}_n$ = set of induced subtrees with n vertices
- $L_G(n) = \max\{\#\text{ leaves in } T | T \in \mathcal{T}_n\}$
- **Leaf sequence** of G: $L_G(n)_{n \in \{0,\ldots,|V|\}}$

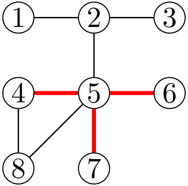


n	0	1	2	3	4	5	6	7	8
$L_G(n)$	0	0	2	2					

Definition

For a graph $G = (V, E)$ and $n \geq 0$

- $\mathcal{T}_n$ = set of induced subtrees with n vertices
- $L_G(n) = \max\{\#\text{ leaves in } T | T \in \mathcal{T}_n\}$
- **Leaf sequence** of G: $L_G(n)_{n\in\{0,\ldots,|V|\}}$

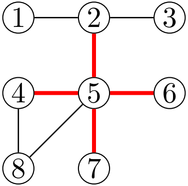


n	0	1	2	3	4	5	6	7	8
$L_G(n)$	0	0	2	2	3				

Definition

For a graph $G = (V, E)$ and $n \geq 0$

- $\mathcal{T}_n$ = set of induced subtrees with n vertices
- $L_G(n) = \max\{\#\text{ leaves in } T | T \in \mathcal{T}_n\}$
- **Leaf sequence** of G: $L_G(n)_{n\in\{0,\ldots,|V|\}}$

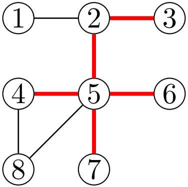


n	0	1	2	3	4	5	6	7	8
$L_G(n)$	0	0	2	2	3	4			

Definition

For a graph $G = (V, E)$ and $n \geq 0$

- $\mathcal{T}_n$ = set of induced subtrees with n vertices
- $L_G(n) = \max\{\#\text{ leaves in } T | T \in \mathcal{T}_n\}$
- **Leaf sequence** of G: $L_G(n)_{n\in\{0,\ldots,|V|\}}$

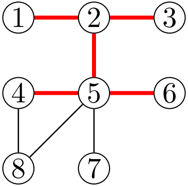


n	0	1	2	3	4	5	6	7	8
$L_G(n)$	0	0	2	2	3	4	4		

Definition

For a graph $G = (V, E)$ and $n \geq 0$

- $\mathcal{T}_n$ = set of induced subtrees with n vertices
- $L_G(n) = \max\{\#\text{ leaves in } T | T \in \mathcal{T}_n\}$
- **Leaf sequence** of G: $L_G(n)_{n \in \{0,\ldots,|V|\}}$

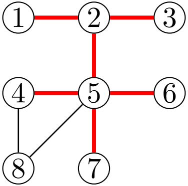


n	0	1	2	3	4	5	6	7	8
$L_G(n)$	0	0	2	2	3	4	4		

Definition

For a graph $G = (V, E)$ and $n \geq 0$

- $\mathcal{T}_n$ = set of induced subtrees with n vertices
- $L_G(n) = \max\{\#\text{ leaves in } T | T \in \mathcal{T}_n\}$
- **Leaf sequence** of G: $L_G(n)_{n\in\{0,\ldots,|V|\}}$

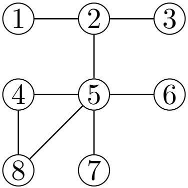


n	0	1	2	3	4	5	6	7	8
$L_G(n)$	0	0	2	2	3	4	4	5	

Definition

For a graph $G = (V, E)$ and $n \geq 0$

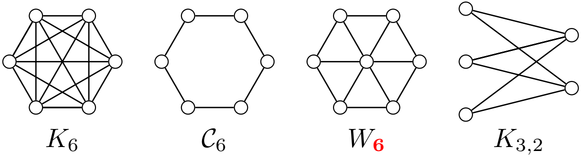
- $\mathcal{T}_n$ = set of induced subtrees with n vertices
- $L_G(n) = \max\{\#\text{ leaves in } T | T \in \mathcal{T}_n\}$
- **Leaf sequence** of G: $L_G(n)_{n\in\{0,\ldots,|V|\}}$



n	0	1	2	3	4	5	6	7	8
$L_G(n)$	0	0	2	2	3	4	4	5	$-\infty$

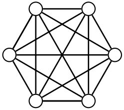
Particular cases

- Leaf sequences known for $K_p, \mathcal{C}_p, W_p, K_{p,q}$

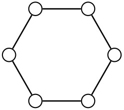


Particular cases

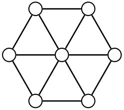
- Leaf sequences known for $K_p, \mathcal{C}_p, W_p, K_{p,q}$



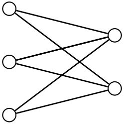
K_6



$\mathcal{C}_6$

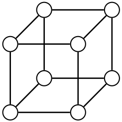


$W_{\mathbf{6}}$

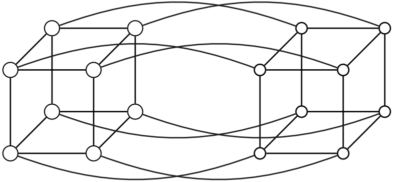


$K_{3,2}$

- Leaf sequences known for H_d with $d \leq 6$



H_3



H_4

Observations

- The leaf sequence can increase by at most 1 each step.
- The leaf sequence is not always non-decreasing.

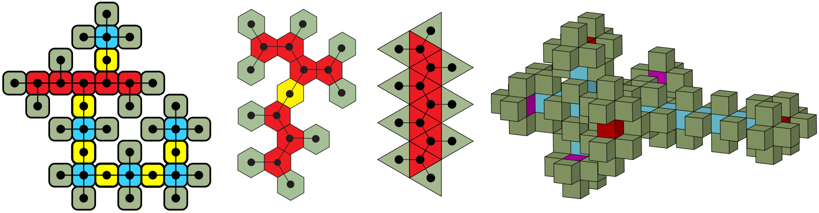
Observations

- The leaf sequence can increase by at most 1 each step.
- The leaf sequence is not always non-decreasing.
- The leaf sequence $L_G(n)_{n \in \{2,\dots,|V|\}}$ is non-decreasing iff G is a tree.

Particular cases: infinite lattices

Theorem (Blondin Massé, de Carufel, Goupil, Samson 2017)

The leaf sequences of the square, triangular, hexagonal and cubic infinite lattices satisfy linear recurrences with asymptotic growth $n/2$ for planar lattices, $28n/41$ for the cubic lattice.



Complexity

Problem (LIS)

- Instance: a graph G and two integers n, ℓ
- Question: Is there an induced subtree of G with n vertices and ℓ leaves?

Theorem (Blondin Massé, de Carufel, Goupil, Lapointe, Nadeau, V. 2018)
The LIS problem is NP-complete.

Complexity

Problem (LIS)

- Instance: a graph G and two integers n, ℓ
- Question: Is there an induced subtree of G with n vertices and ℓ leaves?

Theorem (Blondin Massé, de Carufel, Goupil, Lapointe, Nadeau, V. 2018)

The LIS problem is NP-complete.

Reduction to the NP-complete problem Independent Set:

- Instance: a graph G and an integer $k \geq 1$
- Question: Is there a subset S of vertices that are not 2-by-2 adjacent such that $|S| = k$?

Optimization problem

Problem (MLIS)

- Instance : a graph G
- Question : What is the leaf sequence $L_G(n)_{n\in\{0,\dots,|V|\}}$ of G?

$\rightsquigarrow$ Branch and bound algorithm

Elements of the search space

Consider

- a graph $G = (V, E)$
- colorings $V \to \{green, yellow, red, blue\}$

such that any coloring c satisfies : for all $u, v \in V$

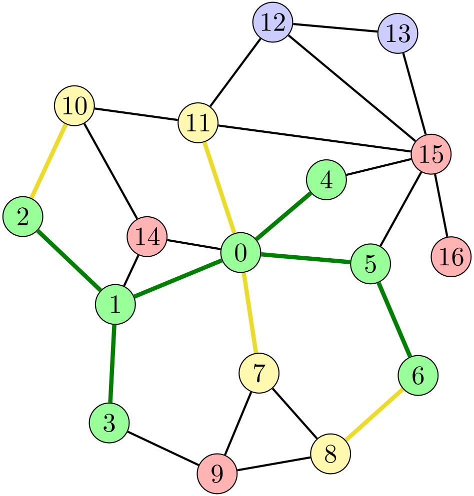
(i) The subgraph induced by $c^{-1}(green)$ is a tree

(ii) $c(u) = green$ and $\{u, v\} \in E \Rightarrow c(v) \in \{green, yellow, red\}$

(iii) $c(u) = yellow \Rightarrow |c^{-1}(green) \cap N(u)| = 1$, where $N(u)$ is the neighborhood of u.

A **configuration** of G is a pair $C = (c, H)$ where c is a coloring and H is a stack of colorings called **historic** of C.

Exemple de configuration



Branching

The initial configuration is $(c_{blue}, \emptyset)$.

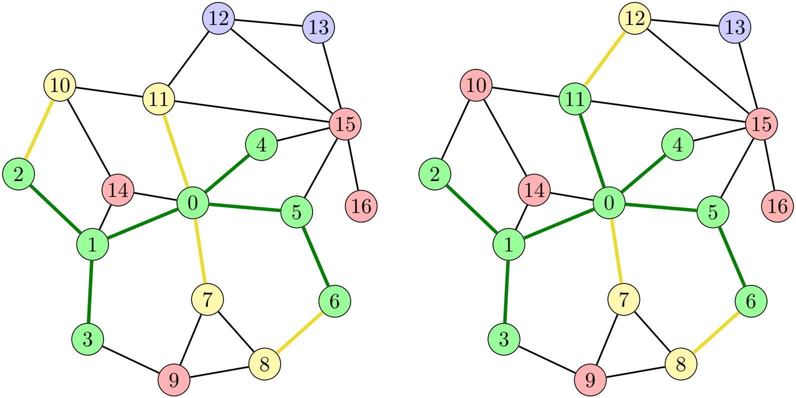
- C.AddToSubtree(v) pushes a copy of c on the stack H, turns the color of v into green, update the colors of the neighbors of v.
- C.ExcludeVertex(v) pushes a copy of c on the stack H, turns the color of v into red.
- C.Undo() modifies c into the configuration on the top of the stack H and deletes it from H.

How to choose v

- C.VertexToAdd() is a non-deterministic function which returns any blue or yellow vertex that can be turned into green without creating any problem. If no such vertex exists, it returns **none**.

How to choose v

- C.VertexToAdd() is a non-deterministic function which returns any blue or yellow vertex that can be turned into green without creating any problem. If no such vertex exists, it returns **none**.



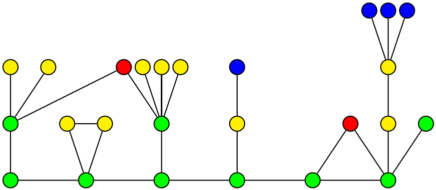
Bounding

C.LEAFPOTENTIAL(n') = upper bound on the number of leaves that can be reached by extending the configuration C (with n green vertices and r red vertices) into a configuration C' with n' green vertices (defined for all n' between n and $|V| - r$).

How to compute the potential?

- Discard red vertices
- Partition vertices according to the distance to the inner green vertices
- Be optimistic while extending the green subgraph

C.LeafPotential(n')

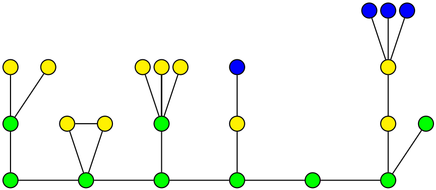


$n = 9$

#leaves $= 3$

$n' = 20$

C.LeafPotential(n')

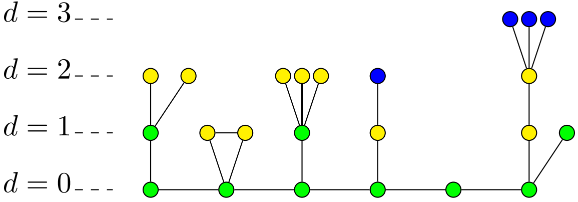


$n = 9$

#leaves $= 3$

$n' = 20$

C.LeafPotential(n')

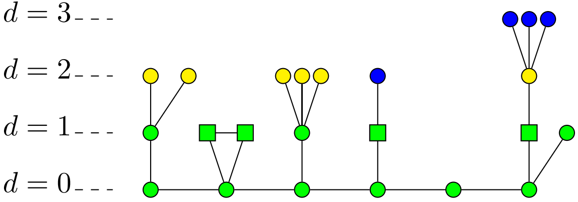


$n = 9$

$n' = 20$

#leaves $= 3$

$C.\text{LeafPotential}(n')$

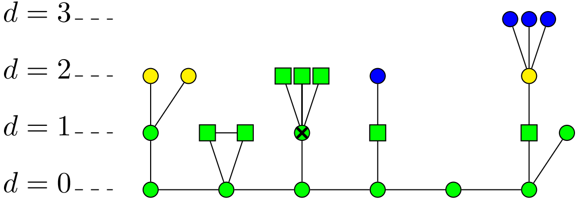


Completion

$n = 9 \quad \Rightarrow 13 \qquad\qquad n' = 20$

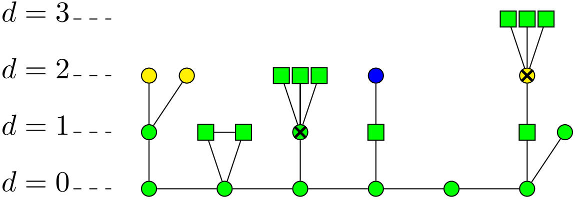
$\#\text{leaves} = 3 \quad \Rightarrow 7$

C.LeafPotential(n')



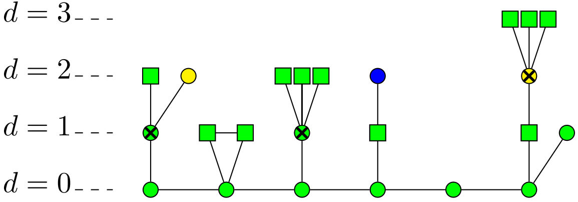
	Completion	$d = 1$	
$n = 9$	$\Rightarrow 13$	$\Rightarrow 16$	$n' = 20$
#leaves $= 3$	$\Rightarrow 7$	$\Rightarrow 9$	

$C.$LeafPotential(n')



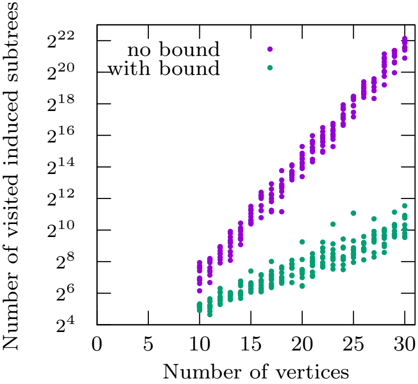
	Completion	$d = 1$	$d \leq 2$	
$n = 9$	$\Rightarrow 13$	$\Rightarrow 16$	$\Rightarrow 19$	$n' = 20$
$\#\text{leaves} = 3$	$\Rightarrow 7$	$\Rightarrow 9$	$\Rightarrow 11$	

C.LeafPotential(n')



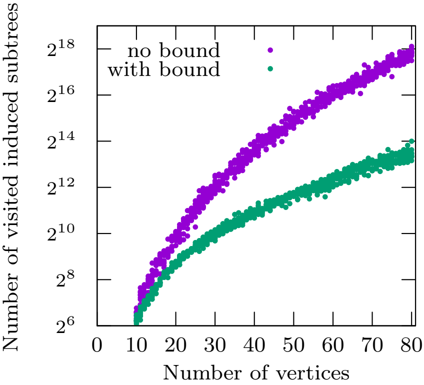
	Completion	$d = 1$	$d \leq 2$	$d \leq 3$	
$n = 9$	$\Rightarrow 13$	$\Rightarrow 16$	$\Rightarrow 19$	$\Rightarrow 20$	$n' = 20$
$\#\text{leaves} = 3$	$\Rightarrow 7$	$\Rightarrow 9$	$\Rightarrow 11$	$\Rightarrow 11$	

Efficiency



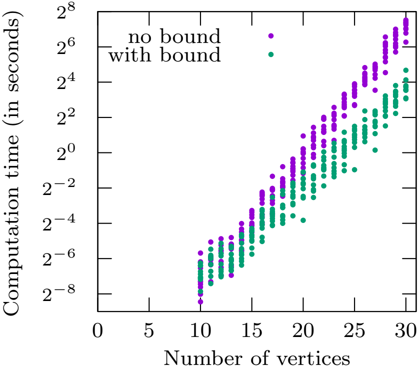
Edge Probability 0.2

Efficiency



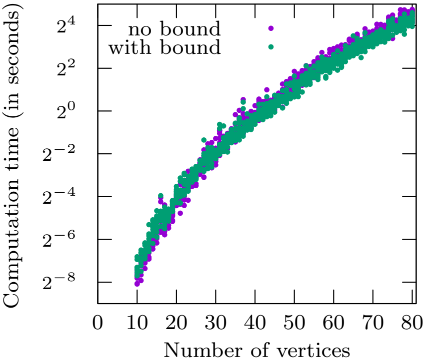
Edge Probability 0.8

Efficiency



Edge Probability 0.2

Efficiency



Edge Probability 0.8

What about trees?

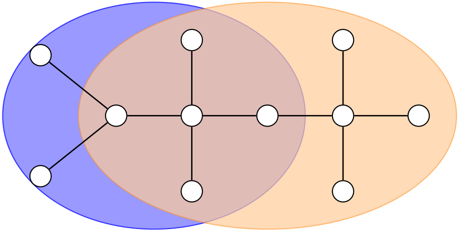
Theorem
(Blondin Massé, de Carufel, Goupil, Lapointe, Nadeau, V. 2018)

For a tree T with m vertices, the leaf sequence $L_T(n)_{n\in\{0,\ldots,m\}}$ is computed in $\mathcal{O}(m^3\Delta)$ time and $\mathcal{O}(m^2)$ space, where Δ is the maximal degree.

Algorithm based on the dynamic programming paradigm

Could we improve the time and space complexity?

- We cannot hope to obtain a procedure computing $L_T(n)$ which deletes leaves successively.
- Counter-example :



$$L_T(7) = 5 \quad \text{et } L_T(9) = 6$$

For more details

- Both algorithms, implemented in SageMath, are available: `github.com/enadeau/fully-leafed-induced-subtrees`
- A. Blondin Massé, J. de Carufel, A. Goupil, M. Lapointe, É. Nadeau & É. Vandomme (2018). *Fully leafed induced subtrees*. Proceedings of the 29th International Workshop on Combinatorial Algorithms, LNCS. To appear. `arxiv.org/abs/1709.09808`