The Complexity of Growing a Graph

Algosensors 2022

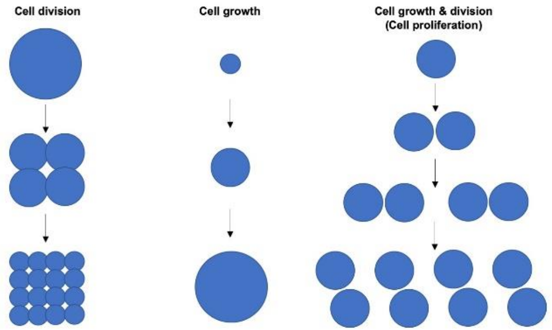
George Mertzios Othon Michail George Skretas

Paul Spirakis Michail Theofilatos

Why study growing graphs?

- Motivation: Networked systems that start from a single entity and grow into well defined structures

- Examples: social networks, sensor deployment, biological systems

- Common notion of a graph growth process which controls growth

Related Work

- No unified model seems to exist
- Literature does not focus on active growth
- Actively Dynamic Network literature: static network
- Network Costructors model: passive growth
- Random Graph Generators, Graph Editing…

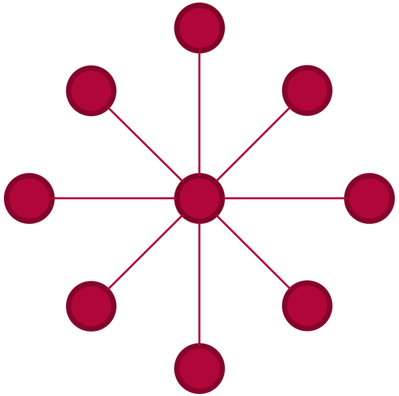
Related Work

- Systems that exhibit growth are very distinct and vary from abstract to geometrical
- *Woods et al* [ITCS’13] : Geometric and movement
- *Michail and Almalki* [Algosensors’22]: Geometric
- In this work: disregard geometry and focus on locality

Our Model

- Initial graph G_0 of a single vertex
- Centralized control
- Operations: Vertex Generation, Edge Activation at birth and Edge Deletion
- Discrete time-steps called slots
- Goal: Compute a growth process that grows G_0 into a target graph G
- Edge Activation Distance for locality constraint

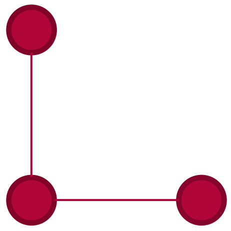
Growth Example



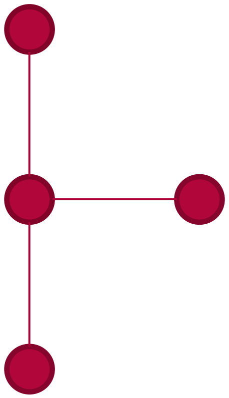
Growth Example

Growth Example

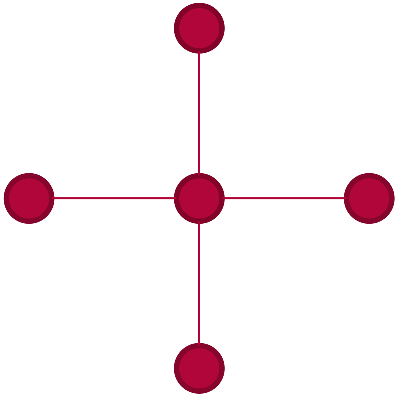
Growth Example



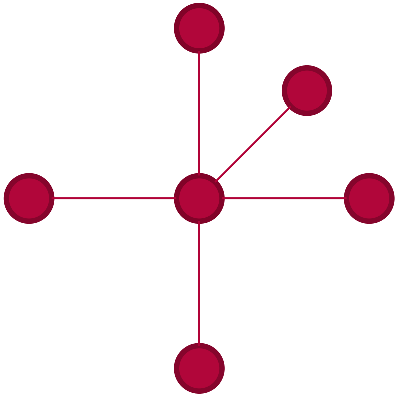
Growth Example



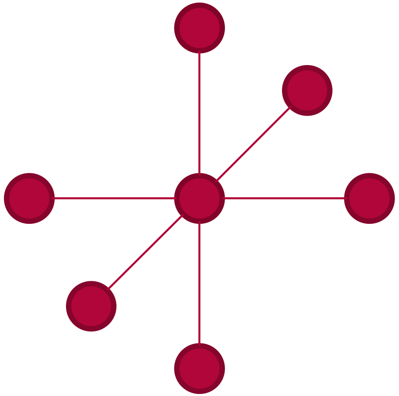
Growth Example



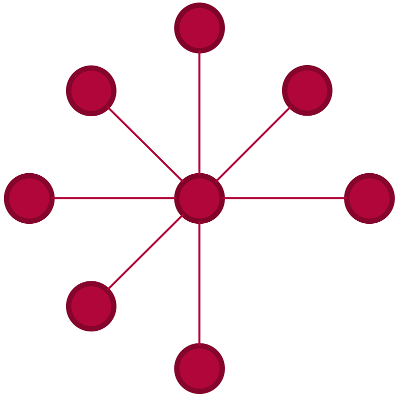
Growth Example



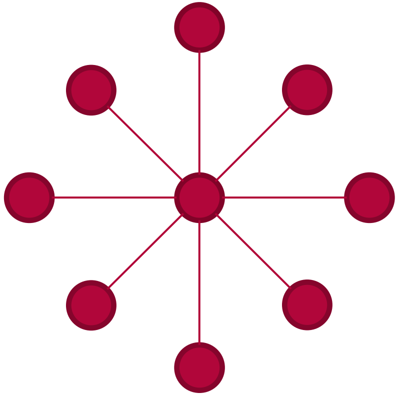
Growth Example



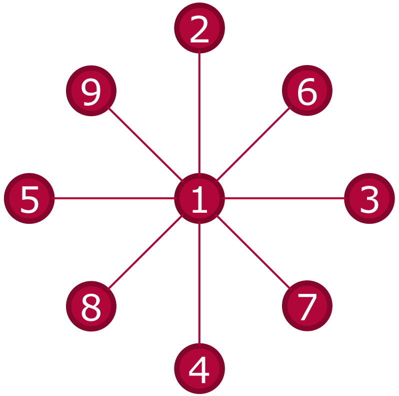
Growth Example



Growth Example



Growth Example



Growth Schedule

a) 1→2
b) 1→3
c) 1→4
d) 1→5
e) 1→6
f) 1→7
g) 1→8
h) 1→9

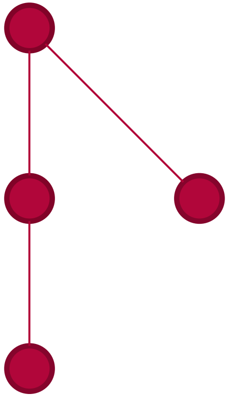
Faster Growth Example

Faster Growth Example

Growth Schedule

a) 1→2

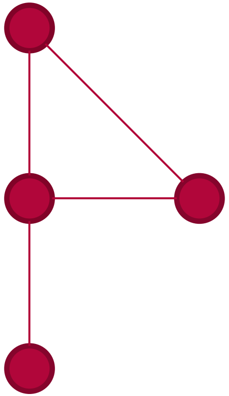
Faster Growth Example



Growth Schedule

a) 1→2
b) 1→4, 2→3

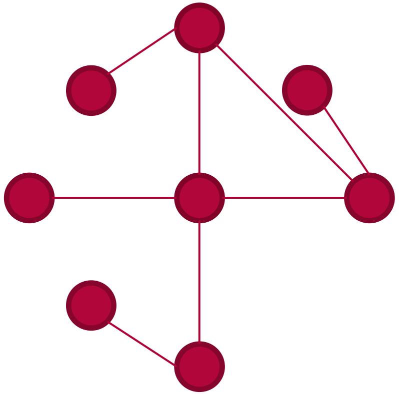
Faster Growth Example



Growth Schedule

a) 1→2
b) 1→4, 2→3 (3,1)

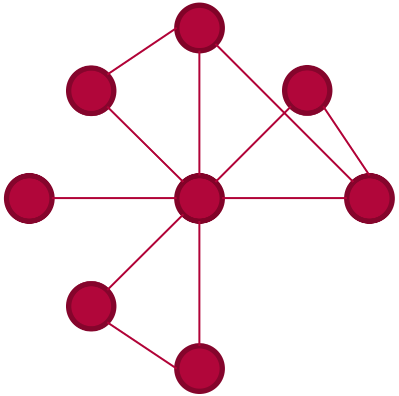
Faster Growth Example



Growth Schedule

a) 1→2
b) 1→4, 2→3 (3,1)
c) 1→5, 2→9, 3→6, 4→8

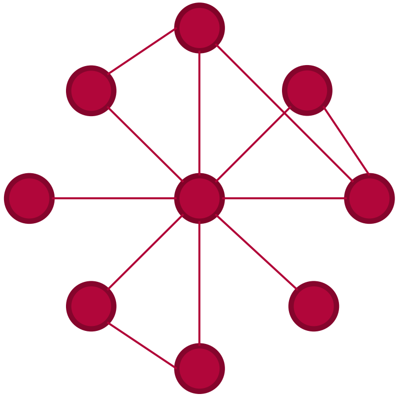
Faster Growth Example



Growth Schedule

a) 1→2
b) 1→4, 2→3 (3,1)
c) 1→5, 2→9 (9,1), 3→6 (6,1),
4→8 (8,1)

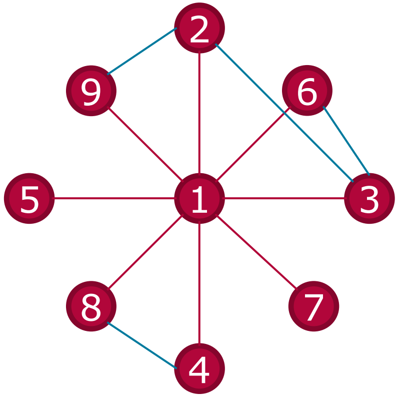
Faster Growth Example



Growth Schedule

a) 1→2
b) 1→4, 2→3 (3,1)
c) 1→5, 2→9 (9,1), 3→6 (6,1),
   4→8 (8,1)
d) 1→7

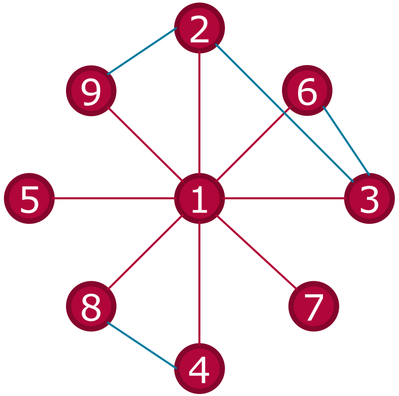
Faster Growth Example



Growth Schedule

a) 1→2
b) 1→4, 2→3 (3,1)
c) 1→5, 2→9 (9,1), 3→6 (6,1),
4→8 (8,1)
d) 1→7

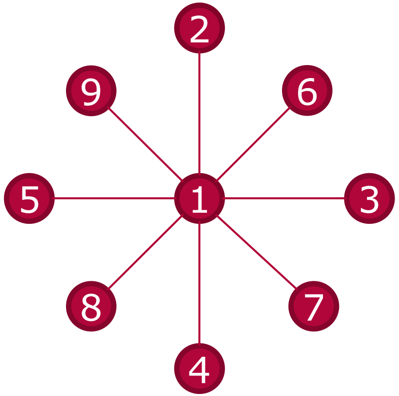
Faster Growth Example



Growth Schedule

a) 1→2
b) 1→4, 2→3 (3,1)
c) 1→5, 2→9 (9,1), 3→6 (6,1), 4→8 (8,1)
d) 1→7, (2,3) (2,9) (3,6) (4,8)

Faster Growth Example



Growth Schedule

a) $1 \rightarrow 2$
b) $1 \rightarrow 4$, $2 \rightarrow 3$ (3,1)
c) $1 \rightarrow 5$, $2 \rightarrow 9$ (9,1), $3 \rightarrow 6$ (6,1), $4 \rightarrow 8$ (8,1)
d) $1 \rightarrow 7$, (2,3) (2,9) (3,6) (4,8)

The Problem

- Trade off between slots and excess edges

a) 1→2
b) 1→3
c) 1→4
d) 1→5
e) 1→6
f) 1→7
g) 1→8
h) 1→9

a) 1→2
b) 1→4, 2→3 (3,1)
c) 1→5, 2→9 (9,1), 3→6 (6,1), 4→8 (8,1)
d) 1→7, (2,3) (2,9) (3,6) (4,8)

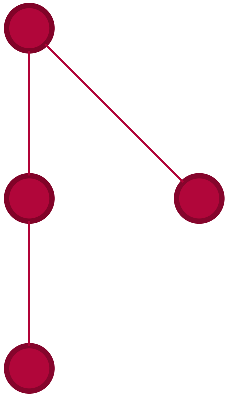
- **Graph Growth Problem**: Given an input graph G, compute in polynomial time a growth schedule with at most ***k*** slots and with at most ***l*** excess edges if it exists.

Edge Activation Distance

Growth Schedule

a) 1→2

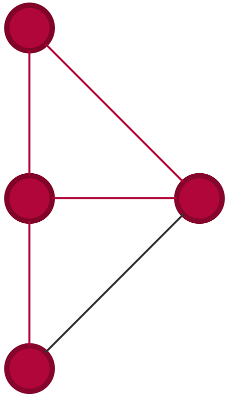
Edge Activation Distance



Growth Schedule

a) 1→2
b) 1→4, 2→3 (3,1)

Edge Activation Distance



Growth Schedule

a) 1→2
b) 1→4, 2→3 (3,1)

Edge Activation Distance

Theorem

For $d = 1$, the trimming algorithm that computes in polynomial time an optimal growth schedule with k slots for any tree graph G.

Lemma

For $d \geq 3$, a graph $G = (V, E)$ with n nodes can be generated with a growth schedule with $\log n$ slots and with $O(n)$ excess edges.

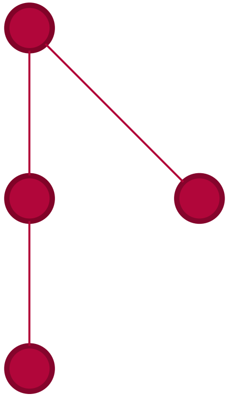
- We focused on d=2 since its more natural and interesting
- Study two edge cases: Very fast schedules of very efficient schedules

Fundamental Properties

- Independency: The vertices generated in the same time slot form an independent set in the final graph

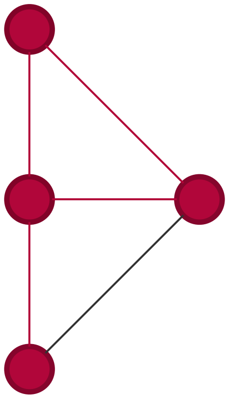
Fundamental Properties

- Independency: The vertices generated in the same time slot form an independent set in the final graph



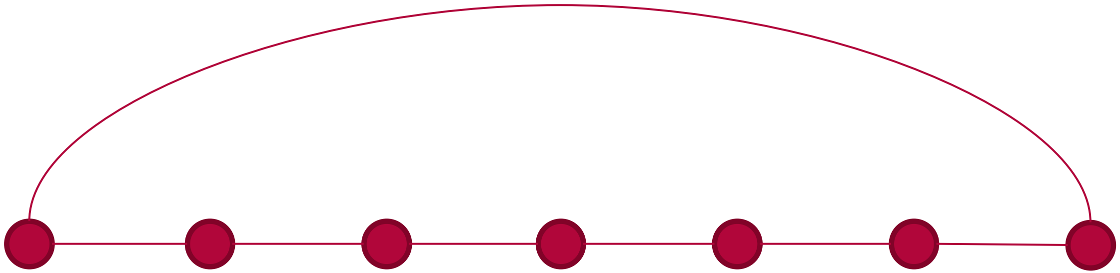
Fundamental Properties

- Independency: The vertices generated in the same time slot form an independent set in the final graph



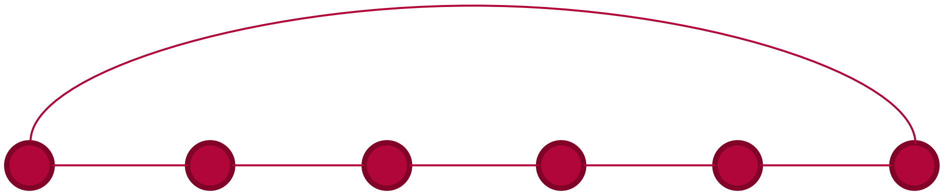
Fundamental Properties

- Independency: The vertices generated in the same time slot form an independent set in the final graph
- Inheritance: The vertices in the birth path of a vertex u must activate an edge with every neighbor v of u



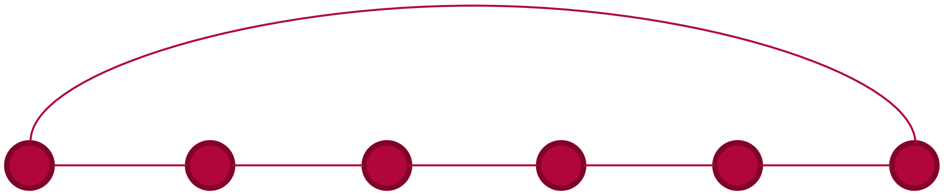
Fundamental Properties

- Independency: The vertices generated in the same time slot form an independent set in the final graph
- Inheritance: The vertices in the birth path of a vertex u must activate an edge with every neighbor v of u



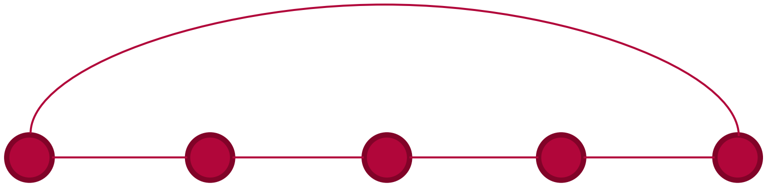
Fundamental Properties

- Independency: The vertices generated in the same time slot form an independent set in the final graph
- Inheritance: The vertices in the birth path of a vertex u must activate an edge with every neighbor v of u



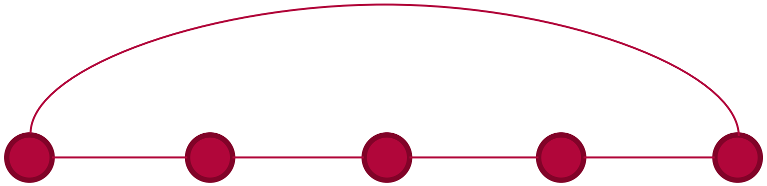
Fundamental Properties

- Independency: The vertices generated in the same time slot form an independent set in the final graph
- Inheritance: The vertices in the birth path of a vertex u must activate an edge with every neighbor v of u



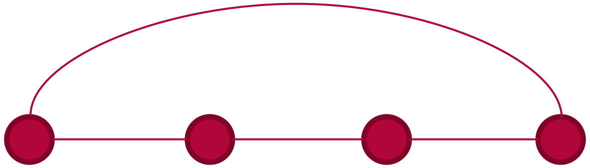
Fundamental Properties

- Independency: The vertices generated in the same time slot form an independent set in the final graph
- Inheritance: The vertices in the birth path of a vertex u must activate an edge with every neighbor v of u



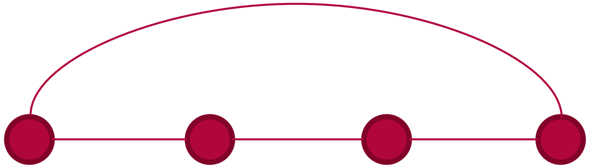
Fundamental Properties

- Independency: The vertices generated in the same time slot form an independent set in the final graph
- Inheritance: The vertices in the birth path of a vertex u must activate an edge with every neighbor v of u



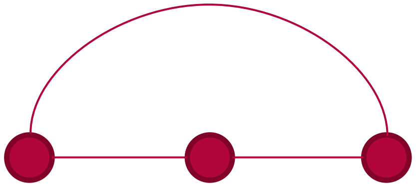
Fundamental Properties

- Independency: The vertices generated in the same time slot form an independent set in the final graph
- Inheritance: The vertices in the birth path of a vertex u must activate an edge with every neighbor v of u



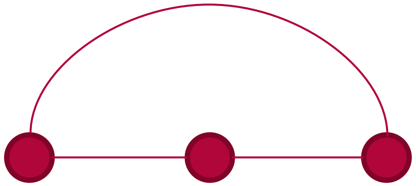
Fundamental Properties

- Independency: The vertices generated in the same time slot form an independent set in the final graph
- Inheritance: The vertices in the birth path of a vertex u must activate an edge with every neighbor v of u



Fundamental Properties

- Independency: The vertices generated in the same time slot form an independent set in the final graph
- Inheritance: The vertices in the birth path of a vertex u must activate an edge with every neighbor v of u

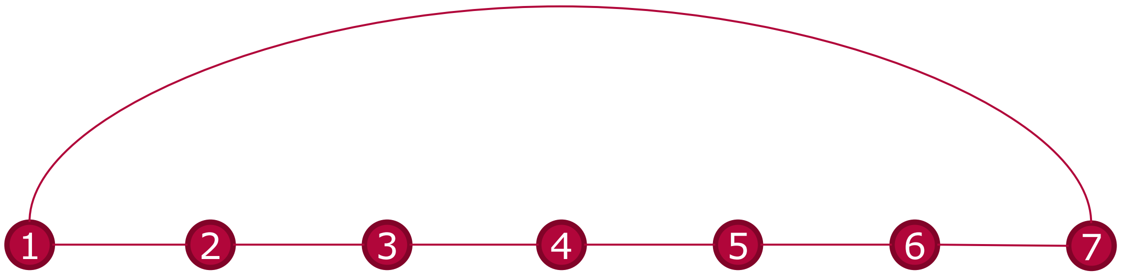


Fundamental Properties

- Independency: The vertices generated in the same time slot form an independent set in the final graph
- Inheritance: The vertices in the birth path of a vertex u must activate an edge with every neighbor v of u

Fundamental Properties

- Independency: The vertices generated in the same time slot form an independent set in the final graph
- Inheritance: The vertices in the birth path of a vertex u must activate an edge with every neighbor v of u



Growth Schedule

a) 1→2
b) 2→3 (3,1)
c) 3→4 (4,1)
d) 4→5 (5,1)
e) 5→6 (6,1)
f) 6→7 (7,1)

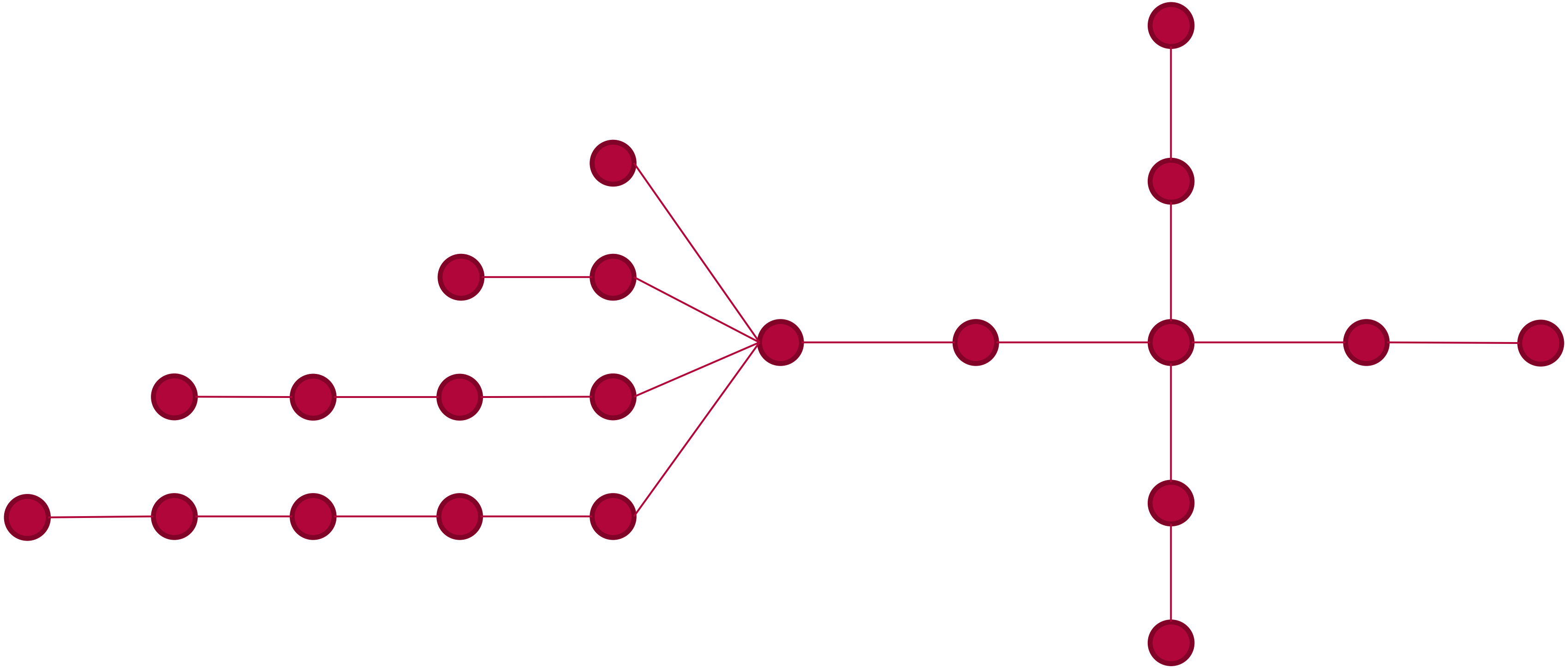
Growth Schedule for Tree Graphs

Theorem

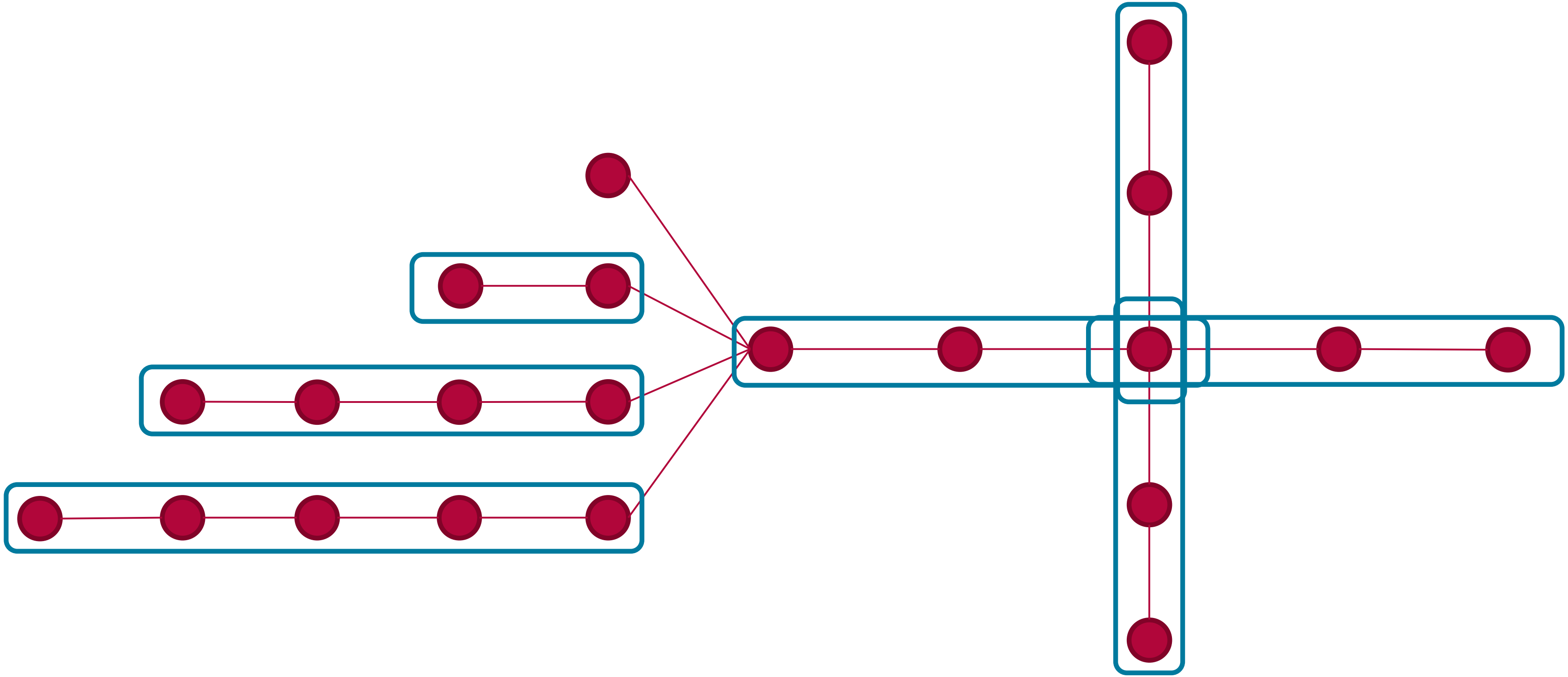
The Tree *algorithm computes, in polynomial time, a growth schedule for any given tree graph G with $O(\log^2 n)$ slots and with $O(n)$ excess edges.*

- Decomposition strategy where vertices are removed in phases until a single node is present

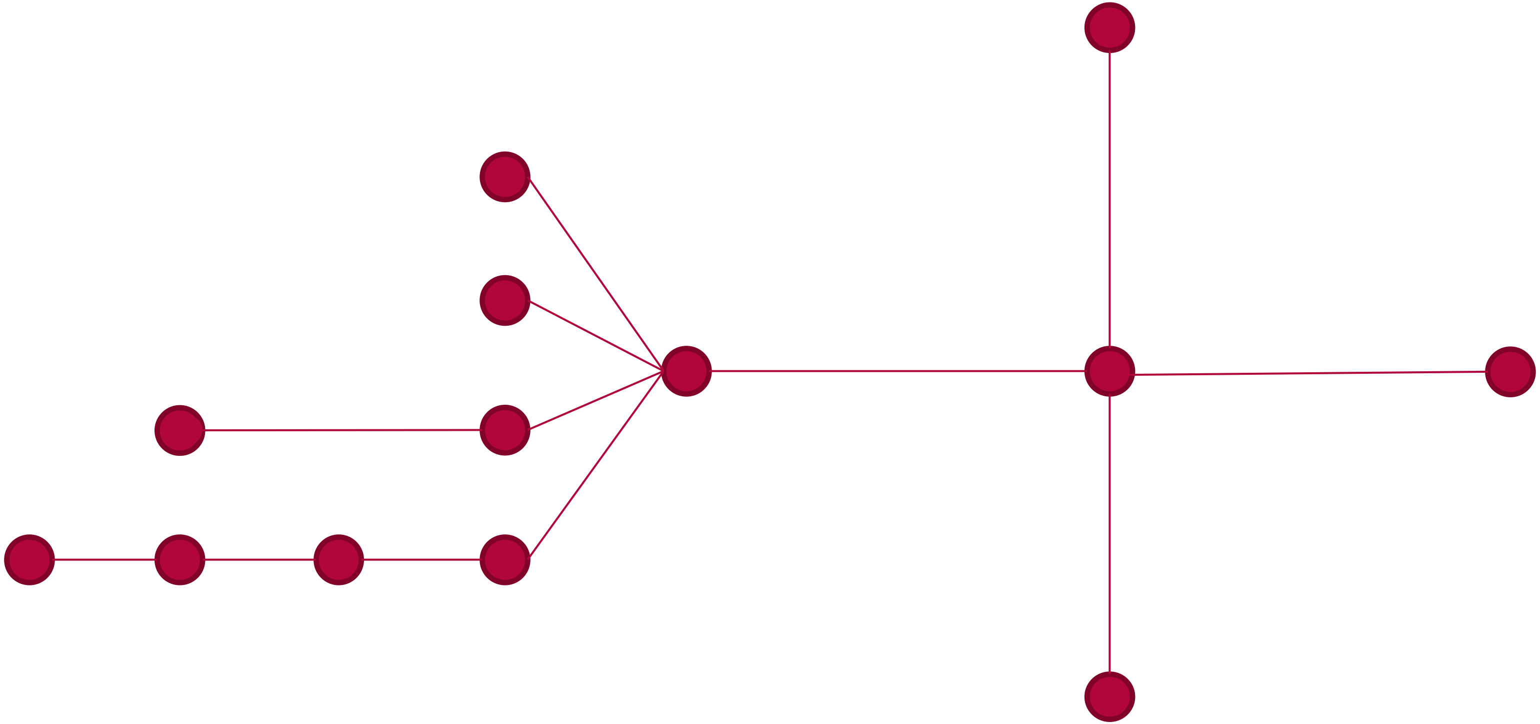
Growth Schedule for Tree Graphs



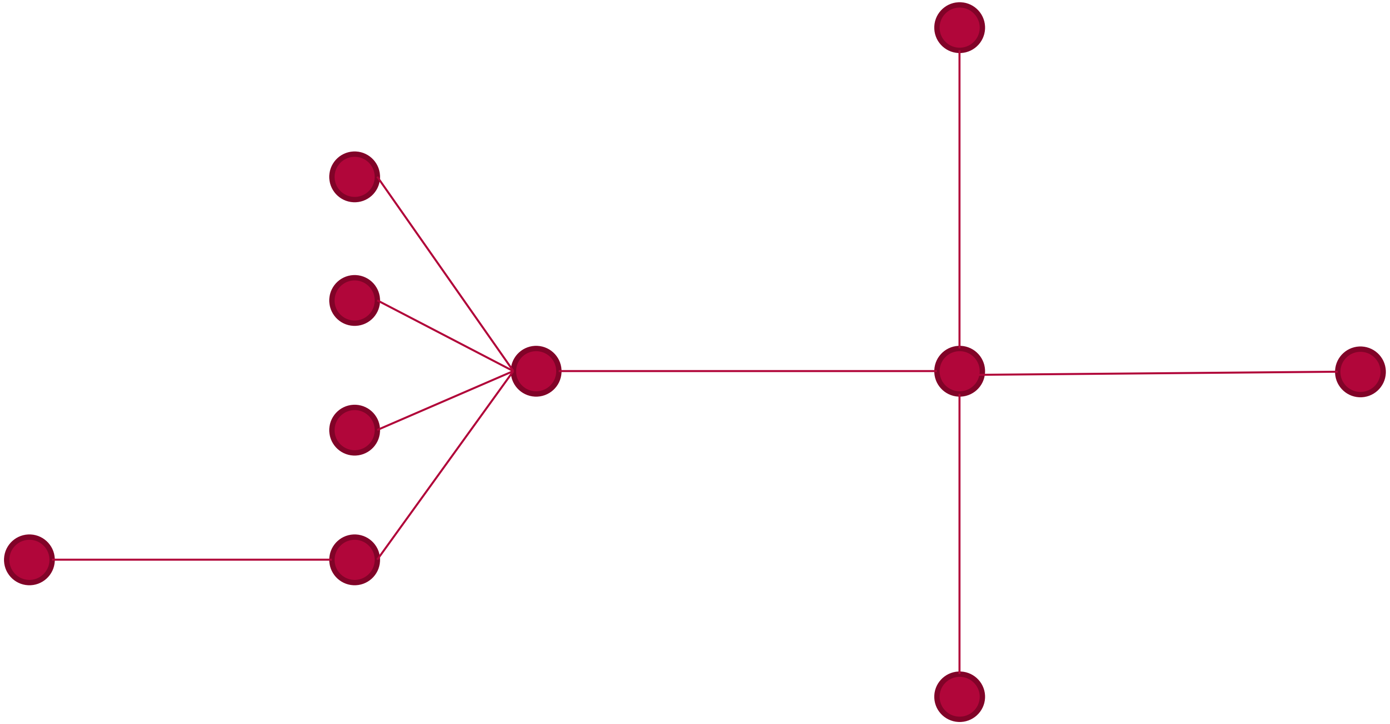
Growth Schedule for Tree Graphs



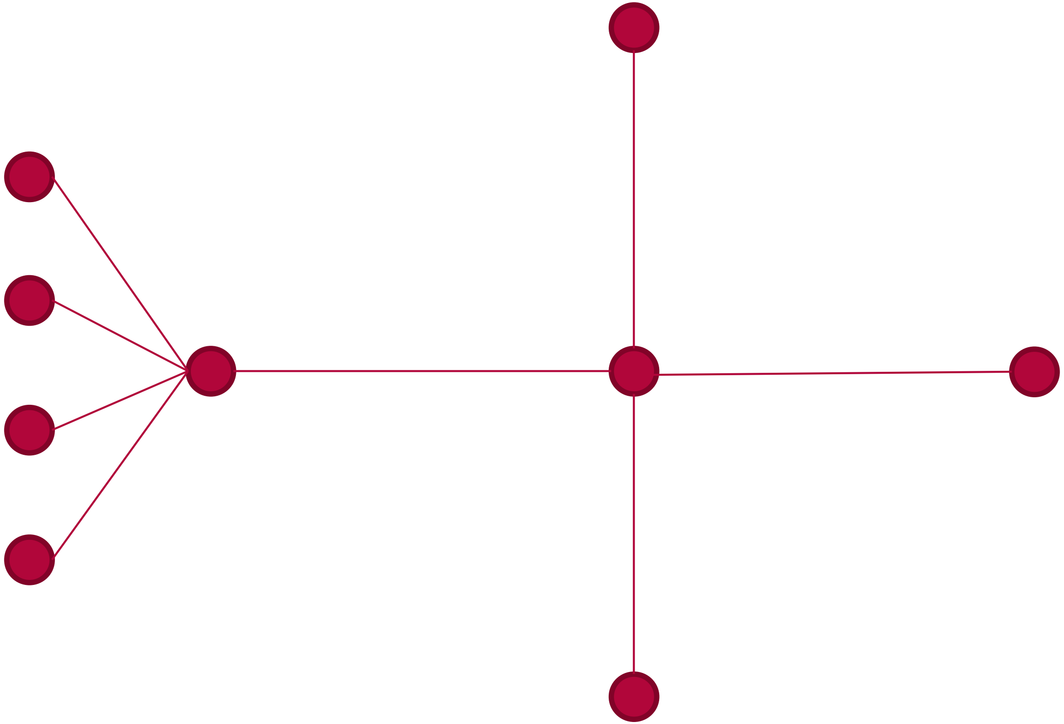
Growth Schedule for Tree Graphs



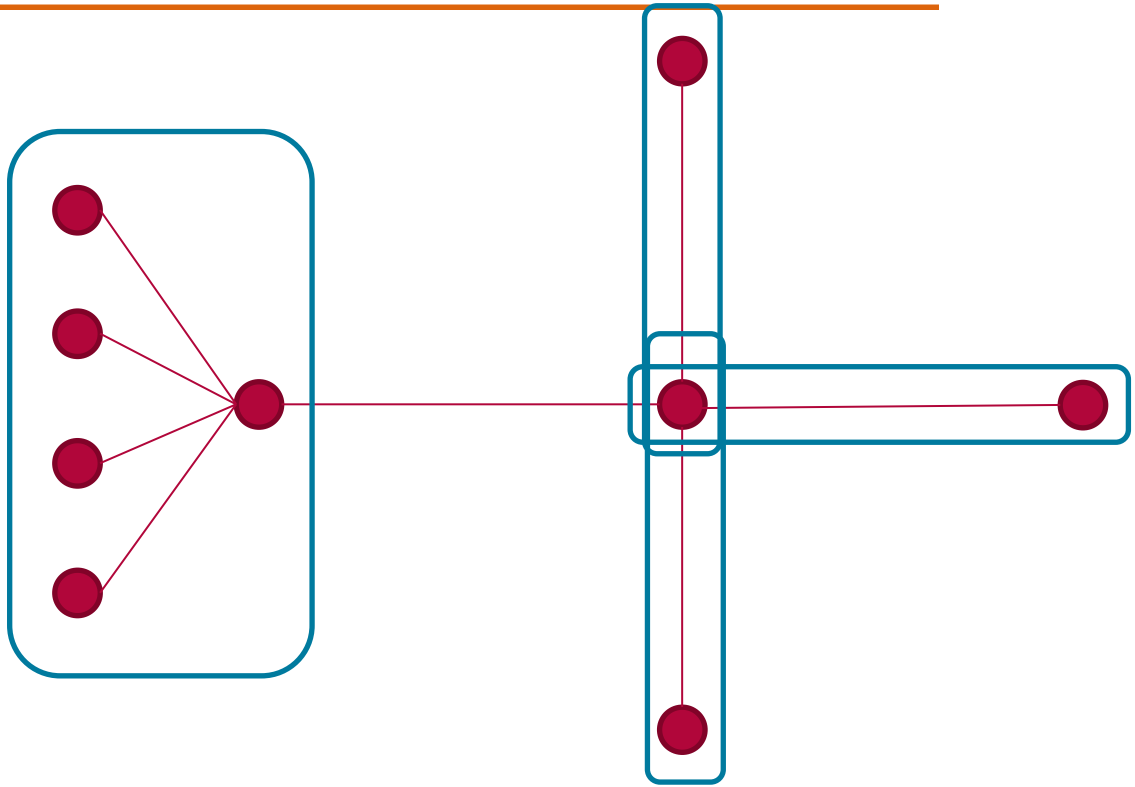
Growth Schedule for Tree Graphs



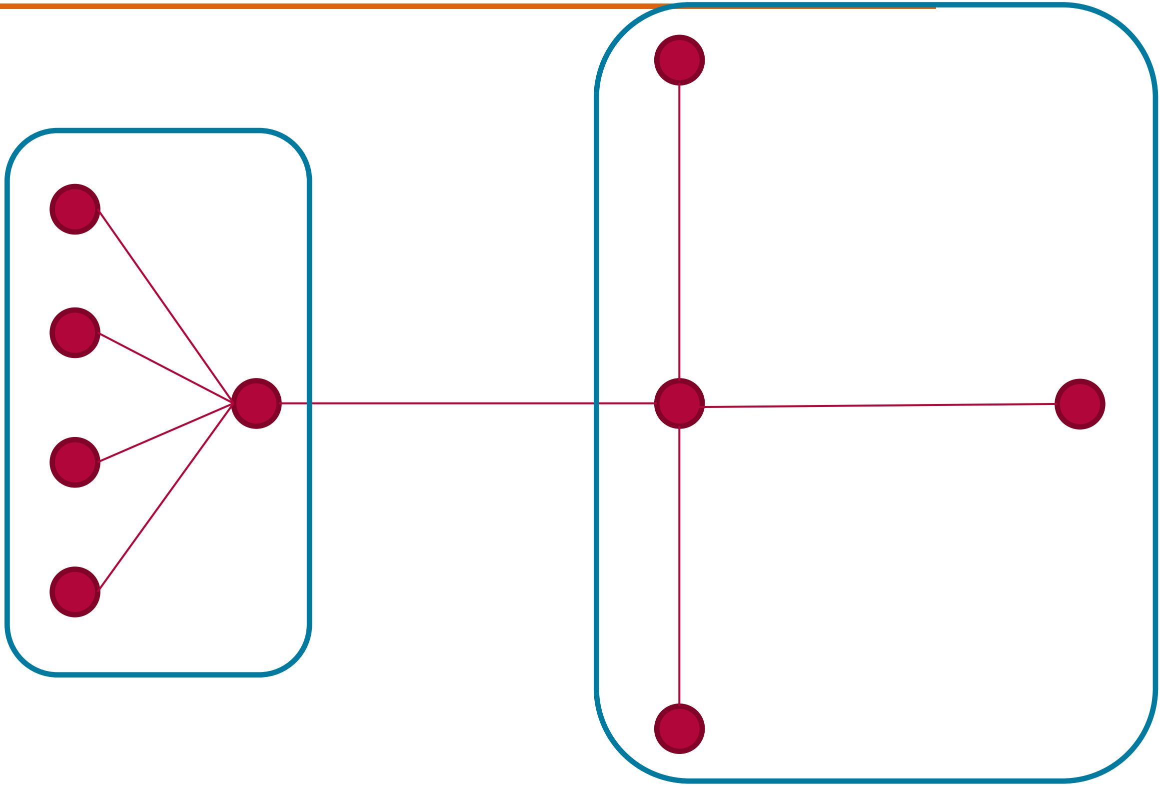
Growth Schedule for Tree Graphs



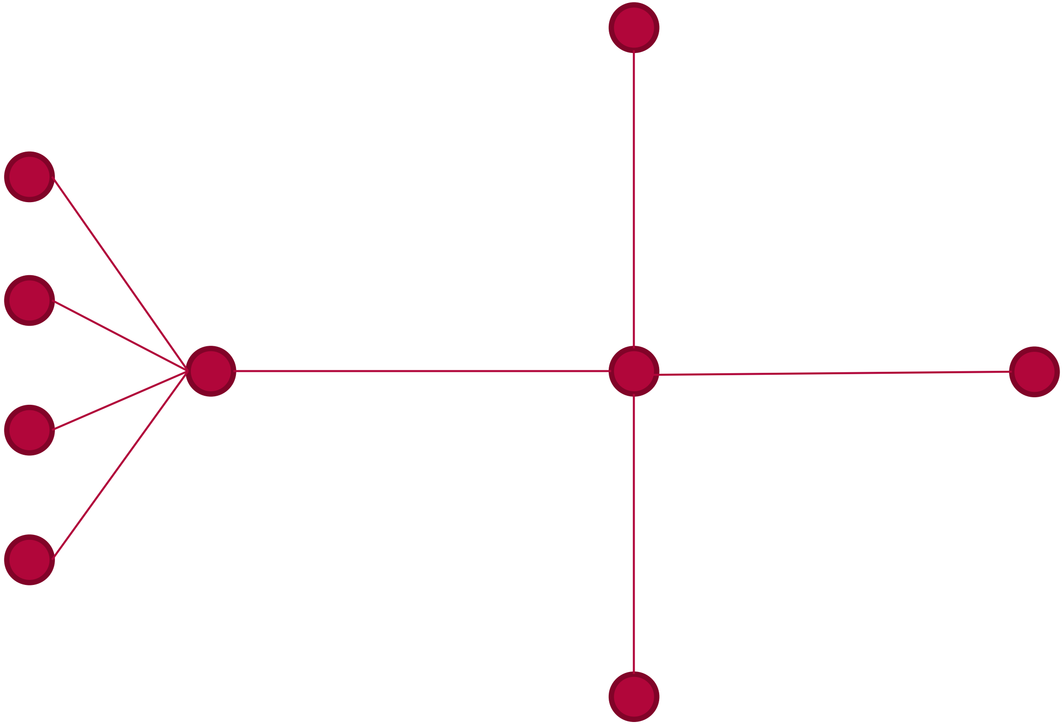
Growth Schedule for Tree Graphs



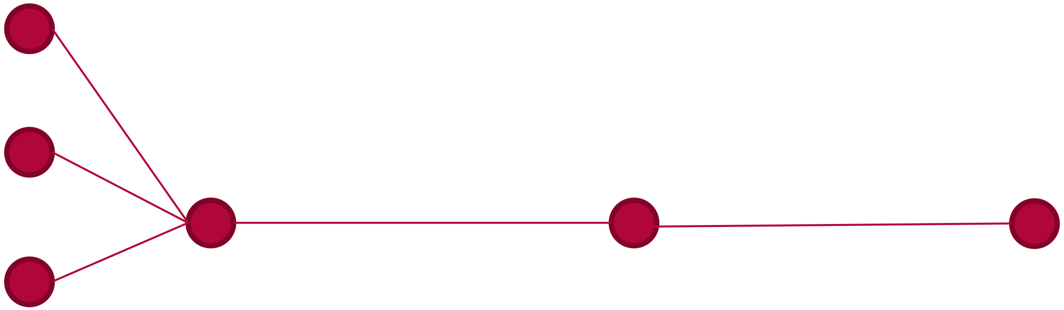
Growth Schedule for Tree Graphs



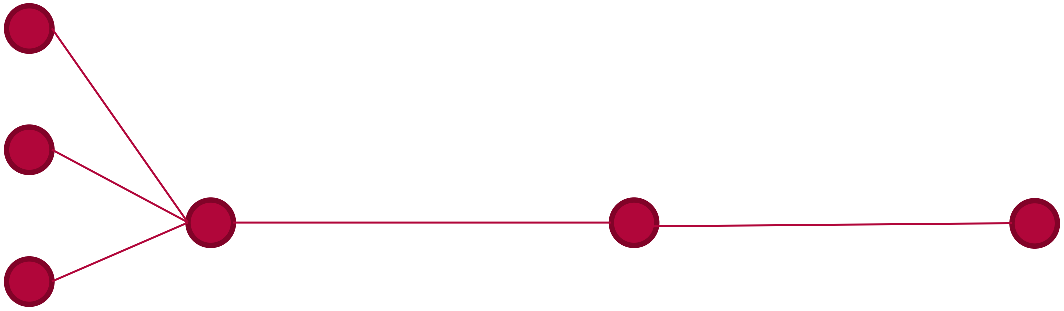
Growth Schedule for Tree Graphs



Growth Schedule for Tree Graphs



Growth Schedule for Tree Graphs



Growth Schedule for Tree Graphs

Growth Schedule for Tree Graphs

Growth Schedule for Tree Graphs

Growth Schedule for Tree Graphs

Growth Schedule for Tree Graphs

Growth Schedule for Tree Graphs

Theorem

The Tree algorithm computes, in polynomial time, a growth schedule for any given tree graph G with $O(\log^2 n)$ slots and with $O(n)$ excess edges.

- Decomposition strategy where vertices are removed in phases until a single node is present
- The phases can be reversed using $O(\log^2 n)$ slots and $O(n)$ excess edges

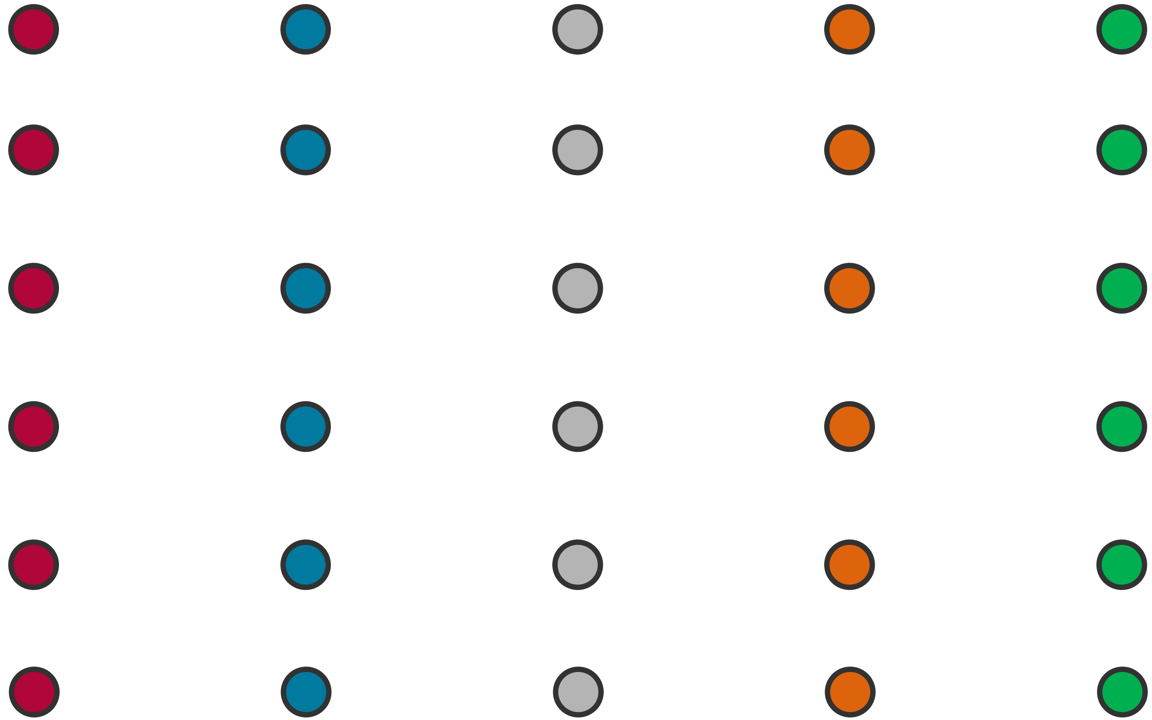
Growth Schedule for Planar Graphs

Theorem

The Planar algorithm computes, in polynomial time, a growth schedule for any given planar graph G with $O(\log n)$ slots and with $O(n \log n)$ excess edges.

- Compute a 5-coloring of the input planar graph
- Grow the vertices of each color class one by one using a star growth schedule for each color

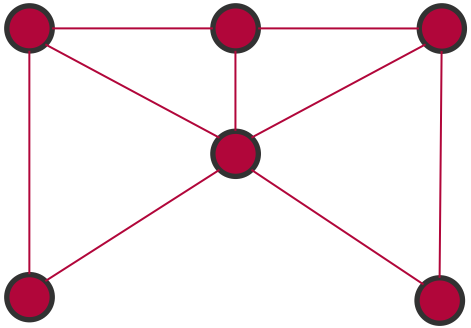
Growth Schedule for Planar Graphs



Zero Waste Growth Schedule Problem

Definition

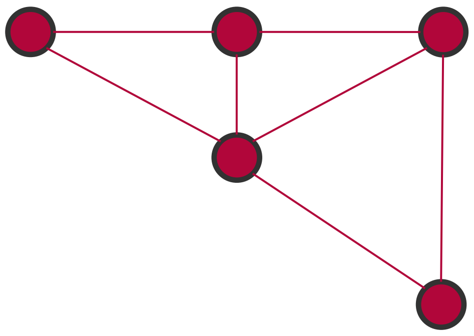
Candidate Vertex: A vertex $v \in V$ can be the last generated vertex in a growth schedule of $\ell = 0$ for G if there exists a vertex $w \in V \setminus v$ such that $N[v] \subseteq N[w]$.



Zero Waste Growth Schedule Problem

Definition

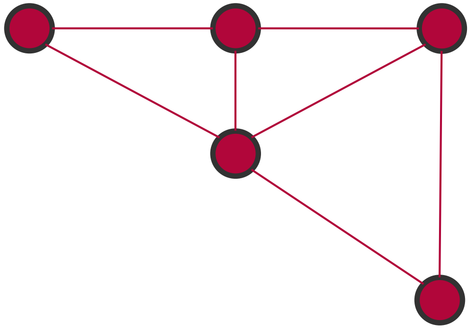
Candidate Vertex: A vertex $v \in V$ can be the last generated vertex in a growth schedule of $\ell = 0$ for G if there exists a vertex $w \in V \setminus v$ such that $N[v] \subseteq N[w]$.



Zero Waste Growth Schedule Problem

Definition

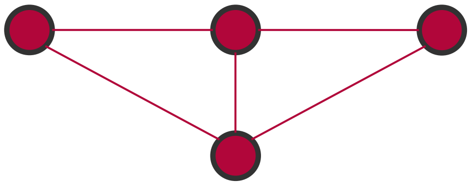
Candidate Vertex: A vertex $v \in V$ can be the last generated vertex in a growth schedule of $\ell = 0$ for G if there exists a vertex $w \in V \setminus v$ such that $N[v] \subseteq N[w]$.



Zero Waste Growth Schedule Problem

Definition

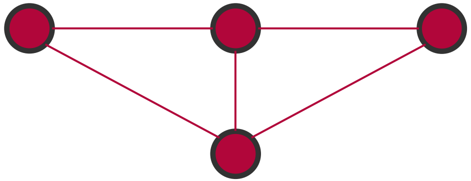
Candidate Vertex: A vertex $v \in V$ can be the last generated vertex in a growth schedule of $\ell = 0$ for G if there exists a vertex $w \in V \setminus v$ such that $N[v] \subseteq N[w]$.



Zero Waste Growth Schedule Problem

Definition

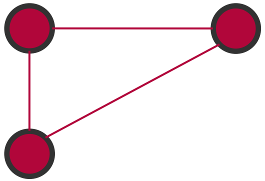
Candidate Vertex: A vertex $v \in V$ can be the last generated vertex in a growth schedule of $\ell = 0$ for G if there exists a vertex $w \in V \setminus v$ such that $N[v] \subseteq N[w]$.



Zero Waste Growth Schedule Problem

Definition

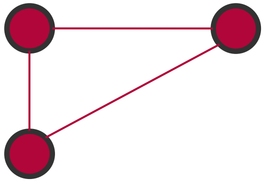
Candidate Vertex: A vertex $v \in V$ can be the last generated vertex in a growth schedule of $\ell = 0$ for G if there exists a vertex $w \in V \setminus v$ such that $N[v] \subseteq N[w]$.



Zero Waste Growth Schedule Problem

Definition

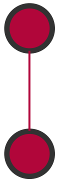
Candidate Vertex: A vertex $v \in V$ can be the last generated vertex in a growth schedule of $\ell = 0$ for G if there exists a vertex $w \in V \setminus v$ such that $N[v] \subseteq N[w]$.



Zero Waste Growth Schedule Problem

Definition

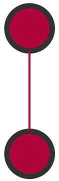
Candidate Vertex: A vertex $v \in V$ can be the last generated vertex in a growth schedule of $\ell = 0$ for G if there exists a vertex $w \in V \setminus v$ such that $N[v] \subseteq N[w]$.



Zero Waste Growth Schedule Problem

Definition

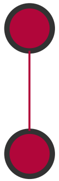
Candidate Vertex: A vertex $v \in V$ can be the last generated vertex in a growth schedule of $\ell = 0$ for G if there exists a vertex $w \in V \setminus v$ such that $N[v] \subseteq N[w]$.



Zero Waste Growth Schedule Problem

Definition

Candidate Vertex: A vertex $v \in V$ can be the last generated vertex in a growth schedule of $\ell = 0$ for G if there exists a vertex $w \in V \setminus v$ such that $N[v] \subseteq N[w]$.



Zero Waste Growth Schedule Problem

Definition

Candidate Vertex: A vertex $v \in V$ can be the last generated vertex in a growth schedule of $\ell = 0$ for G if there exists a vertex $w \in V \setminus v$ such that $N[v] \subseteq N[w]$.

Zero Waste Growth Schedule Problem

Definition

Candidate Vertex: A vertex $v \in V$ can be the last generated vertex in a growth schedule of $\ell = 0$ for G if there exists a vertex $w \in V \setminus v$ such that $N[v] \subseteq N[w]$.

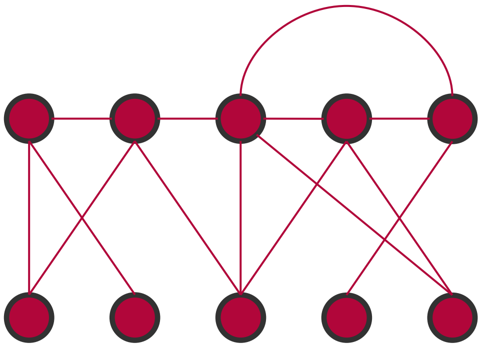
- A graph can be grown with ***l***=0 excess edges if and only if it has a candidate vertex ordering
- The Candidate vertex algorithm can decide in polynomial time, whether a given graph G has a growth schedule with n-1 slots and 0 excess edges.

Faster Growth Algorithm

Lemma

The fast growth algorithm computes in polynomial time a growth schedule σ for any graph $G = (V, E)$, where $|V| = 2^{\delta}$, with $\log n$ slots and $l = 0$ excess edges, if and only if such a σ exists for G.

- Find every candidate vertex and put them in set S
- Find a subset L of set S such that
  - L=n/2
  - L is an independent set
  - Perfect Matching

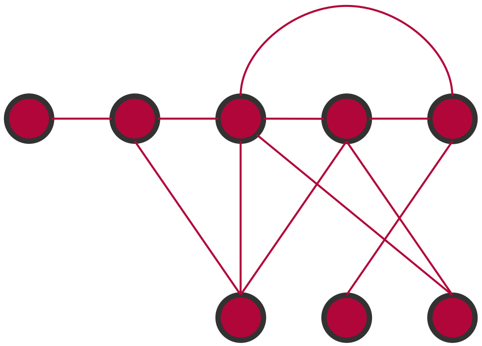


Faster Growth Algorithm

Lemma

The fast growth algorithm computes in polynomial time a growth schedule σ for any graph $G = (V, E)$, where $|V| = 2^\delta$, with $\log n$ slots and $l = 0$ excess edges, if and only if such a σ exists for G.

- Find every candidate vertex and put them in set S
- Find a subset L of set S such that
  - L=n/2
  - L is an independent set
  - Perfect Matching

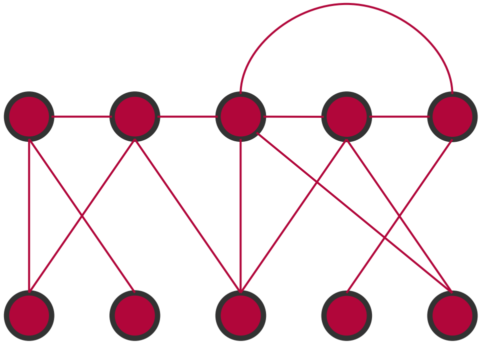


Faster Growth Algorithm

Lemma

The fast growth algorithm computes in polynomial time a growth schedule σ for any graph $G = (V, E)$, where $|V| = 2^{\delta}$, with $\log n$ slots and $l = 0$ excess edges, if and only if such a σ exists for G.

- Find every candidate vertex and put them in set S
- Find a subset L of set S such that
  - L=n/2
  - L is an independent set
  - Perfect Matching



Negative Results

Theorem

The decision version of the zero-waste growth schedule problem is NP-complete.

Theorem

Let $\epsilon > 0$. If there exists a polynomial-time algorithm, which, for every graph G, computes a $n^{\frac{1}{3}-\epsilon}$-approximate growth schedule, then P=NP.

- Both reductions are from the coloring problem

Open Problems

- Distributed Control
- Minimum number of edges for k=n-1 slots
- Parameterized Complexity
- Changes to the model

Thank you!