To Satisfy Impatient Web Surfers is Hard

Fedor V. Fomin[1] Frédéric Giroire[2] Alain Jean-Marie[3]
Dorian Mazauric[2] Nicolas Nisse[2]

[1] University of Bergen, Norway

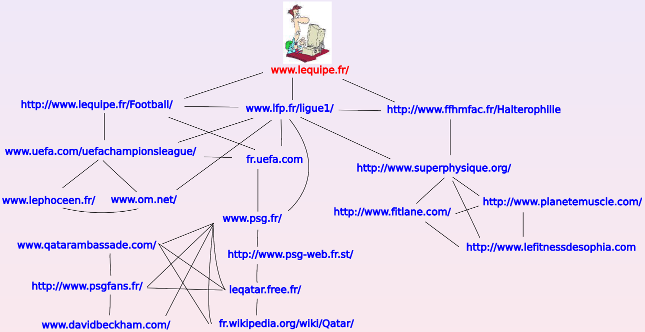
[2] MASCOTTE, INRIA, I3S (CNRS, UNS) Sophia Antipolis, France

[3] LIRMM, MAESTRO, INRIA, Montpellier, France

Séminaire MASCOTTE, October 25^{th} , 2011

Prefetching for faster data access

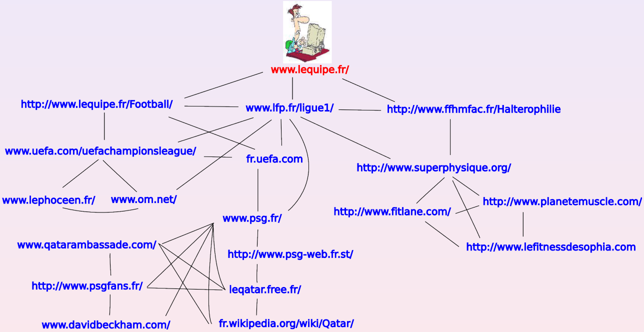
Web: pre-loading web pages before the web Surfer accesses it
Goal: Avoid the web Surfer to wait



CACHE www.lequipe.fr/

Prefetching for faster data access

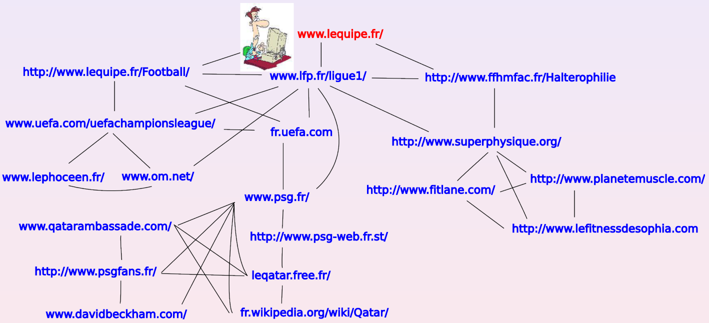
the web Surfer starts from a given web page in cache
try to load web pages in cache →TAKES TIME



CACHE www.lequipe.fr/ www.lfp.fr/ligue1/ http://www.lequ...

Prefetching for faster data access

at some point, web Surfer moves
if web page reached already in the cache OK

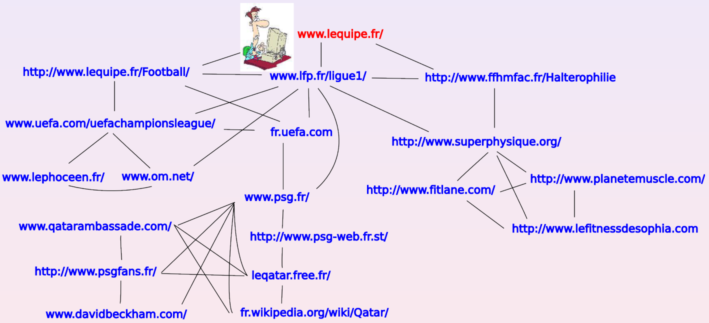


CACHE

www.lequipe.fr/ www.lfp.fr/ligue1/ http://www.lequipe.fr/Football/ http://www.ffhmfac.fr/Halterophilie
www.uefa.com/uefacham...

Prefetching for faster data access

web Surfer follows the hyperlinks in an unpredictable way
web pages to be loaded may be "guessed"

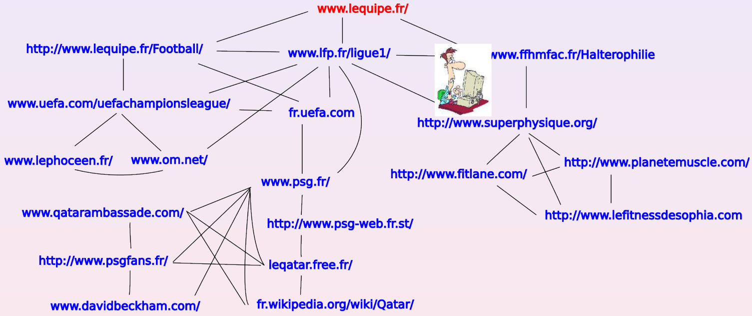


CACHE

www.lequipe.fr/ www.lfp.fr/ligue1/ http://www.lequipe.fr/Football/ http://www.ffhmfac.fr/Halterophilie
www.uefa.com/uefachampionsleague/ fr.uefa.com www.om.net/ www.psg.fr/
http://www.superp...

Prefetching for faster data access

if time is not sufficient...
web Surfer may access a page not in cache



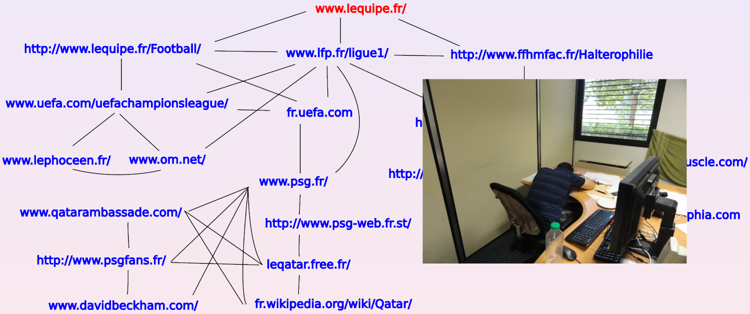
CACHE

www.lequipe.fr/ www.lfp.fr/ligue1/ http://www.lequipe.fr/Football/ http://www.ffhmfac.fr/Halterophilie
www.uefa.com/uefachampionsleague/ fr.uefa.com www.om.net/ www.psg.fr/
http://www.superp...

Prefetching for faster data access

in this case, web Surfer must wait

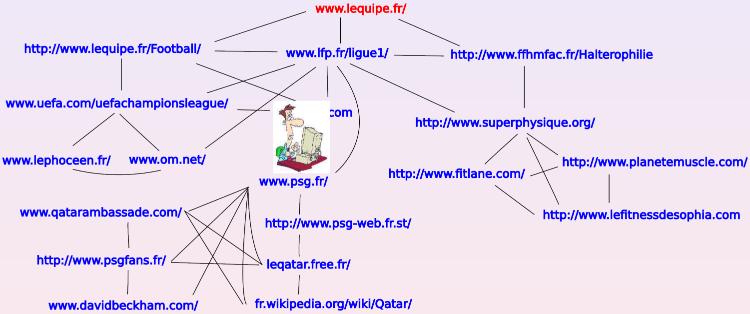
not good for research...



CACHE

www.lequipe.fr/ www.lfp.fr/ligue1/ http://www.lequipe.fr/Football/ http://www.ffhmfac.fr/Halterophilie
www.uefa.com/uefachampionsleague/ fr.uefa.com www.om.net/ www.psg.fr/
http://www.superp...

even if we guessed well
choices might be too numerous

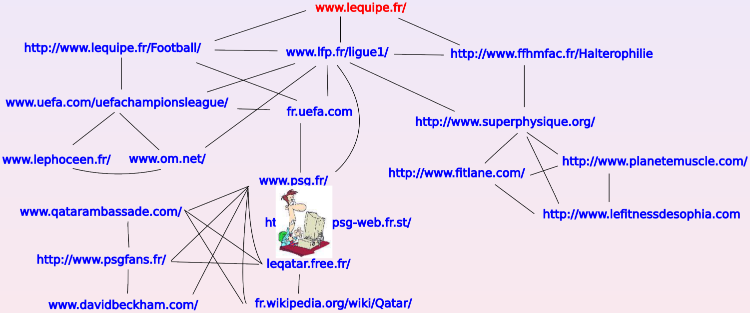


CACHE

www.lequipe.fr/ www.lfp.fr/ligue1/ http://www.lequipe.fr/Football/ http://www.ffhmfac.fr/Halterophilie
www.uefa.com/uefachampionsleague/ fr.uefa.com www.om.net/ www.psg.fr/
http://www.superp...

Prefetching for faster data access

again...
web Surfer may access a page not in cache



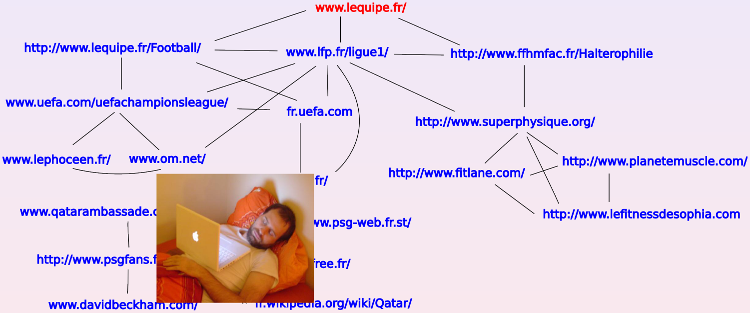
CACHE

www.lequipe.fr/ www.lfp.fr/ligue1/ http://www.lequipe.fr/Football/ http://www.ffhmfac.fr/Halterophilie
www.uefa.com/uefachampionsleague/ fr.uefa.com www.om.net/ www.psg.fr/
http://www.superp...
www.qatarambassade.com/ http://www.psgfans.fr/ http://www.psg-web.fr.st/ www.davidbec...

Prefetching for faster data access

again...

not good for research...



CACHE

www.lequipe.fr/ www.lfp.fr/ligue1/ http://www.lequipe.fr/Football/ http://www.ffhmfac.fr/Halterophilie
www.uefa.com/uefachampionsleague/ fr.uefa.com www.om.net/ www.psg.fr/
http://www.superp...
www.qatarambassade.com/ http://www.psgfans.fr/ http://www.psg-web.fr.st/ www.davidbec...

Prefetching for faster data access

Issue: download speed, NOT size of cache
Instance: network KNOWN ((di)graph)

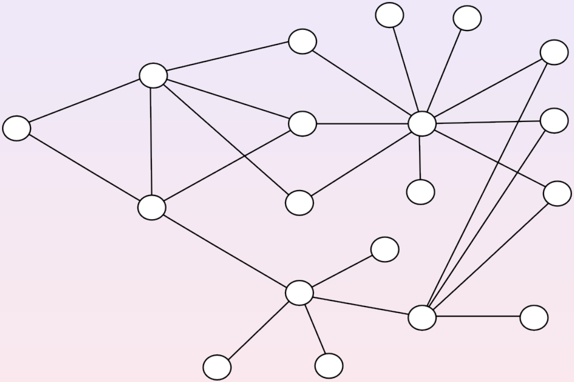
Related work: Probabilistic algorithms
(arcs + transition probabilities)

- Markovian model [Vitter,Krishnan. JACM'96] [Morad,Jean-Marie. ROADEF'10]
- Stochastic Dynamic Programming framework [Joseph,Grunwald. ISCA'97] [Grigoras,Charvillat,Douze. ACM Multimedia'02]

Our work:

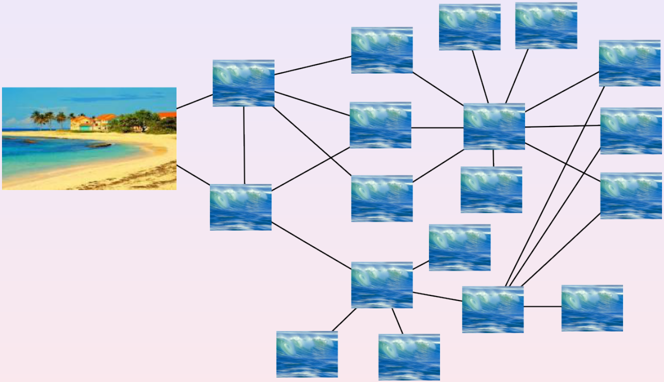
minimize download speed
to insure web Surfer never waits
(worst case, deterministic)

Model



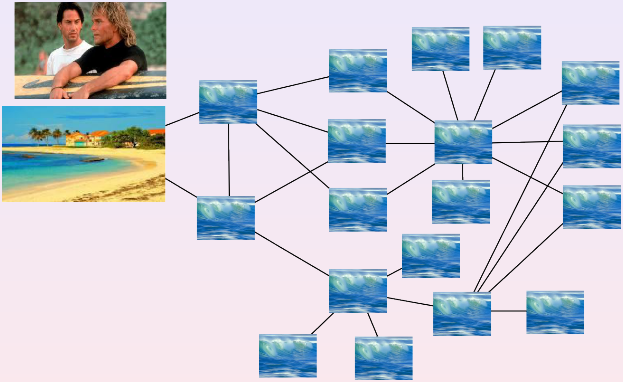
Web = connected (di)graph

Model



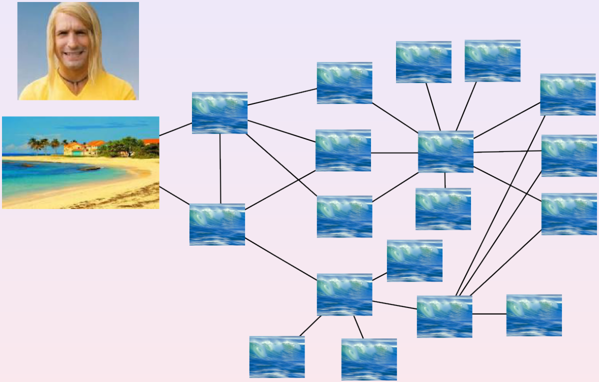
Web = (di)graph with a specified starting point (beach)

Model



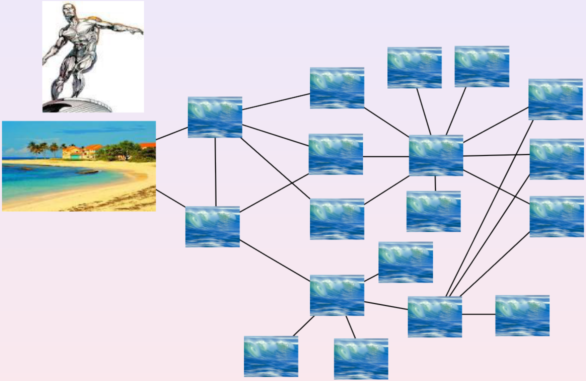
then, we need a (web) Surfer ..not them...

Model



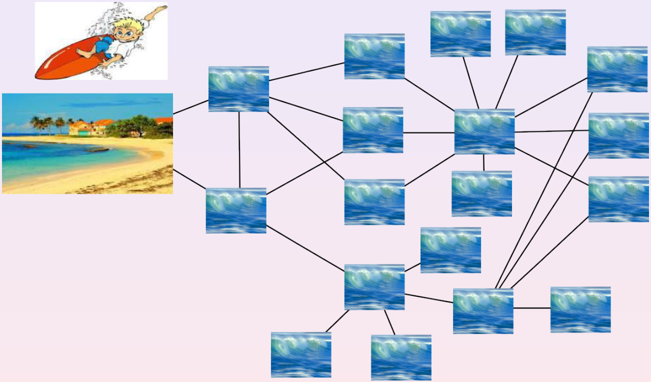
then, we need a (web) Surfer ..definitively not...

Model



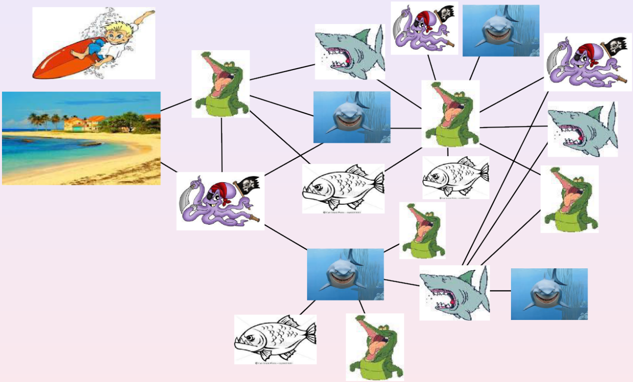
then, we need a (web) Surfer ..almost...

Model



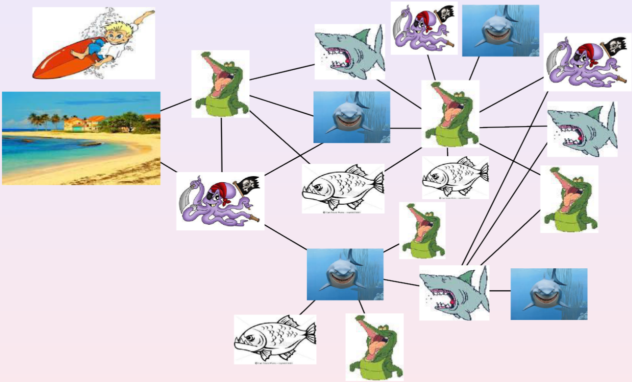
then, we need a (web) Surfer ...let's take this one

Model



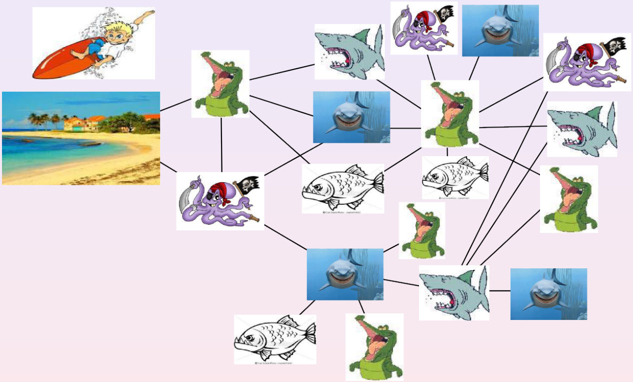
unloaded page = dangerous node

Model



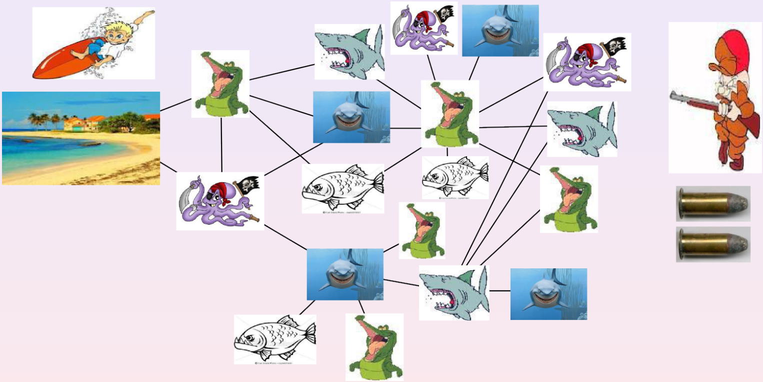
we need someone to help the Surfer oups... sorry

Model



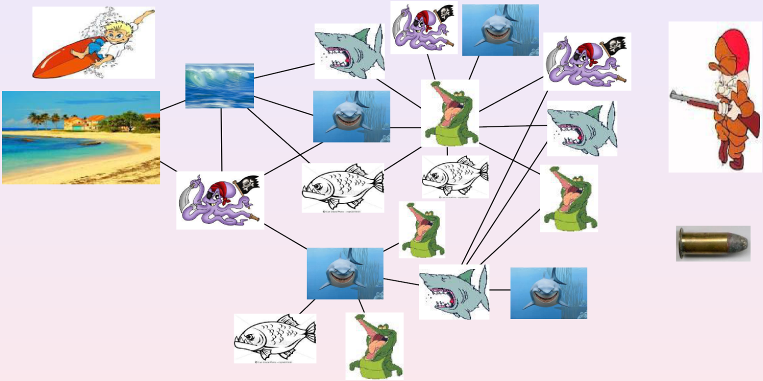
we need someone to help the Surfer let's call it *the Guard*

Model



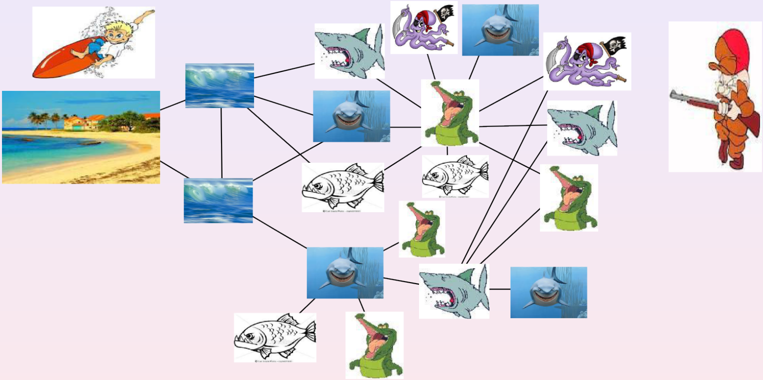
download speed = amount of balls

Model



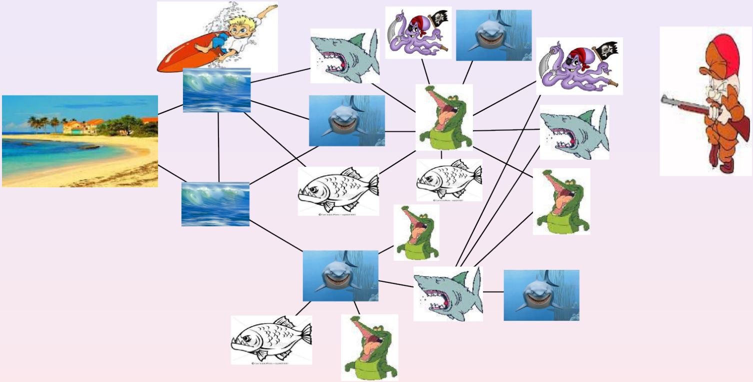
Guard uses one ball to secure one node

Model



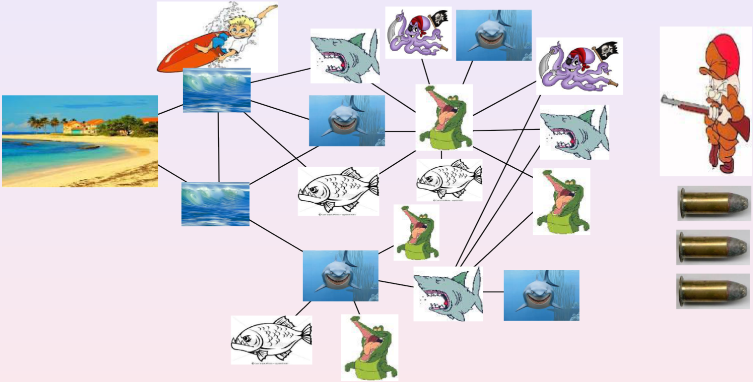
clearly, $degree(beach) \leq \#$ of balls required to save Surfer

Model



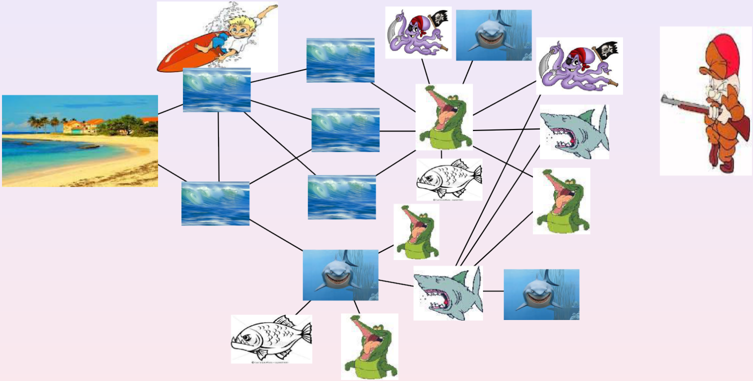
then, Surfer may move

Model



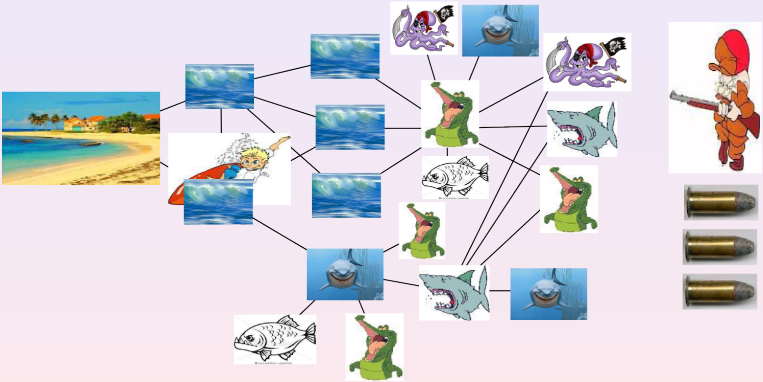
here one more ball is needed

Model



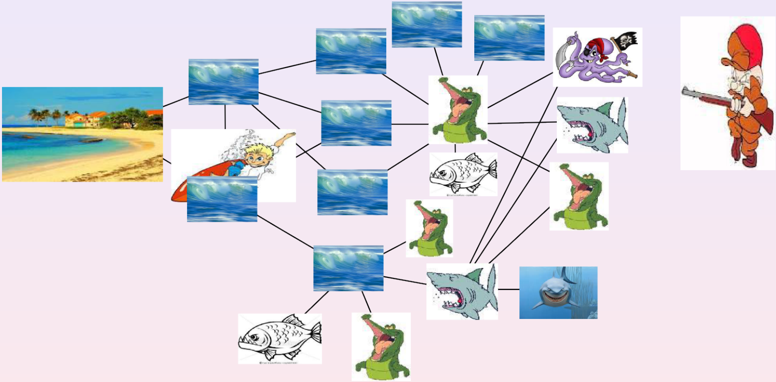
$$degree(beach) \leq \text{amount of balls} \leq max\ \ degree$$

Model



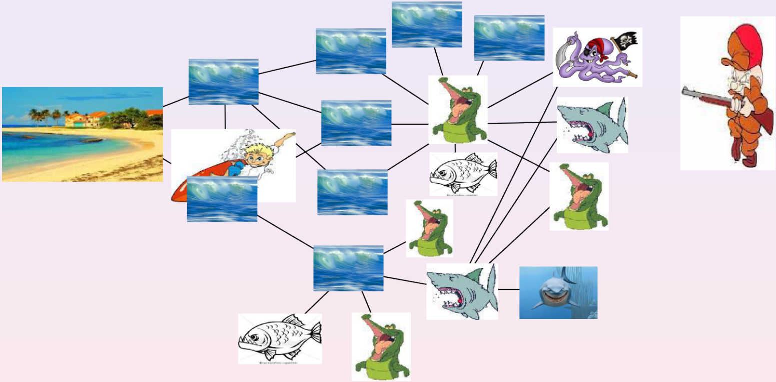
Surfer may move anywhere in its neighborhood

Model



balls may be used to prevent future moves

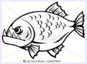
Model



are 3 balls sufficient to make sure Surfer never eaten?

Model: a Two players game

- a *Surfer*  starts from safe homebase v_0 

  in G, a dangerous graph     

- a *Guard*  with some amount k of balls 

Turn by turn:

1. the guard secures $\leq k$ nodes;
2. then, the Surfer may move to an adjacent node.

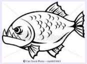
Defeat: Surfer in unsafe node **Victory:** G safe

Minimize amount of balls to win for any Surfer's trajectory

surveillance number of G (**connected**) from v_0: $sn(G, v_0)$

Model: a Two players game

- a *Surfer*  starts from safe homebase v_0 

  in G, a dangerous graph     
- a *Guard*  with some amount k of balls 

Turn by turn:

1. the guard secures $\leq k$ nodes;
2. then, the Surfer may move to an adjacent node.

Defeat: Surfer in unsafe node  **Victory:** G safe 

Minimize amount of balls to win for any Surfer's trajectory

surveillance number of G (**connected**) from v_0: $sn(G, v_0)$

Model: a Two players game

- a *Surfer*  starts from safe homebase v_0 
  in G, a dangerous graph 
- a *Guard*  with some amount k of balls 

Turn by turn:

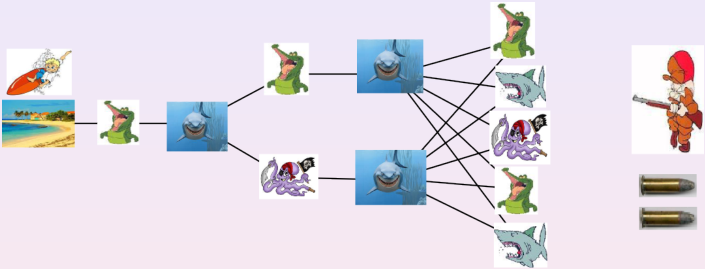
1. the guard secures $\leq k$ nodes;
2. then, the Surfer may move to an adjacent node.

Defeat: Surfer in unsafe node  **Victory:** G safe 

Minimize amount of balls to win for any Surfer's trajectory

surveillance number of G (**connected**) from v_0: $sn(G, v_0)$

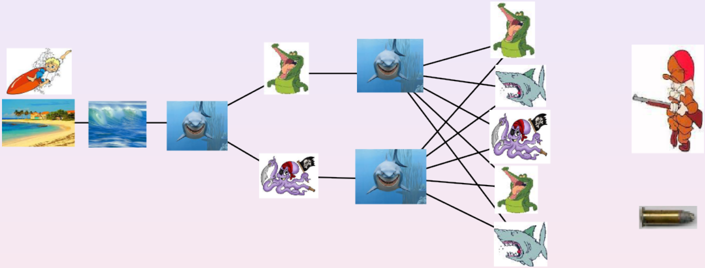
First Example



with 1 ball: after 2 steps, Surfer faces 2 dangerous nodes!!

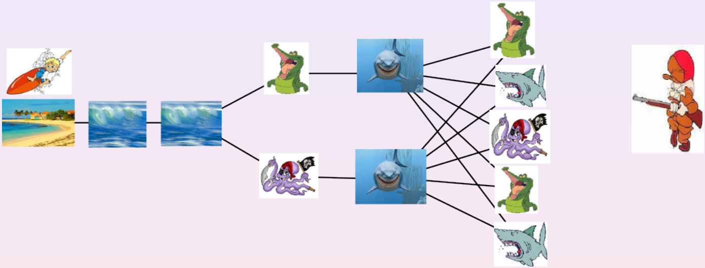
$$sn(G, v_0) > 1$$

First Example



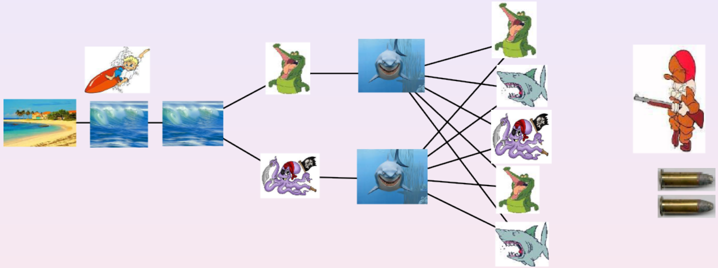
Guard uses his balls

First Example



Guard uses (all) his balls

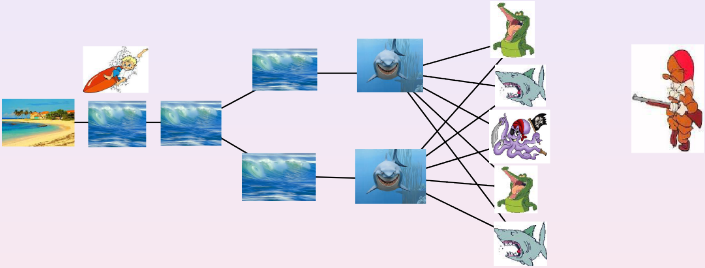
First Example



Guard uses (all) his balls, **then** Surfer may move

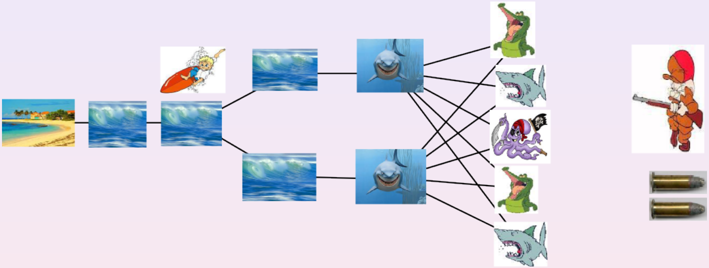
Clearly: worst case if Surfer always move

First Example



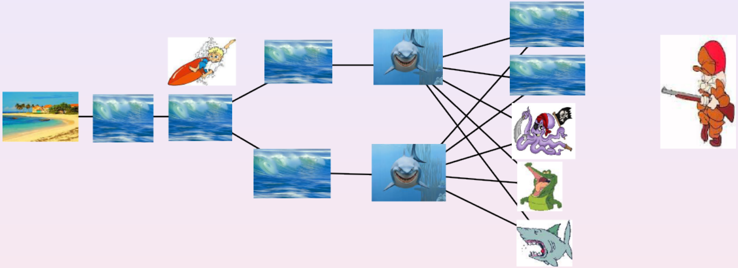
Guard uses (all) his balls, **then** Surfer may move

First Example



Guard uses (all) his balls, **then** Surfer moves

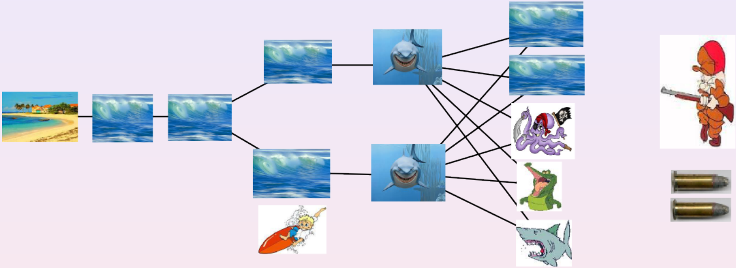
First Example



Guard uses (all) his balls, **then** Surfer moves

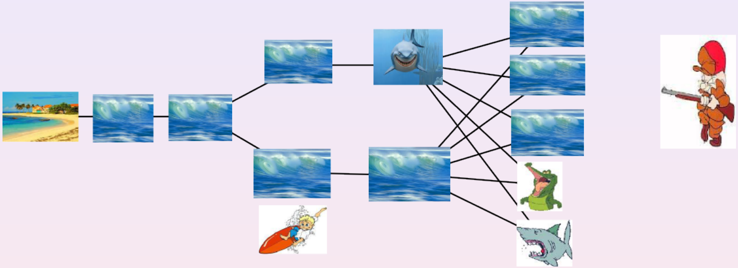
Guard may secure any node in the graph

First Example



Guard uses (all) his balls, **then** Surfer moves

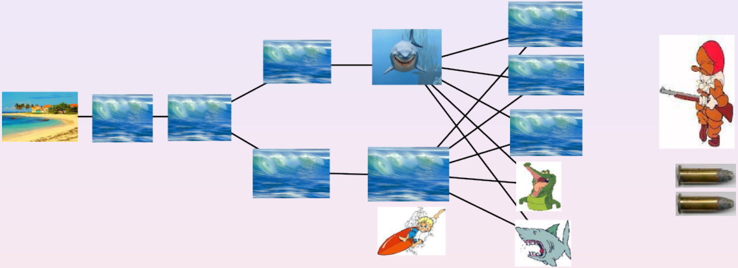
First Example



Guard uses (all) his balls, **then** Surfer moves

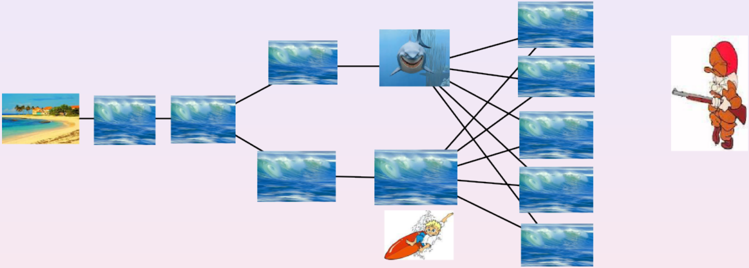
strategy: safe nodes $+$ Surfer's node $\Rightarrow \leq k$ nodes to secure

First Example



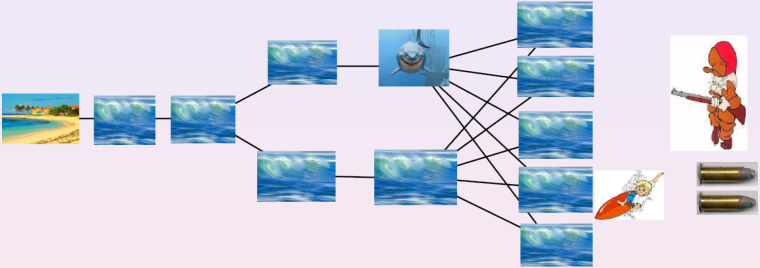
Guard uses (all) his balls, **then** Surfer moves

First Example



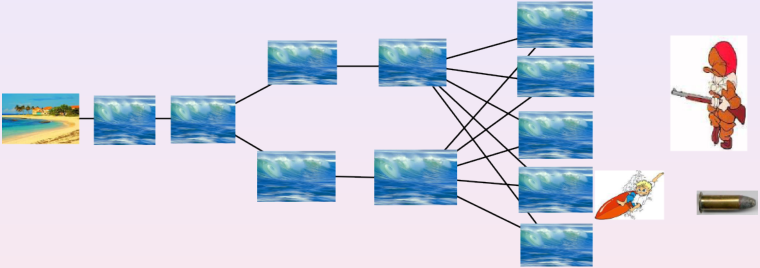
Guard uses (all) his balls, **then** Surfer moves

First Example



Guard uses (all) his balls, **then** Surfer moves

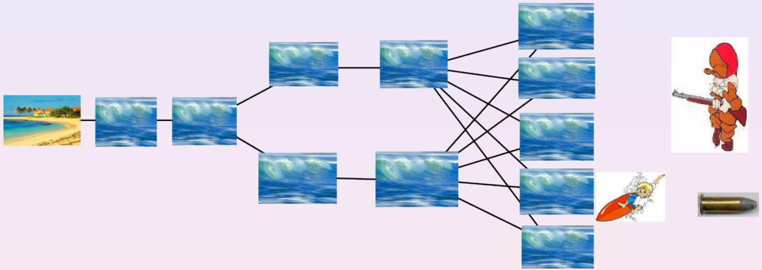
First Example



Guard uses (all) his balls, **then** Surfer moves

All nodes safe: Victory against this trajectory of the Surfer

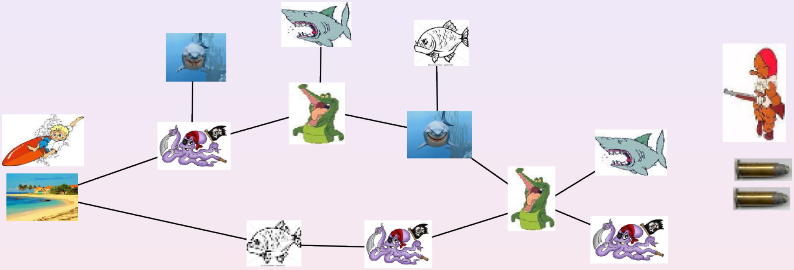
First Example



In this example, all Surfer's trajectory similar (by symmetry)

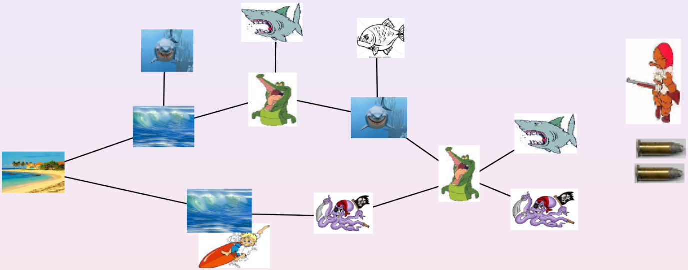
victory whatever Surfer's trajectory $\Rightarrow sn(G, v_0) = 2$

Second Example: which paths for the Surfer?



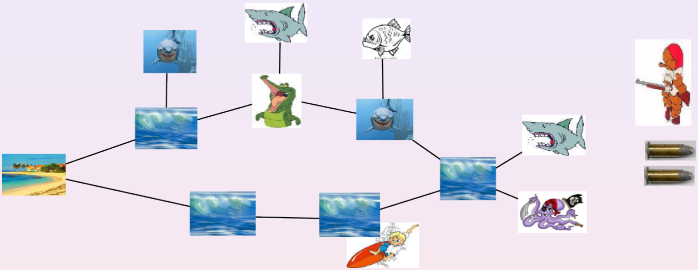
degree of homebase $= 2 \Rightarrow \geq 2$ balls required!! $sn(G, v_0) \geq 2$

Question: Is it more difficult to protect “fast” Surfers?



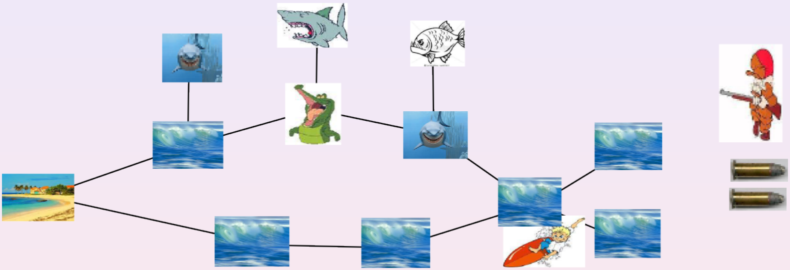
let's try with 2 balls
if Surfer moves anti-clockwise...

Second Example: which paths for the Surfer?



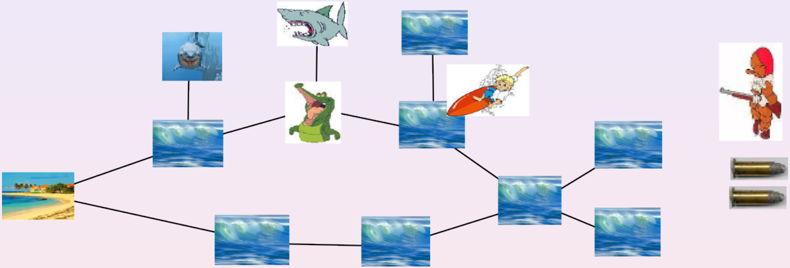
let's try with 2 balls
if Surfer moves anti-clockwise...

Second Example: which paths for the Surfer?



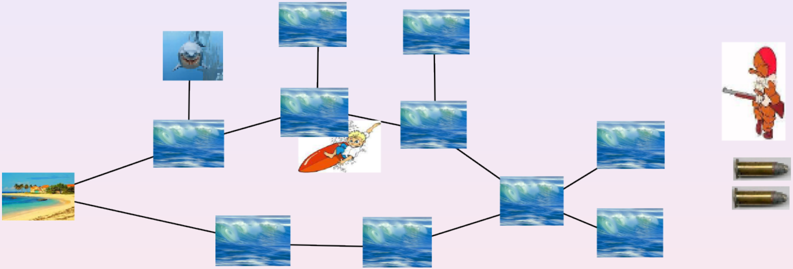
let's try with 2 balls
if Surfer moves anti-clockwise...

Second Example: which paths for the Surfer?



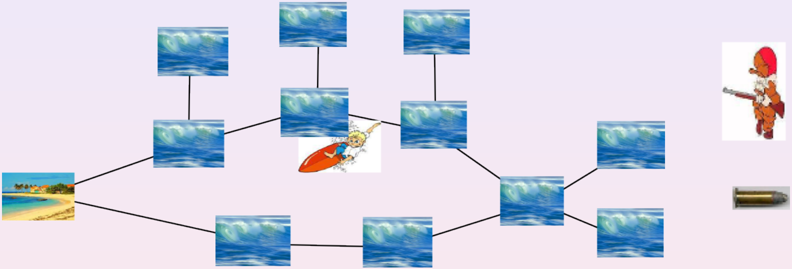
let's try with 2 balls
if Surfer moves anti-clockwise...

Second Example: which paths for the Surfer?



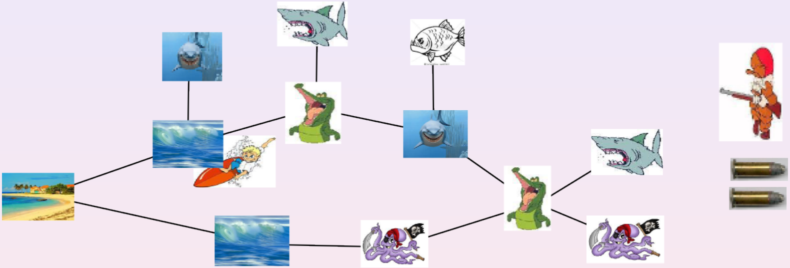
let's try with 2 balls
if Surfer moves anti-clockwise...

Second Example: which paths for the Surfer?



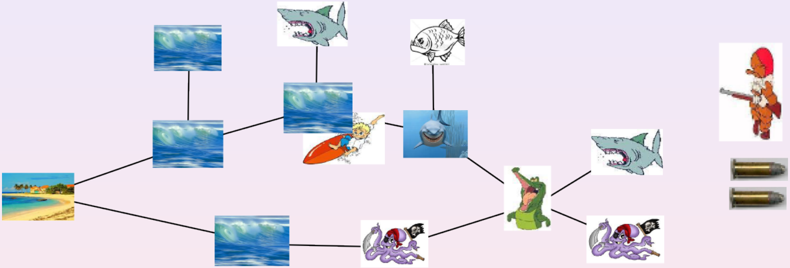
let's try with 2 balls
if Surfer moves anti-clockwise, Guard manage to secure G!!

Second Example: which paths for the Surfer?



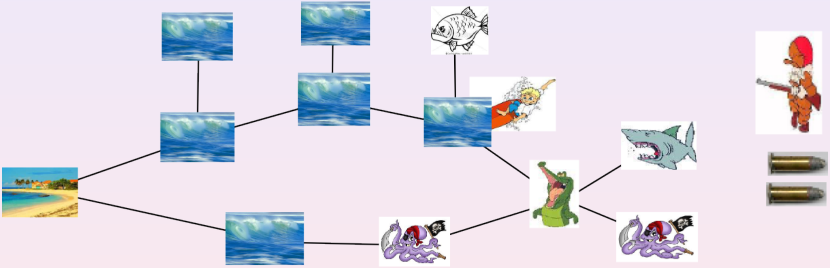
let's try with 2 balls
if Surfer moves clockwise...

Second Example: which paths for the Surfer?



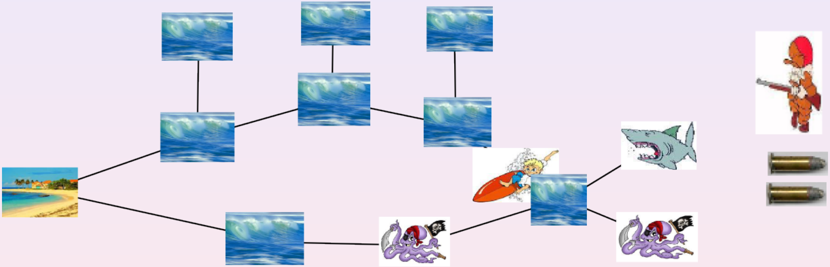
let's try with 2 balls
if Surfer moves clockwise...

Second Example: which paths for the Surfer?



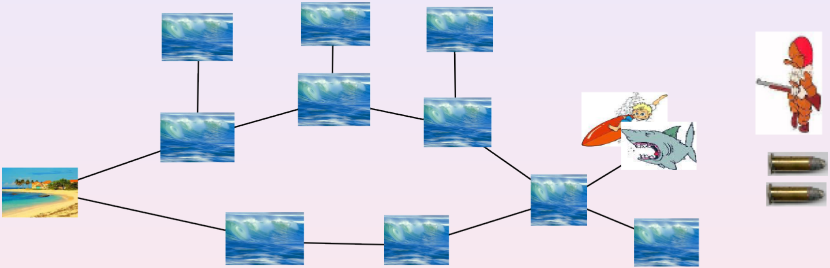
let's try with 2 balls
if Surfer moves clockwise...

Second Example: which paths for the Surfer?



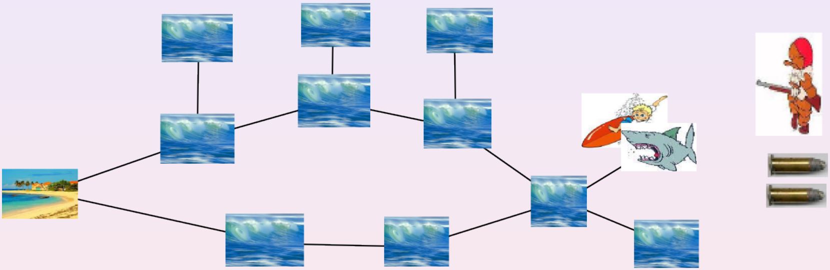
let's try with 2 balls
if Surfer moves clockwise...

Second Example: which paths for the Surfer?



let's try with 2 balls
if Surfer moves clockwise, a shark is happy!!

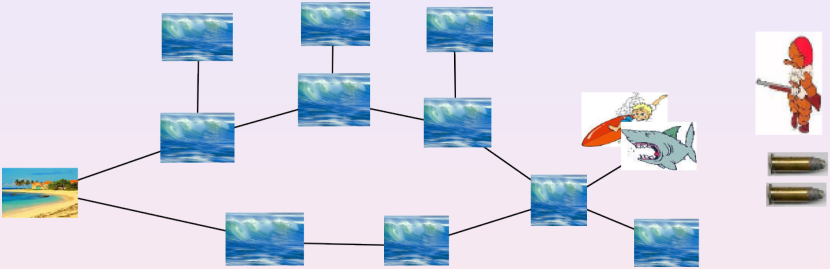
Second Example: which paths for the Surfer?



following a longest path may be more dangerous for the Surfer

we cannot restrict our study to shortest paths :(

Second Example: which paths for the Surfer?

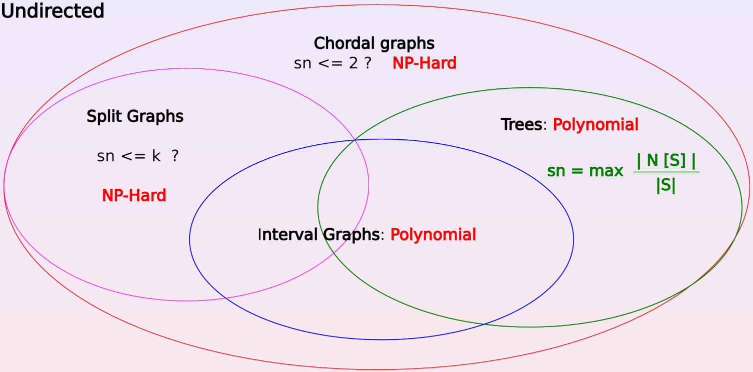


.... however, we can restrict our study to induced paths

Theorem 1: Worst case when Surfer follows induced paths

Restricting Surfer to induced paths does not decrease *sn*

Undirected



Directed

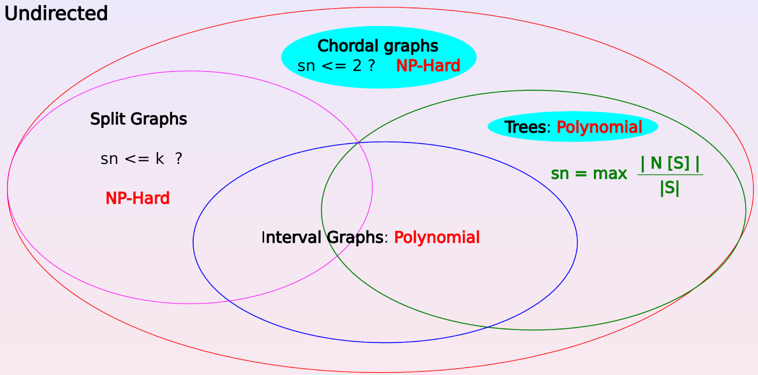
DAGs: sn <=4 ? **P-SPACE-Complete**

DAGs: sn <=2 ? **NP-Hard**

General

(out)-**degree (v0)** $\leq$ **sn** $\leq$ max (degree (v0) ; **max degree** -1) max out-degree

$O(2^n)$ exact algorithm



Directed

DAGs: sn <=4 ? P-SPACE-Complete

DAGs: sn <=2 ? NP-Hard

General

(out)-degree (v0) ≤ sn ≤ max (degree (v0) ; max degree -1)

max out-degree

$O(2^n)$ exact algorithm

Chordal graph: no induced cycle of length > 3.

reduction from

3-Hitting Set (NP-complete)

Inputs:

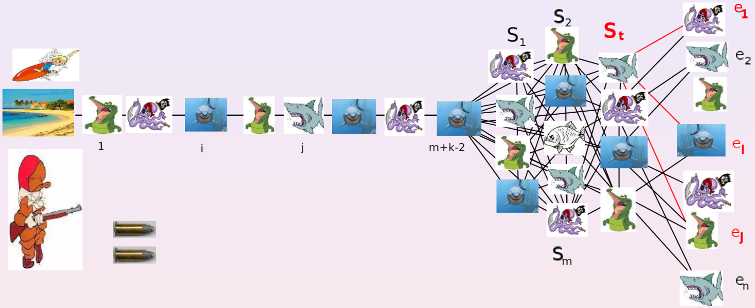
- set $E = \{e_1, \cdots, e_n\}$
- $\mathcal{S} = \{S_1, \cdots, S_\ell = \{e_i, e_j, e_t\}, \cdots, S_m\}$ of triples of E
- integer $k \geq 1$

Question: $\exists?$ $H \subseteq E$ such that

- $|H| \leq k$
- $H \cap S_i \neq \emptyset$ for any $i \leq m$

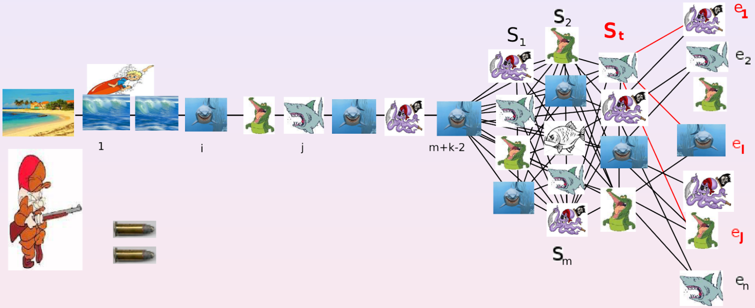
NP-hardness in Chordal Graphs $sn \leq 2$?

Input: $E = \{e_1, \cdots, e_n\}$, $\mathcal{S} = \{S_1, \cdots, S_m\}$ and $k \geq 1$.

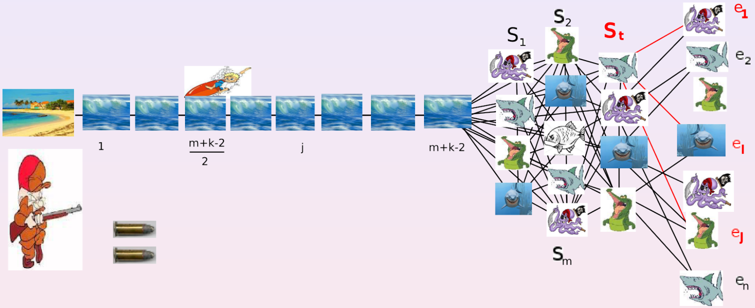


NP-hardness in Chordal Graphs

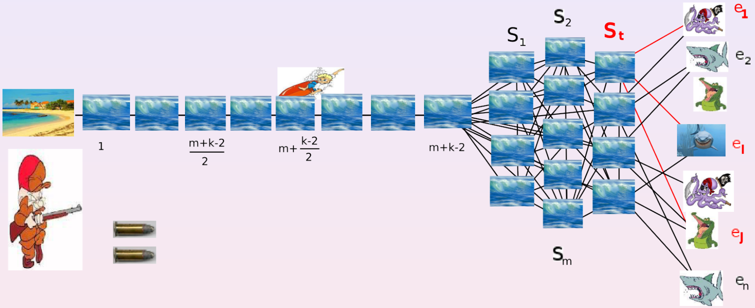
Input: $E = \{e_1, \cdots, e_n\}$, $\mathcal{S} = \{S_1, \cdots, S_m\}$ and $k \geq 1$.



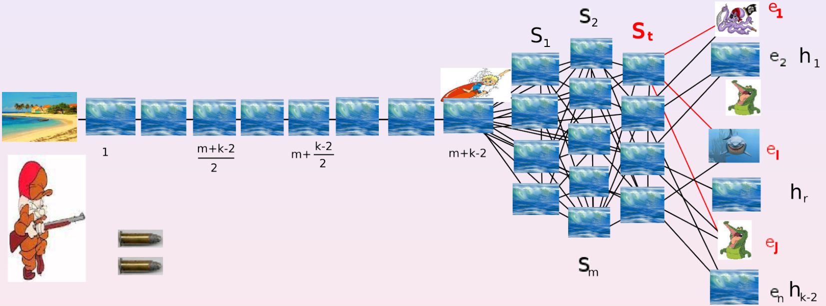
Input: $E = \{e_1, \cdots, e_n\}$, $\mathcal{S} = \{S_1, \cdots, S_m\}$ and $k \geq 1$.



Input: $E = \{e_1, \cdots, e_n\}$, $\mathcal{S} = \{S_1, \cdots, S_m\}$ and $k \geq 1$.



Input: $E = \{e_1, \cdots, e_n\}$, $\mathcal{S} = \{S_1, \cdots, S_m\}$ and $k \geq 1$.



$$sn(G, v_0) \leq 2 \Leftrightarrow hittingSet(E, \mathcal{S}) \leq k$$

DAG: Directed Acyclic Graph.

reduction from

3-QSAT (PSPACE-complete)

Inputs:

- set of variables $\{x_1, \cdots, x_n, y_1, \cdots, y_n\}$
- logical formula $\Phi(x_1, \cdots, x_n, y_1, \cdots, y_n)$

Question: is it true?

$$\forall x_1 \exists y_1 \forall x_2 \exists y_2 \cdots \forall x_n \exists y_n \;\; \Phi(x_1, \cdots, x_n, y_1, \cdots, y_n)$$

DAG: Directed Acyclic Graph.

reduction from

3-QSAT (PSPACE-complete)

Inputs:

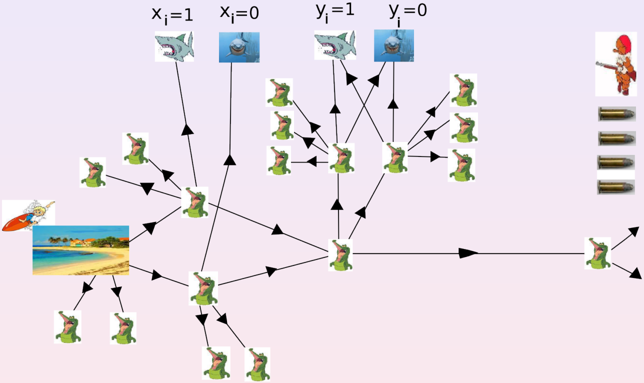
- set of variables $\{x_1, \cdots, x_n, y_1, \cdots, y_n\}$
- logical formula $\Phi(x_1, \cdots, x_n, y_1, \cdots, y_n)$

Question: is it true?

$$\forall x_1 \exists y_1 \forall x_2 \exists y_2 \cdots \forall x_n \exists y_n \ \Phi(x_1, \cdots, x_n, y_1, \cdots, y_n)$$

- "$\forall x_i$" depends on Surfer's path
- "$\exists y_i$" depends on Guard's strategy

Input: $\{x_1, \cdots, x_n, y_1, \cdots, y_n\}$ and $\Phi(x_1, \cdots, x_n, y_1, \cdots, y_n)$

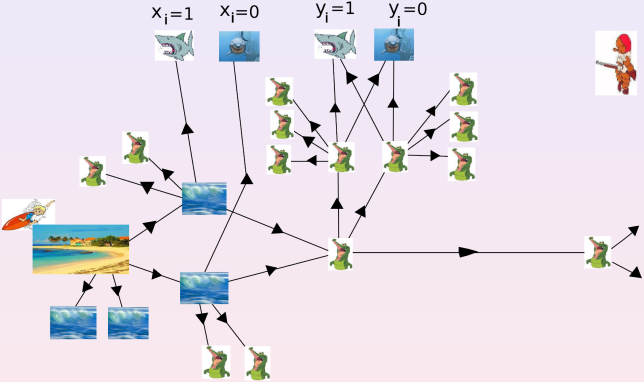


Gadget for x_i and y_i:

When Surfer reaches the right, 2 "sharks" have left

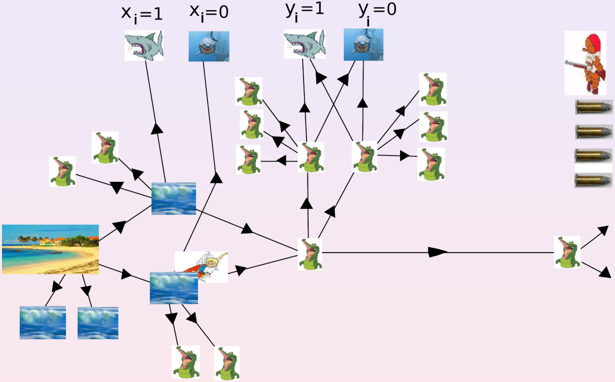
$\Rightarrow$ assignment of x_i (Surfer choice) and y_i (Guard choice)

Input: $\{x_1, \cdots, x_n, y_1, \cdots, y_n\}$ and $\Phi(x_1, \cdots, x_n, y_1, \cdots, y_n)$



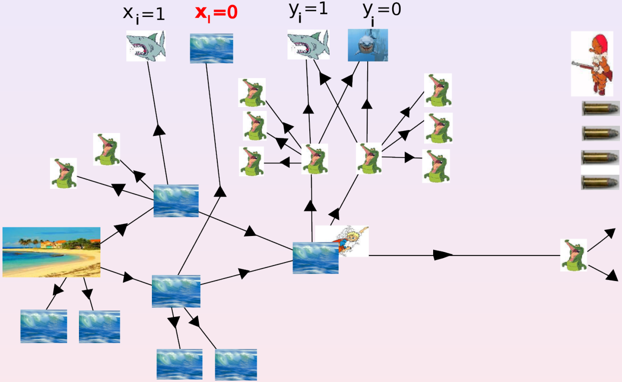
Gadget for x_i and y_i:
When Surfer reaches the right, 2 "sharks" have left
$\Rightarrow$ assignment of x_i (Surfer choice) and y_i (Guard choice)

Input: $\{x_1, \cdots, x_n, y_1, \cdots, y_n\}$ and $\Phi(x_1, \cdots, x_n, y_1, \cdots, y_n)$



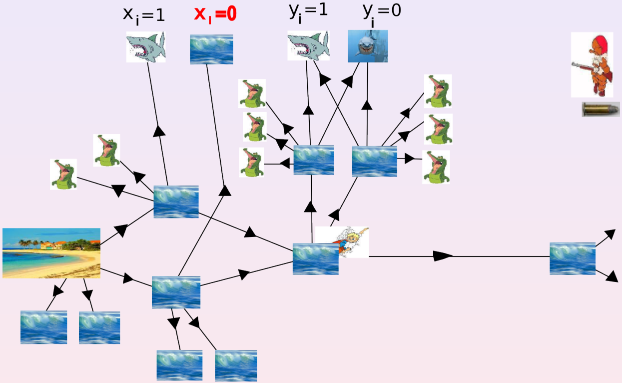
Gadget for x_i and y_i:
When Surfer reaches the right, 2 "sharks" have left
$\Rightarrow$ assignment of x_i (Surfer choice) and y_i (Guard choice)

Input: $\{x_1, \cdots, x_n, y_1, \cdots, y_n\}$ and $\Phi(x_1, \cdots, x_n, y_1, \cdots, y_n)$



Gadget for x_i and y_i:
When Surfer reaches the right, 2 "sharks" have left
$\Rightarrow$ assignment of x_i (Surfer choice) and y_i (Guard choice)

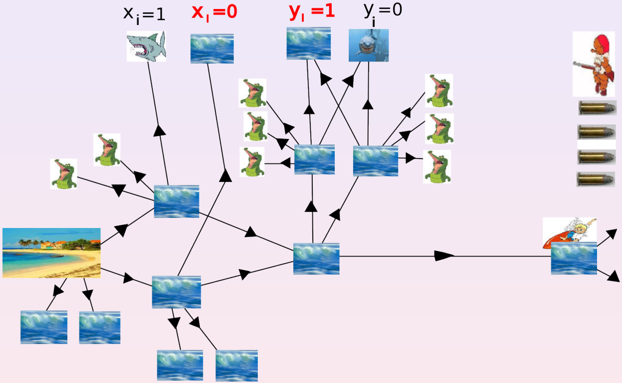
Input: $\{x_1, \cdots, x_n, y_1, \cdots, y_n\}$ and $\Phi(x_1, \cdots, x_n, y_1, \cdots, y_n)$



Gadget for x_i and y_i:
When Surfer reaches the right, 2 "sharks" have left
$\Rightarrow$ assignment of x_i (Surfer choice) and y_i (Guard choice)

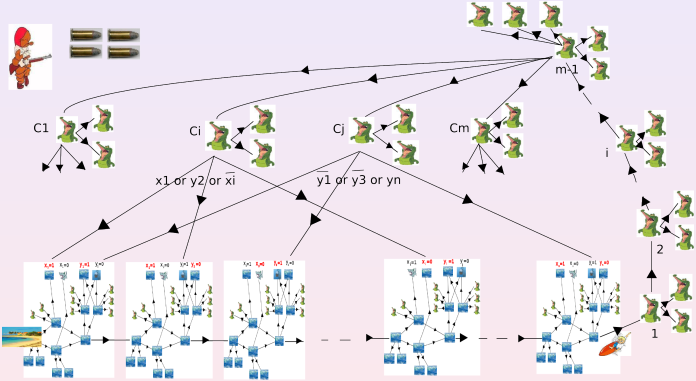
PSPACE-hardness in DAGs

Input: $\{x_1, \cdots, x_n, y_1, \cdots, y_n\}$ and $\Phi(x_1, \cdots, x_n, y_1, \cdots, y_n)$



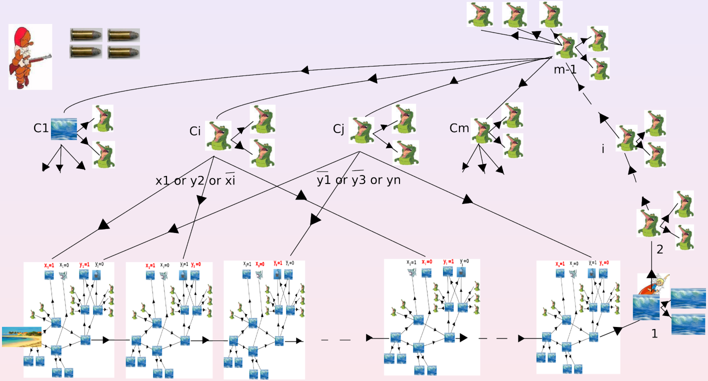
Gadget for x_i and y_i:
When Surfer reaches the right, 2 "sharks" have left
⇒ assignment of x_i (Surfer choice) and y_i (Guard choice)

Input: $\{x_1, \cdots, x_n, y_1, \cdots, y_n\}$ and $\Phi(x_1, \cdots, x_n, y_1, \cdots, y_n)$



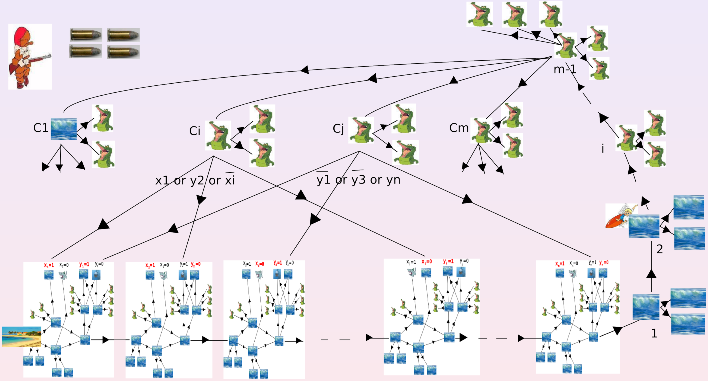
When Surfer reaches the bottom-right, all variables have been assigned: x_i's by Surfer, y_i's by Guard

Input: $\{x_1, \cdots, x_n, y_1, \cdots, y_n\}$ and $\Phi(x_1, \cdots, x_n, y_1, \cdots, y_n)$



When Surfer reaches the bottom-right, all variables have been assigned: x_i's by Surfer, y_i's by Guard

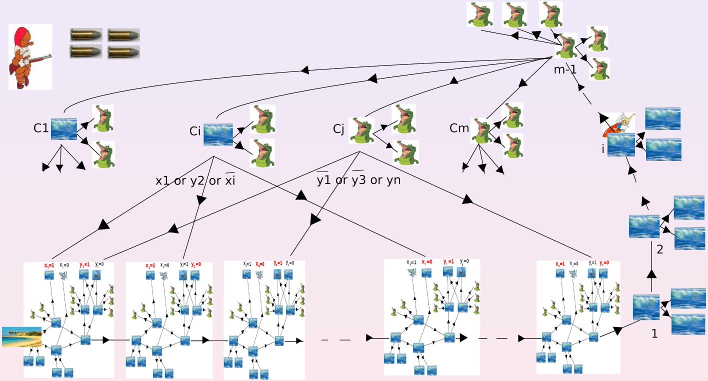
Input: $\{x_1, \cdots, x_n, y_1, \cdots, y_n\}$ and $\Phi(x_1, \cdots, x_n, y_1, \cdots, y_n)$



When Surfer reaches the bottom-right, all variables have been assigned: x_i's by Surfer, y_i's by Guard

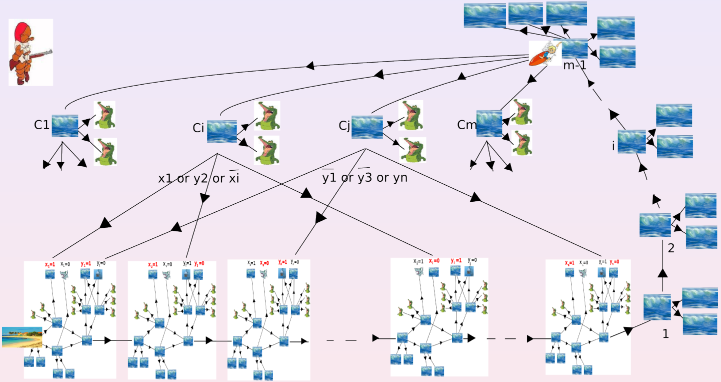
PSPACE-hardness in DAGs

Input: $\{x_1, \cdots, x_n, y_1, \cdots, y_n\}$ and $\Phi(x_1, \cdots, x_n, y_1, \cdots, y_n)$



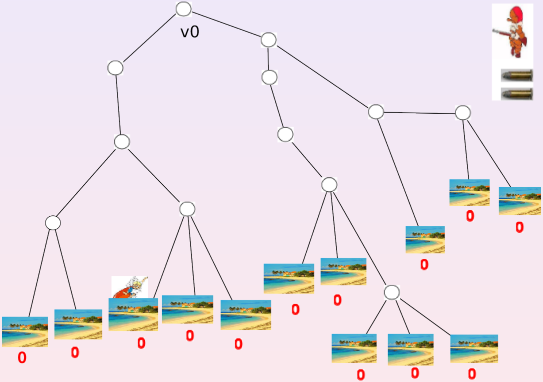
When Surfer reaches the bottom-right, all variables have been assigned: x_i's by Surfer, y_i's by Guard

Input: $\{x_1, \cdots, x_n, y_1, \cdots, y_n\}$ and $\Phi(x_1, \cdots, x_n, y_1, \cdots, y_n)$



$$sn(D, v_0) \leq 4 \Leftrightarrow \forall x_1 \cdots \exists y_n \Phi \ \ true$$

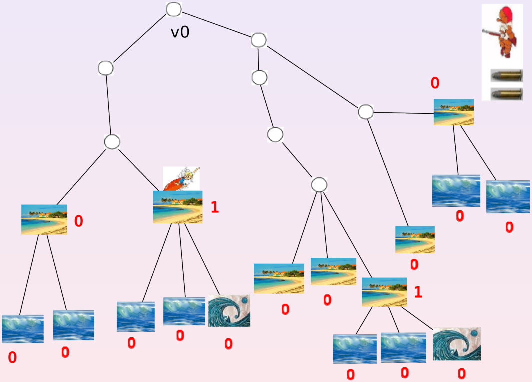
Dynamic Programming: amount of balls fixed to k



$\color{red}{label(v)} = \min \#$ of nodes to be secured **before starting** to win in T_v starting in v, with k balls $\quad \color{red}{label(leaf) = 0}$

Polynomial Algorithm in Trees

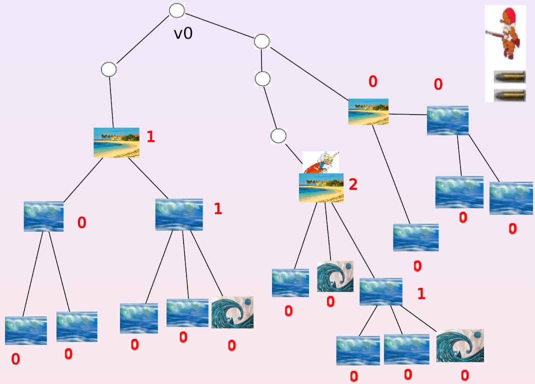
Dynamic Programming: amount of balls fixed to k



$\color{red}{label(v)}$ = min # of nodes to be secured **before starting** to win in T_v starting in v, with k balls

Polynomial Algorithm in Trees

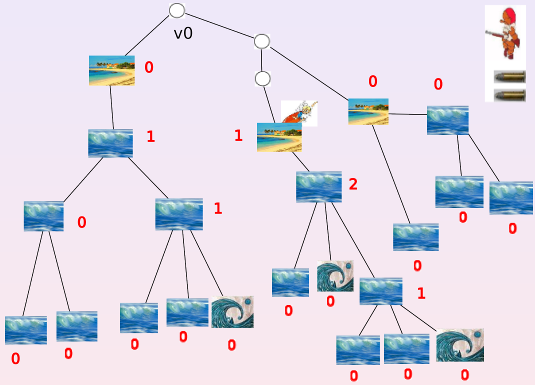
Dynamic Programming: amount of balls fixed to k



$$label(v) = \max\{0; deg(v) - k + \sum_{w \ child \ of \ v} label(w)\}$$

Polynomial Algorithm in Trees

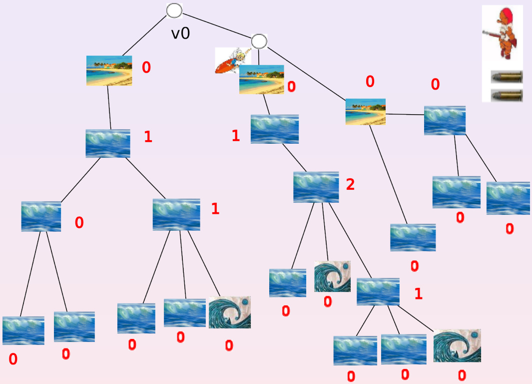
Dynamic Programming: amount of balls fixed to k



$$label(v) = \max\{0; deg(v) - k + \sum_{w \text{ child of } v} label(w)\}$$

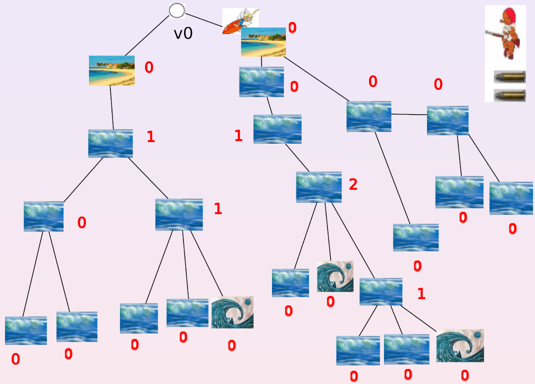
Polynomial Algorithm in Trees

Dynamic Programming: amount of balls fixed to k



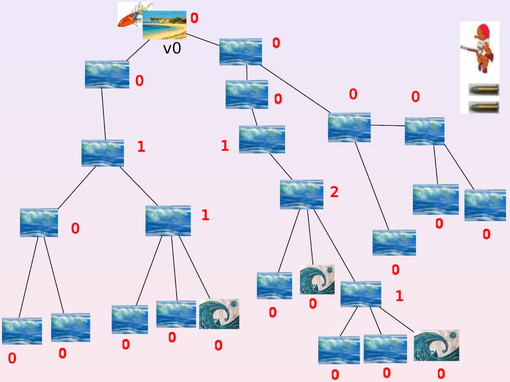
$$label(v) = \max\{0; deg(v) - k + \sum_{w \ child \ of \ v} label(w)\}$$

Dynamic Programming: amount of balls fixed to k



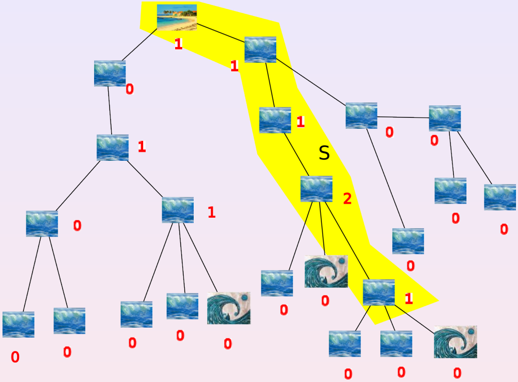
$$label(v) = \max\{0; deg(v) - k + \sum_{w \ child \ of \ v} label(w)\}$$

Dynamic Programming: amount of balls fixed to k



$$sn(T, v_0) \leq k \Leftrightarrow label(v_0) = 0$$

Polynomial Algorithm in Trees



Upper bound: if $k < sn(T, v_0)$, $label(v_0) > 0$
maximal subtree S of nodes v with $label(v) > 0$ containing v_0

$\frac{|N[S]|-1}{|S|} > k$ hence max $\frac{|N[S]|-1}{|S|} \geq sn(T, v_0)$

Positive Results

Combinatorial characterization in Trees

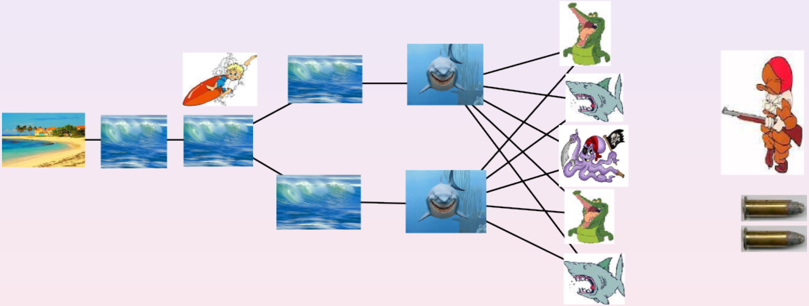
For any tree T, $v_0 \in V(T)$, $sn(T, v_0) = \max \lceil \frac{|N[S]|-1}{|S|} \rceil$, taken for any subtree S containing v_0.

Exact Algorithms

- $O(2^n)$ algorithm in n-node graphs;
- $sn(T, v_0)$ can be computed in time $O(n \log n)$ in any n-node tree T and for any $v_0 \in V(T)$;
- $sn(G, v_0)$ can be computed in time $O(n \cdot \Delta^3)$ in any n-node interval graph G with maximum degree Δ and for any $v_0 \in V(T)$.

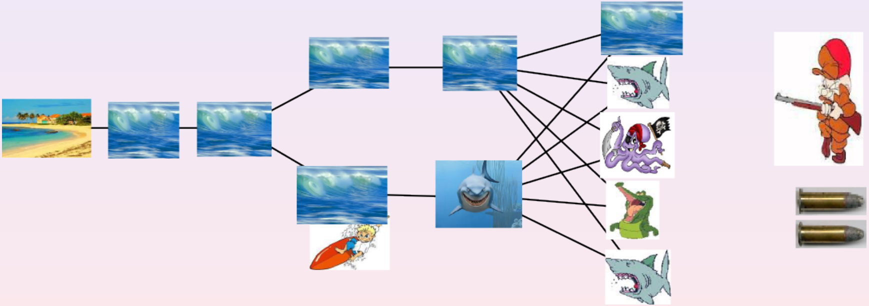
Further Work: Connected Variant

Constraint: safe vertices must induce a connected subgraph



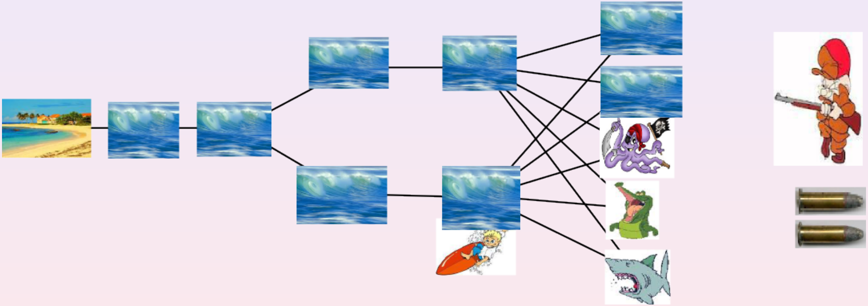
Further Work: Connected Variant

Constraint: safe vertices must induce a connected subgraph



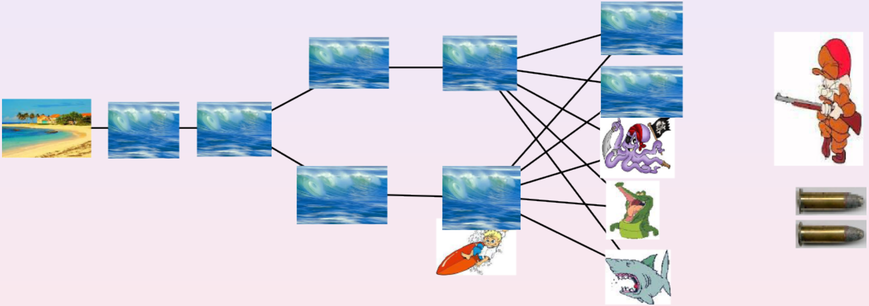
Further Work: Connected Variant

Constraint: safe vertices must induce a connected subgraph



Further Work: Connected Variant

Constraint: safe vertices must induce a connected subgraph



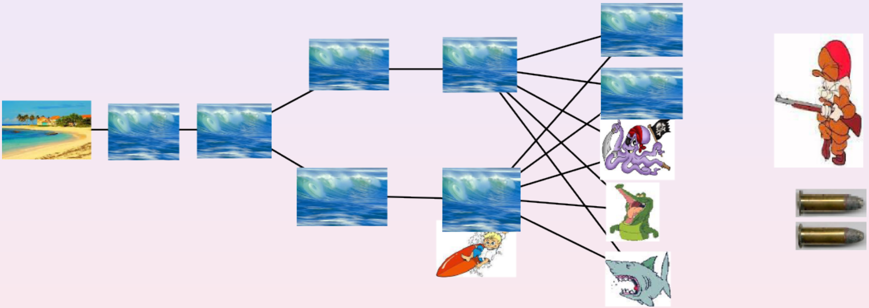
Connectivity costs:

connected-$sn(G, v_0) = 3 > sn(G, v_0) = 2$

All previous results hold for the connected variant

Further Work: Connected Variant

Constraint: safe vertices must induce a connected subgraph



Connectivity costs:

connected-$sn(G, v_0) = 3 > sn(G, v_0) = 2$

$\exists$? G and v_0 such that $c\text{-}sn(G, v_0) \geq sn(G, v_0) + 2$????

Open Questions

- complexity in bounded degree graphs?
  (polynomial if $\Delta \leq 3$)
- complexity in bounded treewidth graphs?
- $\exists ? c < 2$ and $O(c^n)$ algorithm in n-node graphs?
- $sn(G, v_0) = \max \lceil \frac{|N[S]|-1}{|S|} \rceil$, taken for any connected subgraph S containing v_0?
- cost of connectivity? $\frac{connected\ \ sn}{sn} \leq cte$?
  $\exists ?$ G and v_0 such that $c\text{-}sn(G, v_0) \geq sn(G, v_0) + 2$????
- ...

Thank you for your attention[1]

[1] No seafood... no animal has been hurt when preparing this talk.