Minimal Percolating Sets with Time-Dependent Bootstrap Percolation

Yuyuan Luo
Mentor: Dr. Laura Schaposnik

PRIMES Conference
May 19, 2019

Introduction

Idea

Bootstrap percolation: A deterministic process on a network where nodes become infected once a certain number r of their neighbors are infected.

Introduction

Idea

Bootstrap percolation: A deterministic process on a network where nodes become infected once a certain number r of their neighbors are infected.

- Applications:
  - Fluid flow in porous areas
  - Orientational ordering processes of magnetic alloys
  - Failure of units in structured collection of computer memory
  - Modeling infectious diseases

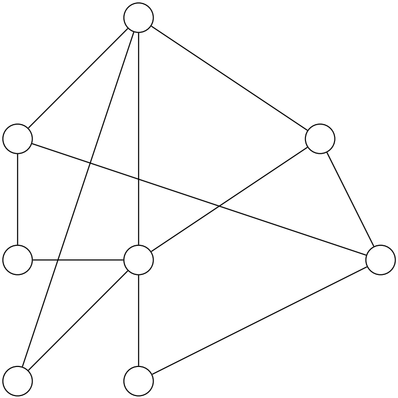
Introduction

Idea

Bootstrap percolation: A deterministic process on a network where nodes become infected once a certain number r of their neighbors are infected.

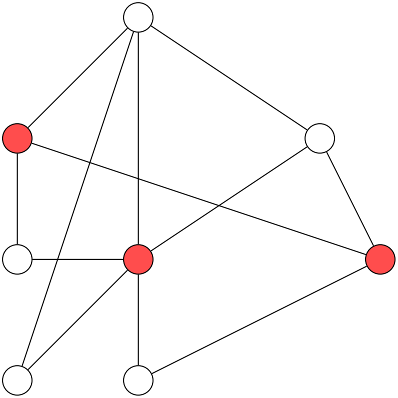
- Applications:
  - Fluid flow in porous areas
  - Orientational ordering processes of magnetic alloys
  - Failure of units in structured collection of computer memory
  - Modeling infectious diseases
- Advantages when used to model infectious diseases
  - Individual behavior
  - Spatial aspects
  - Mixing of individuals

Bootstrap percolation: an example



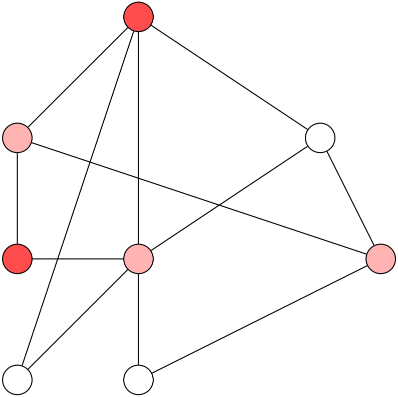
Bootstrap percolation: an example

$t = 0$
$r = 2$



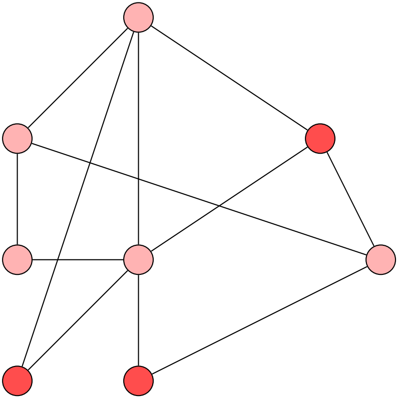
Bootstrap percolation: an example

$t = 1$
$r = 2$



Bootstrap percolation: an example

$t = 2$
$r = 2$



r-Bootstrap percolation: definition

Definition

Suppose we have a graph $G(V, E)$, some positive integer r, and a set $A_0 \subset V$ of initially infected nodes. Then in the $r-$bootstrap percolation process, the set of infected vertices at time $t+1$ is given by

$$A_{t+1} = A_t \cup \{v \in V(G) : |N(v) \cap A_t| \geq r\}$$

where $N(v)$ is the set of adjacent vertices to v.

Minimal Percolating Set

Notation

Let $P(A)$ represent the set of nodes infected by the end of the process for some set $A := A_0$.

Minimal Percolating Set

Notation

Let $P(A)$ represent the set of nodes infected by the end of the process for some set $A := A_0$.

Definition

A **percolating set** is a set A such that $P(A) = V$.

Minimal Percolating Set

Notation

Let $P(A)$ represent the set of nodes infected by the end of the process for some set $A := A_0$.

Definition

A **percolating set** is a set A such that $P(A) = V$.

Definition

A percolating set A is **minimal** if for any $v \in A$,

$$P(A \setminus \{v\}) \neq V.$$

Example: Minimal Percolating Set

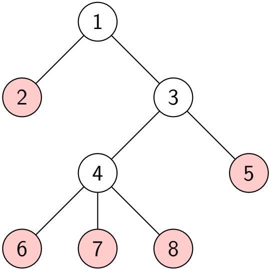


Figure: In this tree, having nodes $2, 5, 6, 7, 8$ initially infected is sufficient to ensure that the whole tree is infected at some finite time for $r = 2$. Further, since each of them has only 1 neighbor, removing any will cause the remaining set to not percolate.

A new model: time-dependent percolation

Motivation

Rate of disease spread is not constant in nature.

Idea

Instead of r, use $F(t)$.

A new model: time-dependent percolation

Motivation

Rate of disease spread is not constant in nature.

Idea

Instead of r, use $F(t)$.

Definition

Consider a graph $G(V, E)$, a function $F(t)$, and a set $A_0 \subset V$ of initially infected nodes. Then in the $F(t)-$bootstrap percolation process the set of infected vertices at time $t+1$ is

$$A_{t+1} = A_t \cup \{v \in V(G) : |N(v) \cap A_t| \geq \lceil F(t) \rceil\}.$$

Smallest Minimal Percolating Sets

Problem Statement

Obtain an algorithm that finds a smallest minimal percolating set for any given tree $T(V, E)$ and percolation function $F(t)$.

Smallest Minimal Percolating Sets

Problem Statement

Obtain an algorithm that finds a smallest minimal percolating set for any given tree $T(V,E)$ and percolation function $F(t)$.

Notation

We denote by $t_f(v)$ the largest time t we allow a node to become infected.

Smallest Minimal Percolating Sets

Problem Statement

Obtain an algorithm that finds a smallest minimal percolating set for any given tree $T(V, E)$ and percolation function $F(t)$.

Notation

We denote by $t_f(v)$ the largest time t we allow a node to become infected.

Definition

Suppose we have a set of initially infected nodes A_0. A node v is **isolated** if it is impossible to infect it by time $t_f(v)$ by adding any one other node to A_0 that is not itself.

Smallest Minimal Percolating Sets

Problem Statement

Obtain an algorithm that finds a smallest minimal percolating set for any given tree $T(V, E)$ and percolation function $F(t)$.

Notation

We denote by $t_f(v)$ the largest time t we allow a node to become infected.

Definition

Suppose we have a set of initially infected nodes A_0. A node v is **isolated** if it is impossible to infect it by time $t_f(v)$ by adding any one other node to A_0 that is not itself.

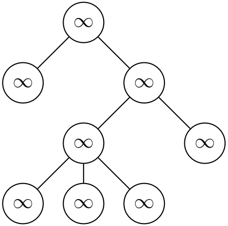
Notation

Let v be a node that is not isolated. We denote by $t_p(v)$ the largest time t at which $F(t)$ is not larger than the number of infected neighbors of v at t.

Algorithm for $F(t)$-BP on $T(V, E)$

- **Step 1. Initialize tree:** Let $A_0 = \varnothing$. For $v \in V$ set $t_f(v)$ arbitrarily large, and set it to true for needing to be considered.
- **Step 2. Percolate using current A_0.** Stop the algorithm if $P(A_0) = V$: the resulting A_0 is a smallest minimal percolating set.
- **Step 3. Infection.** For an unconsidered v furthest away from the root (isolated first).
  - if v is isolated, add v to A_0.
  - otherwise, set t_f of the parent of v to $t_p(v) - 1$ if it is smaller than the current t_f of the parent.
- **Step 4. Go to step 2.**

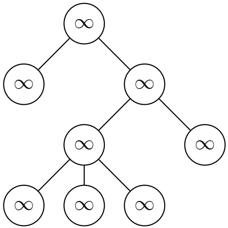
$F(t) = t$



Algorithm for $F(t)$-BP on $T(V, E)$

- **Step 1. Initialize tree:** Let $A_0 = \varnothing$. For $v \in V$ set $t_f(v)$ arbitrarily large, and set it to true for needing to be considered.
- **Step 2. Percolate using current A_0.** Stop the algorithm if $P(A_0) = V$: the resulting A_0 is a smallest minimal percolating set.
- **Step 3. Infection.** For an unconsidered v furthest away from the root (isolated first).
  - if v is isolated, add v to A_0.
  - otherwise, set t_f of the parent of v to $t_p(v) - 1$ if it is smaller than the current t_f of the parent.
- **Step 4. Go to step 2.**

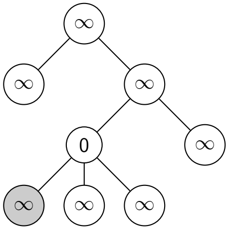
$F(t) = t$



Algorithm for $F(t)$-BP on $T(V, E)$

- **Step 1. Initialize tree:** Let $A_0 = \varnothing$. For $v \in V$ set $t_f(v)$ arbitrarily large, and set it to true for needing to be considered.
- **Step 2. Percolate using current A_0.** Stop the algorithm if $P(A_0) = V$: the resulting A_0 is a smallest minimal percolating set.
- **Step 3. Infection.** For an unconsidered v furthest away from the root (isolated first).
  - if v is isolated, add v to A_0.
  - otherwise, set t_f of the parent of v to $t_p(v) - 1$ if it is smaller than the current t_f of the parent.
- **Step 4. Go to step 2.**

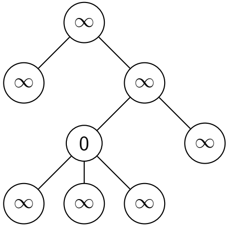
$F(t) = t$



Algorithm for $F(t)$-BP on $T(V, E)$

- **Step 1. Initialize tree:** Let $A_0 = \varnothing$. For $v \in V$ set $t_f(v)$ arbitrarily large, and set it to true for needing to be considered.
- **Step 2. Percolate using current A_0.** Stop the algorithm if $P(A_0) = V$: the resulting A_0 is a smallest minimal percolating set.
- **Step 3. Infection.** For an unconsidered v furthest away from the root (isolated first).
  - if v is isolated, add v to A_0.
  - otherwise, set t_f of the parent of v to $t_p(v) - 1$ if it is smaller than the current t_f of the parent.
- **Step 4. Go to step 2.**

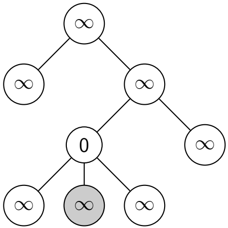
$F(t) = t$



Algorithm for $F(t)$-BP on $T(V, E)$

- **Step 1. Initialize tree:** Let $A_0 = \varnothing$. For $v \in V$ set $t_f(v)$ arbitrarily large, and set it to true for needing to be considered.
- **Step 2. Percolate using current A_0.** Stop the algorithm if $P(A_0) = V$: the resulting A_0 is a smallest minimal percolating set.
- **Step 3. Infection.** For an unconsidered v furthest away from the root (isolated first).
  - if v is isolated, add v to A_0.
  - otherwise, set t_f of the parent of v to $t_p(v) - 1$ if it is smaller than the current t_f of the parent.
- **Step 4. Go to step 2.**

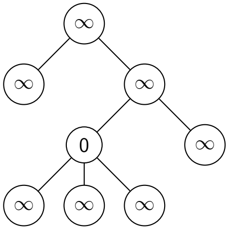
$F(t) = t$



Algorithm for $F(t)$-BP on $T(V, E)$

- **Step 1. Initialize tree:** Let $A_0 = \varnothing$. For $v \in V$ set $t_f(v)$ arbitrarily large, and set it to true for needing to be considered.
- **Step 2. Percolate using current A_0.** Stop the algorithm if $P(A_0) = V$: the resulting A_0 is a smallest minimal percolating set.
- **Step 3. Infection.** For an unconsidered v furthest away from the root (isolated first).
  - if v is isolated, add v to A_0.
  - otherwise, set t_f of the parent of v to $t_p(v) - 1$ if it is smaller than the current t_f of the parent.
- **Step 4. Go to step 2.**

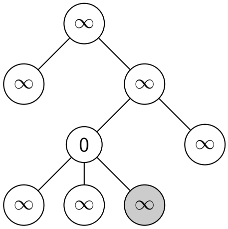
$F(t) = t$



Algorithm for $F(t)$-BP on $T(V, E)$

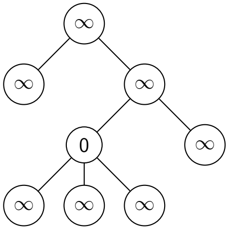
- **Step 1. Initialize tree:** Let $A_0 = \varnothing$. For $v \in V$ set $t_f(v)$ arbitrarily large, and set it to true for needing to be considered.
- **Step 2. Percolate using current A_0.** Stop the algorithm if $P(A_0) = V$: the resulting A_0 is a smallest minimal percolating set.
- **Step 3. Infection.** For an unconsidered v furthest away from the root (isolated first).
  - if v is isolated, add v to A_0.
  - otherwise, set t_f of the parent of v to $t_p(v) - 1$ if it is smaller than the current t_f of the parent.
- **Step 4. Go to step 2.**

$F(t) = t$



- **Step 1. Initialize tree:** Let $A_0 = \varnothing$. For $v \in V$ set $t_f(v)$ arbitrarily large, and set it to true for needing to be considered.
- **Step 2. Percolate using current A_0.** Stop the algorithm if $P(A_0) = V$: the resulting A_0 is a smallest minimal percolating set.
- **Step 3. Infection.** For an unconsidered v furthest away from the root (isolated first).
  - if v is isolated, add v to A_0.
  - otherwise, set t_f of the parent of v to $t_p(v) - 1$ if it is smaller than the current t_f of the parent.
- **Step 4. Go to step 2.**

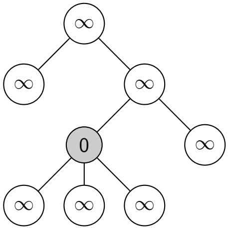
$F(t) = t$



Algorithm for $F(t)$-BP on $T(V, E)$

- **Step 1. Initialize tree:** Let $A_0 = \varnothing$. For $v \in V$ set $t_f(v)$ arbitrarily large, and set it to true for needing to be considered.
- **Step 2. Percolate using current A_0.** Stop the algorithm if $P(A_0) = V$: the resulting A_0 is a smallest minimal percolating set.
- **Step 3. Infection.** For an unconsidered v furthest away from the root (isolated first).
  - if v is isolated, add v to A_0.
  - otherwise, set t_f of the parent of v to $t_p(v) - 1$ if it is smaller than the current t_f of the parent.
- **Step 4. Go to step 2.**

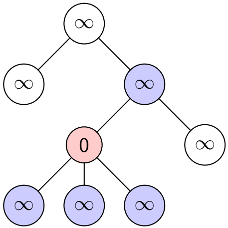
$F(t) = t$



Algorithm for $F(t)$-BP on $T(V, E)$

- **Step 1. Initialize tree:** Let $A_0 = \varnothing$. For $v \in V$ set $t_f(v)$ arbitrarily large, and set it to true for needing to be considered.
- **Step 2. Percolate using current A_0.** Stop the algorithm if $P(A_0) = V$: the resulting A_0 is a smallest minimal percolating set.
- **Step 3. Infection.** For an unconsidered v furthest away from the root (isolated first).
  - if v is isolated, add v to A_0.
  - otherwise, set t_f of the parent of v to $t_p(v) - 1$ if it is smaller than the current t_f of the parent.
- **Step 4. Go to step 2.**

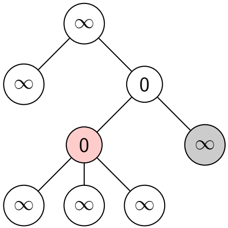
$F(t) = t$



Algorithm for $F(t)$-BP on $T(V, E)$

- **Step 1. Initialize tree:** Let $A_0 = \varnothing$. For $v \in V$ set $t_f(v)$ arbitrarily large, and set it to true for needing to be considered.
- **Step 2. Percolate using current A_0.** Stop the algorithm if $P(A_0) = V$: the resulting A_0 is a smallest minimal percolating set.
- **Step 3. Infection.** For an unconsidered v furthest away from the root (isolated first).
  - if v is isolated, add v to A_0.
  - otherwise, set t_f of the parent of v to $t_p(v) - 1$ if it is smaller than the current t_f of the parent.
- **Step 4. Go to step 2.**

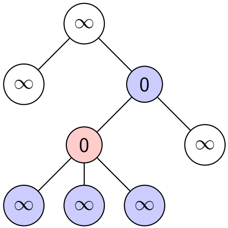
$F(t) = t$



Algorithm for $F(t)$-BP on $T(V,E)$

- **Step 1. Initialize tree:** Let $A_0 = \varnothing$. For $v \in V$ set $t_f(v)$ arbitrarily large, and set it to true for needing to be considered.
- **Step 2. Percolate using current A_0.** Stop the algorithm if $P(A_0) = V$: the resulting A_0 is a smallest minimal percolating set.
- **Step 3. Infection.** For an unconsidered v furthest away from the root (isolated first).
  - if v is isolated, add v to A_0.
  - otherwise, set t_f of the parent of v to $t_p(v) - 1$ if it is smaller than the current t_f of the parent.
- **Step 4. Go to step 2.**

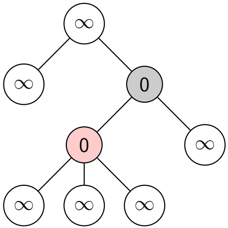
$F(t) = t$



Algorithm for $F(t)$-BP on $T(V, E)$

- **Step 1. Initialize tree:** Let $A_0 = \varnothing$. For $v \in V$ set $t_f(v)$ arbitrarily large, and set it to true for needing to be considered.
- **Step 2. Percolate using current A_0.** Stop the algorithm if $P(A_0) = V$: the resulting A_0 is a smallest minimal percolating set.
- **Step 3. Infection.** For an unconsidered v furthest away from the root (isolated first).
  - if v is isolated, add v to A_0.
  - otherwise, set t_f of the parent of v to $t_p(v) - 1$ if it is smaller than the current t_f of the parent.
- **Step 4. Go to step 2.**

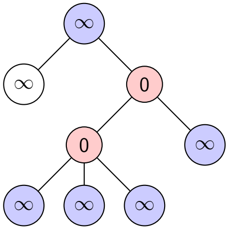
$F(t) = t$



Algorithm for $F(t)$-BP on $T(V,E)$

- **Step 1. Initialize tree:** Let $A_0 = \varnothing$. For $v \in V$ set $t_f(v)$ arbitrarily large, and set it to true for needing to be considered.
- **Step 2. Percolate using current A_0.** Stop the algorithm if $P(A_0) = V$: the resulting A_0 is a smallest minimal percolating set.
- **Step 3. Infection.** For an unconsidered v furthest away from the root (isolated first).
  - if v is isolated, add v to A_0.
  - otherwise, set t_f of the parent of v to $t_p(v) - 1$ if it is smaller than the current t_f of the parent.
- **Step 4. Go to step 2.**

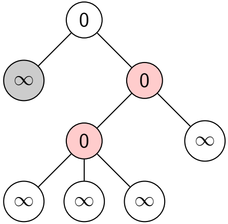
$F(t) = t$



Algorithm for $F(t)$-BP on $T(V, E)$

- **Step 1. Initialize tree:** Let $A_0 = \varnothing$. For $v \in V$ set $t_f(v)$ arbitrarily large, and set it to true for needing to be considered.
- **Step 2. Percolate using current A_0.** Stop the algorithm if $P(A_0) = V$: the resulting A_0 is a smallest minimal percolating set.
- **Step 3. Infection.** For an unconsidered v furthest away from the root (isolated first).
  - if v is isolated, add v to A_0.
  - otherwise, set t_f of the parent of v to $t_p(v) - 1$ if it is smaller than the current t_f of the parent.
- **Step 4. Go to step 2.**

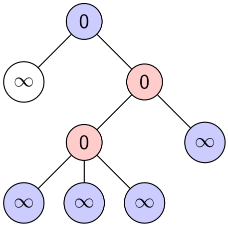
$F(t) = t$



Algorithm for $F(t)$-BP on $T(V, E)$

- **Step 1. Initialize tree:** Let $A_0 = \varnothing$. For $v \in V$ set $t_f(v)$ arbitrarily large, and set it to true for needing to be considered.
- **Step 2. Percolate using current A_0.** Stop the algorithm if $P(A_0) = V$: the resulting A_0 is a smallest minimal percolating set.
- **Step 3. Infection.** For an unconsidered v furthest away from the root (isolated first).
  - if v is isolated, add v to A_0.
  - otherwise, set t_f of the parent of v to $t_p(v) - 1$ if it is smaller than the current t_f of the parent.
- **Step 4. Go to step 2.**

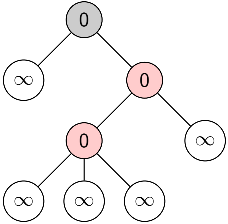
$F(t) = t$



Algorithm for $F(t)$-BP on $T(V, E)$

- **Step 1. Initialize tree:** Let $A_0 = \varnothing$. For $v \in V$ set $t_f(v)$ arbitrarily large, and set it to true for needing to be considered..
- **Step 2. Percolate using current A_0.** Stop the algorithm if $P(A_0) = V$: the resulting A_0 is a smallest minimal percolating set.
- **Step 3. Infection.** For an unconsidered v furthest away from the root (isolated first).
  - if v is isolated, add v to A_0.
  - otherwise, set t_f of the parent of v to $t_p(v) - 1$ if it is smaller than the current t_f of the parent.
- **Step 4. Go to step 2.**

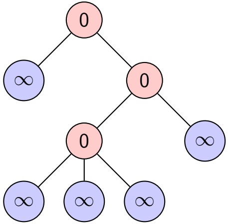
$F(t) = t$



Algorithm for $F(t)$-BP on $T(V, E)$

- **Step 1. Initialize tree:** Let $A_0 = \varnothing$. For $v \in V$ set $t_f(v)$ arbitrarily large, and set it to true for needing to be considered.
- **Step 2. Percolate using current A_0.** Stop the algorithm if $P(A_0) = V$: the resulting A_0 is a smallest minimal percolating set.
- **Step 3. Infection.** For an unconsidered v furthest away from the root (isolated first).
  - if v is isolated, add v to A_0.
  - otherwise, set t_f of the parent of v to $t_p(v) - 1$ if it is smaller than the current t_f of the parent.
- **Step 4. Go to step 2.**

$F(t) = t$



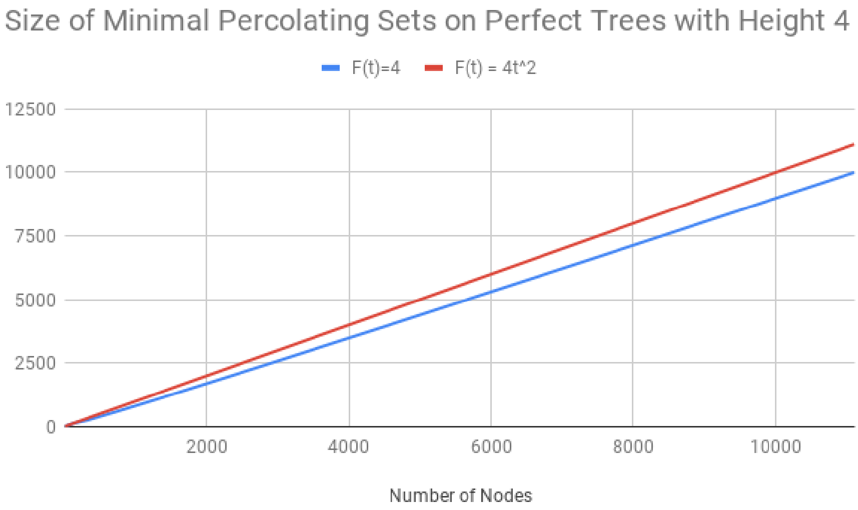


Figure: The size of smallest minimal percolating sets on perfect trees with height 4, with a constant and a non-constant percolation function

Future Directions

- Devise a similar algorithm for largest minimal percolating set.
- Study the size of largest and smallest minimal percolating sets in specific trees and lattices, with specific types of percolation functions.

Acknowledgements

- Dr. Laura Schaposnik
- Dr. Tanya Khovanova
- My parents
- MIT PRIMES program