On the Maximal Number of Leaves in Induced Subtrees of Series-Parallel Graphs

M. Abdenbi, **A. Blondin Massé** and A. Goupil

LaCIM
Université du Québec à Montréal

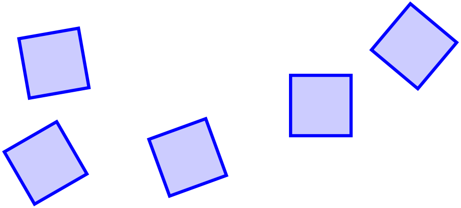
June 20, 2018
GASCom 2018

Content

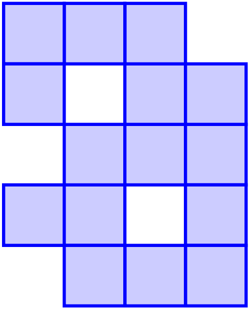
Content

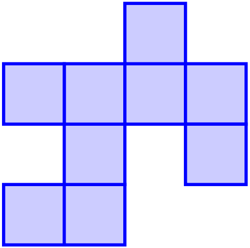
The Initial Question

Alain Goupil: Alexandre, if I give you n **unit squares** to build a (tree) polyomino, what is the maximum number of **leaves** that you can obtain?

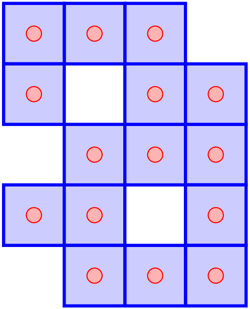


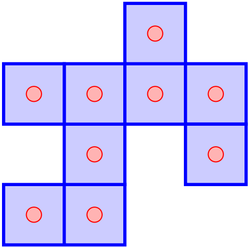
Tree Polyominoes



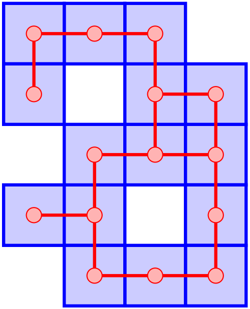


Tree Polyominoes

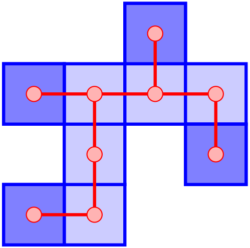




Tree Polyominoes



Not a tree!



A tree with **4** leaves!

The Polyomino Case

n	Example	ℓ
1		
2		
3		
4		

n	Example	ℓ
5		
6		
7		

The Polyomino Case

n	Example	ℓ
1		0
2		
3		
4		

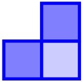
n	Example	ℓ
5		
6		
7		

The Polyomino Case

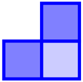
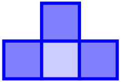
n	Example	ℓ
1		0
2		2
3		
4		

n	Example	ℓ
5		
6		
7		

The Polyomino Case

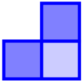
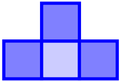
n	Example	ℓ
1		0
2		2
3		2
4		

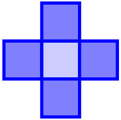
n	Example	ℓ
5		
6		
7		

n	Example	ℓ
1		0
2		2
3		2
4		3

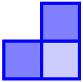
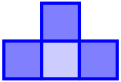
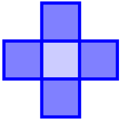
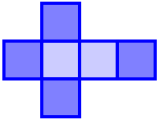
n	Example	ℓ
5		
6		
7		

The Polyomino Case

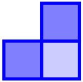
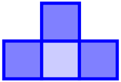
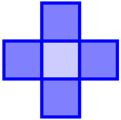
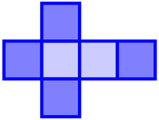
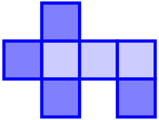
n	Example	ℓ
1		0
2		2
3		2
4		3

n	Example	ℓ
5		4
6		
7		

The Polyomino Case

n	Example	ℓ
1		0
2		2
3		2
4		3
5		4
6		4
7		

The Polyomino Case

n	Example	ℓ
1		0
2		2
3		2
4		3
5		4
6		4
7		4

Leaf Function

- Let $\mathcal{T}(n)$ be the set of **all tree polyominoes of n cells**.
- Let

$$L_{\text{Squ}}(n) = \max\{\#\text{ of leaves of } P \mid P \in \mathcal{T}(n)\}.$$

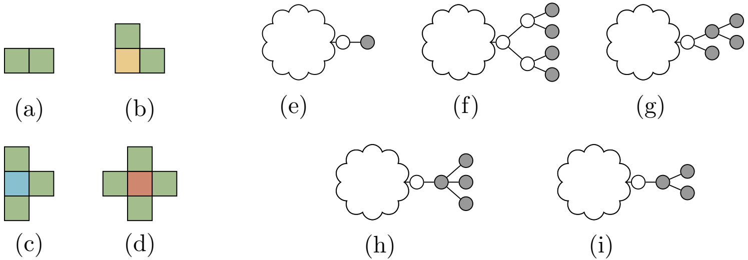
- (B. M., de Carufel, Goupil, Samson, 2017) Then

$$L_{\text{Squ}}(n) = \begin{cases} 0, & \text{if } n = 0, 1; \\ 2, & \text{if } n = 2; \\ n - 1, & \text{if } n = 3, 4, 5; \\ L_{\text{Squ}}(n-4) + 2, & \text{if } n \geq 6. \end{cases}$$

- First values

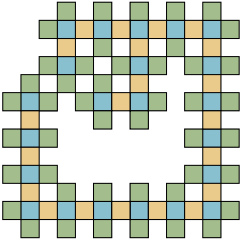
n	0	1	2	3	4	5	6	7	8	9	10	11	$\cdots$
$L_{Squ}(n)$	0	0	2	2	3	4	4	4	5	6	6	6	$\cdots$

Idea of the proof

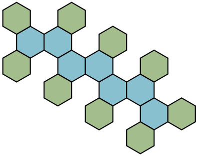


- **Lower bound**: by construction;
- **Upper bound**: by minimum counter-example.

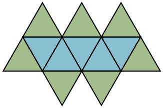
Polyominoes



Polyhexes

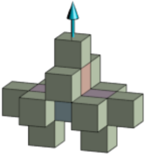


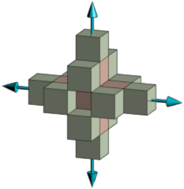
Polyiamonds



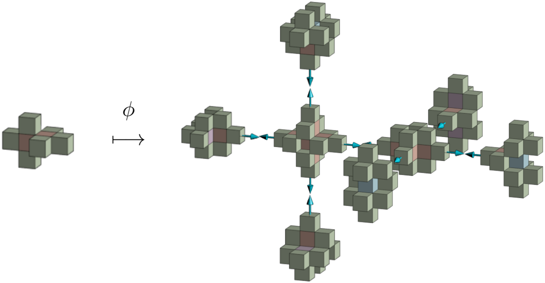
$$L_{\text{Tri}}(n) = L_{\text{Hex}} = \begin{cases} 0, & \text{if } n = 0, 1; \\ 2, & \text{if } n = 2, 3; \\ L_{\text{Tri}}(n-2) + 1, & \text{if } n \geq 4. \end{cases}$$

Cubic Lattice





$n = 15, \ell = 11$ $\quad$ $n = 17, \ell = 12$



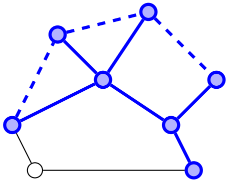
Generalization

- Clearly, the 2D lattices and cubic lattice seen before are **particular cases** of a more general problem;
- More precisely, those lattices are **infinite simple graphs**;
- Tree-like **polyforms** and **polycubes** correspond to **subtrees** of those graphs;
- What can be said about arbitrary **finite graphs** or even **infinite graphs**?
- Are there known results from the **graph theoretical community** about this more general problem?

Content

Problem Definition

- Let $G = (V, E)$ be a (finite) **simple graph**;
- Let $\mathcal{T}(i)$ be the set of all **induced subtrees** of G having i vertices;



Not an induced subtree!

- A induced subtree is called **fully leafed** if it has the **maximum number** of leaves for a **fixed size** i.
- This problem does not seem to have been studied before...

Spanning Trees

- (Payan et al., 1984) MLST = Maximum Leaf **Spanning Tree**;
- (Garey and Johnson, 1979) MLST is **NP-hard**;
- (Boukerche, Cheng and Linus, 2005) Application of MLST to energy-aware networks;
- The problem of identifying **fully leafed induced subtrees** is also **NP-hard**;
- However, replacing **"spanning tree"** by **"induced subtree"** yields an additional complexity with respect to **parametrized algorithms**.

Hardness of the Problem

Theorem (B.M., de Carufel, Goupil, Lapointe, Nadeau, Vandomme, 2018)

Given a simple graph G, the problem of deciding whether there exists an induced subtree of G with i vertices and ℓ leaves is **NP-complete**.

Proof.

Reduction to the **Independent Set** problem, by adding a **universal vertex** to the graph. □

Leaf Function of Usual Families (1/2)

Complete graph:

$$L_{K_n}(i) = \begin{cases} 0, & \text{if } i = 0, 1; \\ 2, & \text{if } i = 2; \\ -\infty, & \text{if } 3 \leq i \leq n; \end{cases}$$

Cycle graph:

$$L_{\mathcal{C}_n}(i) = \begin{cases} 0, & \text{if } i = 0, 1; \\ 2, & \text{if } 2 \leq i < n; \\ -\infty, & \text{if } i = n. \end{cases}$$

Leaf Function of Usual Families (2/2)

Wheel graph:

$$L_{W_n}(i) = \begin{cases} 0, & \text{if } i = 0, 1; \\ 2, & \text{if } i = 2; \\ i - 1, & \text{if } 3 \leq i \leq \lfloor \frac{n}{2} \rfloor + 1; \\ 2, & \text{if } \lfloor \frac{n}{2} \rfloor + 2 \leq i \leq n - 1; \\ -\infty, & \text{if } n \leq i \leq n + 1. \end{cases}$$

Complete bipartite graph:

$$L_{K_{p,q}}(i) = \begin{cases} 0, & \text{if } i = 0, 1; \\ 2, & \text{if } i = 2; \\ i - 1, & \text{if } 3 \leq i \leq \max(p, q) + 1; \\ -\infty, & \text{if } \max(p, q) + 2 \leq i \leq p + q. \end{cases}$$

- Highly **symmetric**;
- But surprisingly more **intricate**;
- A related (hard) problem is called **snake-in-the-box**;
- Obtained with a **branch-and-bound** algorithm;
- Some **symmetries** have been exploited, but not all!

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$L_{Q_2}(n)$	0	0	2	2	*													
$L_{Q_3}(n)$	0	0	2	2	3	2	*	*	*									
$L_{Q_4}(n)$	0	0	2	2	3	4	3	4	3	4	*	*	*	*	*	*	*	
$L_{Q_5}(n)$	0	0	2	2	3	4	5	4	5	6	6	6	7	7	7	8	8	8
$L_{Q_6}(n)$	0	0	2	2	3	4	5	6	5	6	7	8	8	9	9	10	10	11
n	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	...
$L_{Q_5}(n)$	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*			
$L_{Q_6}(n)$	11	12	12	13	13	14	14	15	15	16	16	17	17	18	18	18	*	...

The Tree Case

- The problem becomes **polynomial** when the considered graph is itself a **tree**;
- **Note:** In that case, **subtree** = **induced subtree**.

Theorem (B.M., de Carufel, Goupil, Lapointe, Nadeau, Vandomme, 2018)

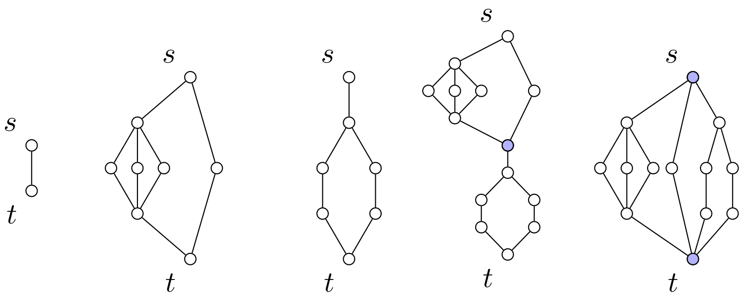
Let $T = (V, E)$ be an undirected tree with $n \geq 2$ vertices. Then L_T can be computed in $\mathcal{O}(n^3\Delta)$ time and $\mathcal{O}(n^2)$ space where Δ denotes the **maximal degree** of a vertex in T.

Parametrized Algorithms

- In **general**, the problem is **hard**;
- In the case of **trees**, it becomes **polynomial**;
- This suggests that a **parametrized algorithm** could be polynomial assuming that a parameter called the **tree-width** of the graph is **bounded**.
- Intuitively, **tree-width** indicates how **"close"** to a tree any given graph is;
- Graphs of tree-width equal to 1: **series-parallel graphs**!

Content

Series-Parallel Graphs



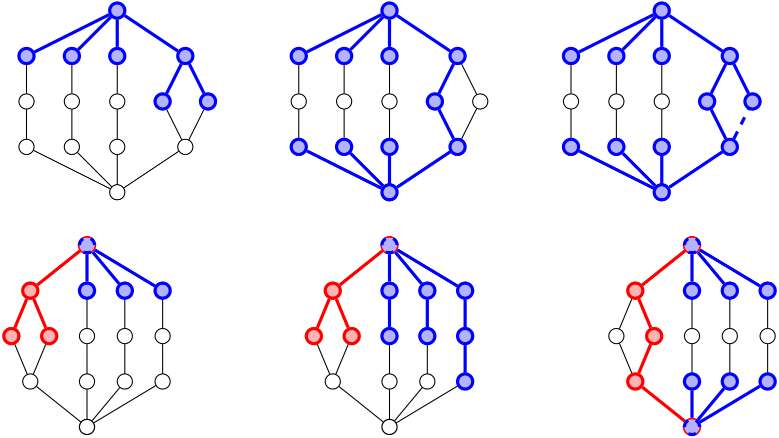
(a) (b) (c) (d) (e)

- (MacMahon, 1890) Number of SP-graphs of given size (A000084).
- (Riordan and Shannon, 1942) Provided a formal proof of the recursive formula given by MacMahon.

Recursive Optimality

- Series-parallel graphs are built **recursively**;
- Is it possible that **fully leafed induced subtrees** are obtained by **"merging"** two **fully leafed induced subtrees**?
- The basic idea is **correct**;
- However, we have to take into account **special induced subforests**...

Induced Subtrees and Induced Subforests



Leaf Function of SP Graphs

It only remains to write **all possible cases**:

- With respect to the two type of compositions: **series**/**parallel**;
- With respect to the following **four** cases:
  - **Both** s **and** t are included in the induced subtree;
  - **Only** s is included in the induced subtree;
  - **Only** t is included in the induced subtree;
  - **Neither** s **nor** t is included in the induced subtree;
- We must also keep track of the **proximity** of the subtrees to either s or t to prevent the creation of **cycles**.

Enriched Leaf Function (1/2)

Let $G = (V, E, s, t)$ be an SP-graph and i an integer, with $2 \leq i \leq |G|$. We denote by $L(G, i, \sigma, \tau)$ the maximum number of leaves that can be realized by an induced subtree T of **size** i of G, where the parameters σ and τ are defined by

$$\sigma = \begin{cases} 0, & \text{if } s \in T, \deg_T(s) > 1; \\ 1, & \text{if } s \in T, \deg_T(s) = 1; \\ 2, & \text{if } s \notin T, |N(s) \cap T| \neq 0; \\ 3, & \text{if } s \notin T, |N(s) \cap T| = 0, \end{cases}$$

and

$$\tau = \begin{cases} 0, & \text{if } t \in T, \deg_T(t) > 1; \\ 1, & \text{if } t \in T, \deg_T(t) = 1; \\ 2, & \text{if } t \notin T, |N(t) \cap T| \neq 0; \\ 3, & \text{if } t \notin T, |N(t) \cap T| = 0. \end{cases}$$

Enriched Leaf Function (2/2)

Similarly, let $F(G, i, \sigma, \tau)$ be the maximum number of leaves that can be realized by an **induced subforest** of size i whose two connected components T_s and T_t containing s and t respectively are such that

$$\sigma = \begin{cases} 0, & \text{if } \deg_{T_s}(s) > 1; \\ 1, & \text{if } \deg_{T_s}(s) = 1, \end{cases}$$

and

$$\tau = \begin{cases} 0, & \text{if } \deg_{T_t}(t) > 1; \\ 1, & \text{if } \deg_{T_t}(t) = 1. \end{cases}$$

Two Additional Tools

- Let

$$\mathrm{DistN} = \{2,3\} \times \{2,3\} \cup \{0,1\} \times \{3\} \cup \{3\} \times \{0,1\},$$

- $\Rightarrow$ Indicates that the subtrees are **distant** enough so that no cycle can ever be **"created"** by the composition;
- For $(a,b) \in \{0,1\} \times \{0,1\}$, let ℓ be the **leaf loss function** defined by $\ell(a,b) = a + b$
- $\Rightarrow$ Indicates the number of leaves that are **lost** after a composition.

Series Composition, tree case (1/2)

(ST1) **T is included in G_1.**

$$ST_1(G, i, \sigma, \tau) = L(G_1, i, \sigma, \tau_1),$$

where

$$\tau = \begin{cases} 2, & \text{if } \{s,t\} \in E_2 \text{ and } \tau_1 \in \{0,1\}; \\ 3, & \text{otherwise.} \end{cases}$$

(ST2) **T is included in G_2.**

$$ST_2(G, i, \sigma, \tau) = L(G_2, i, \sigma_2, \tau)$$

where

$$\sigma = \begin{cases} 2, & \text{if } \{s,t\} \in E_1 \text{ and } \sigma_2 \in \{0,1\}; \\ 3, & \text{otherwise.} \end{cases}$$

(ST3) **T is included neither in G_1 nor G_2.**

$$ST_3(G, i, \sigma, \tau) = \max_{\substack{(i_1,i_2) \vdash i+1 \\ \tau_1, \sigma_2 \in \{0,1\}}} \{L(G_1, i_1, \sigma, \tau_1) + L(G_2, i_2, \sigma_2, \tau) - \ell(\tau_1, \sigma_2)\}$$

Series Composition, tree case (2/2)

Combining the three cases, we have

$$L(G, i, \sigma, \tau) = \max_{j \in \{1,2,3\}} \{ST_j(G, i, \sigma, \tau)\} \quad (1)$$

Series Composition, forest case (1/2)

(SF1) T_s **is included in** G_1 **and** T_t **is included in** G_2**.**

$$SF_1(G, i, \sigma, \tau) = \max_{\substack{(i_1, i_2) \vdash i \\ (\tau_1, \sigma_2) \in \text{DistN}}} \{L(G_1, i_1, \sigma, \tau_1) + L(G_2, i_2, \sigma_2, \tau)\}$$

(SF2) T_s **sticks out of** G_1 **in** G_2 **and** T_t **is included in** G_2**.**

$$SF_2(G, i, \sigma, \tau) = \max_{\substack{(i_1, i_2) \vdash i+1 \\ \tau_1, \sigma_2 \in \{0,1\}}} \{L(G_1, i_1, \sigma, \tau_1) + F(G_2, i_2, \sigma_2, \tau) - \ell(\tau_1, \sigma_2)\}$$

(SF3) T_s **is included in** G_1 **and** T_t **sticks out of** G_2 **in** G_1**.**

$$SF_3(G, i, \sigma, \tau) = \max_{\substack{(i_1, i_2) \vdash i+1 \\ \tau_1, \sigma_2 \in \{0,1\}}} \{F(G_1, i_1, \sigma, \tau_1) + L(G_2, i_2, \sigma_2, \tau) - \ell(\tau_1, \sigma_2)\}$$

Series Composition, forest case (2/2)

Combining all three cases, we obtain the expression

$$F(G, i, \sigma, \tau) = \max_{j=1,2,3} \{SF_j(G_1 \bowtie G_2, i, \sigma, \tau)\} \qquad (2)$$

Parallel Composition and Basic Cases

- Similar to the **series composition** case;
- **More cases** in particular in the case of **forest cases**;
- See the **proceedings** for the formulas (too many cases...);
- The basic cases are **easy**;

Content

Concluding Remarks

- The case of **bounded tree-width**?
- Taking into account the graph's **automorphism group**?
- Application to **mathematical chemistry**;
- Our recursive formulas can "easily" be extended to **count** and **generate** all fully leafed induced subtrees;
- Link with **species theory**?