Request Complexity of VNet Topology Extraction: Dictionary-Based Attacks

Yvonne-Anne Pignolet, Stefan Schmid and *Gilles Tredan*

4 mai 2013

Introduction

- Cloud infrastructure = **secret**
  - Strategic reasons
  - Security reasons: bottlenecks,...
- Cloud providers now rent *virtual networks*
- Cloud provider answers customer requests:
  - **Yes** or **No**

Cloud providers

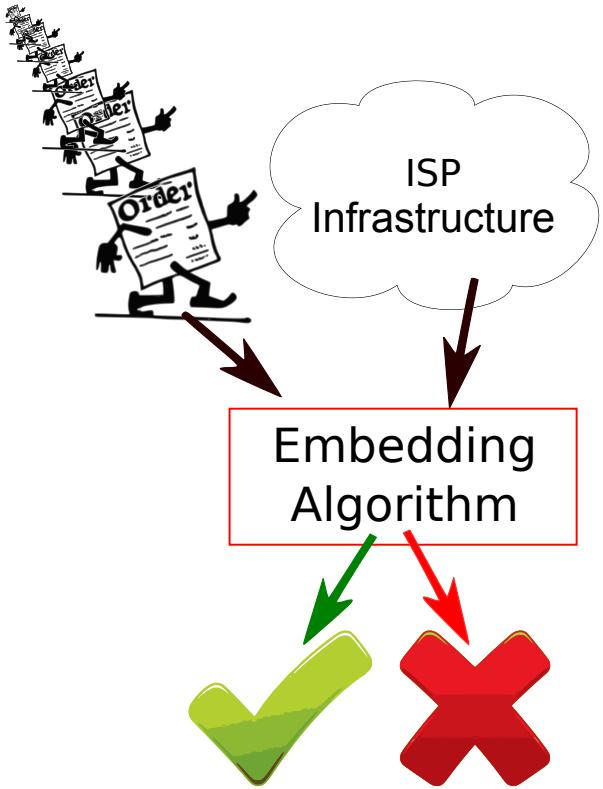
Do these answers leak information about the topology ?

Our model

Cloud Infrastructure H = static symmetric graph

Our questions

- Can we infer ISP infrastructure H using only ISP answers ?
- How many requests do we need ?



A request:

- A graph G to realise
- Along with capacity requirements $\in [0, 1]$

In the following: only unit capacities

Our model

Cloud Infrastructure H = static symmetric graph

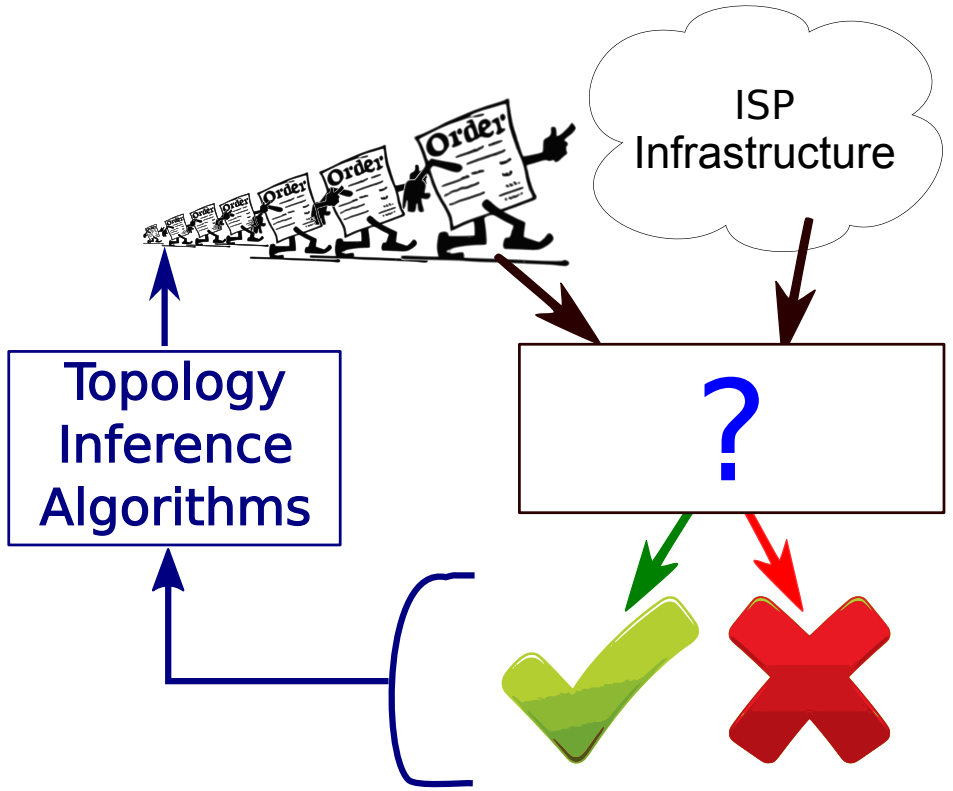
Our questions

- Can we infer ISP infrastructure H using only ISP answers ?
- How many requests do we need ?

A request:

- A graph G to realise
- Along with capacity requirements $\in [0, 1]$

In the following: only unit capacities



Embedding properties

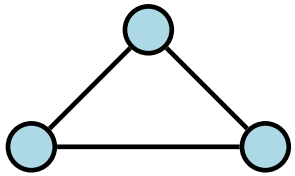
$G \mapsto^{?} H$: **Yes** if:

- Every node in G is mapped to exactly one node in H
- Every edge in G is mapped to exactly one *path* in H

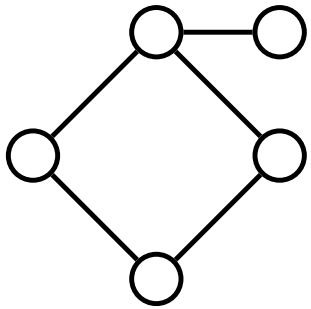
path in $H \Rightarrow$

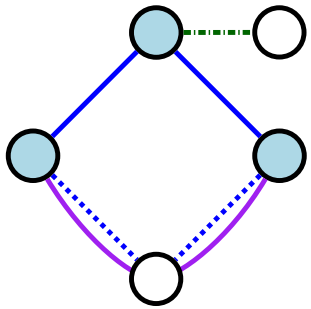
- each node on the path is a *relay* node
- relaying costs ϵ for each node on the path

Request



Substrate





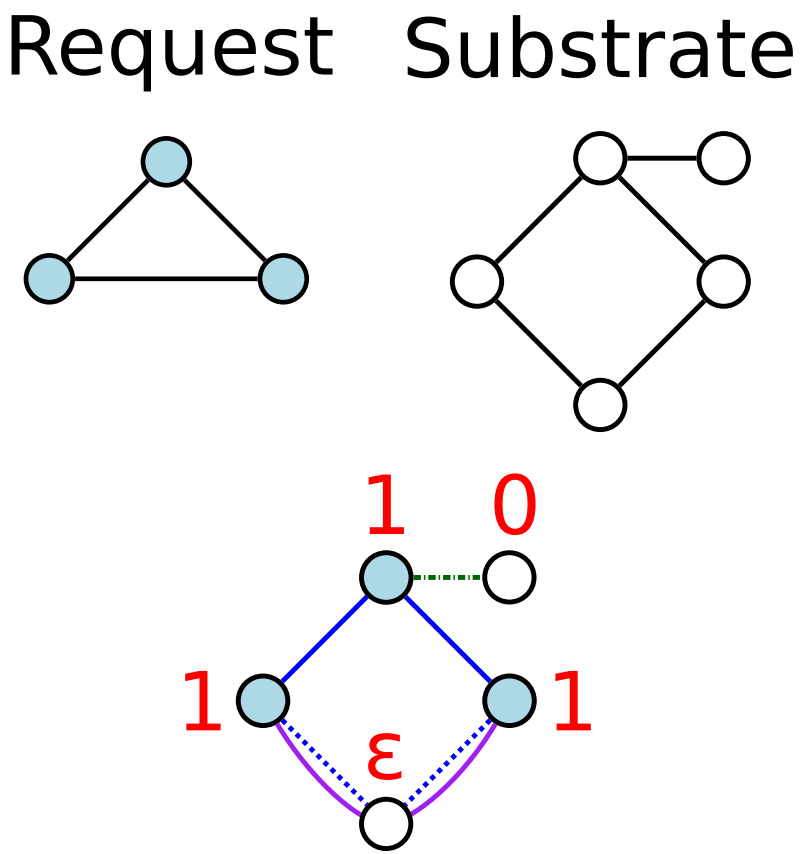
Embedding properties

$G \mapsto^{?} H$: **Yes** if:

- Every node in G is mapped to exactly one node in H
- Every edge in G is mapped to exactly one *path* in H

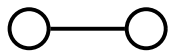
path in $H \Rightarrow$

- each node on the path is a *relay* node
- relaying costs ϵ for each node on the path



Tree

If we know H is a tree

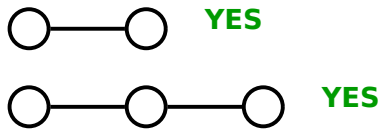


Principle: **grow** a topology in H:
$G_1, \ldots G_k$ s.t. $\forall i < j, G_i \subset G_j$

1. find a diameter
2. mark extremities explored.
3. for each unexplored vertex
   1. append longest chains
   2. mark extremity explored
4. mark vertex explored

Tree

If we know H is a tree

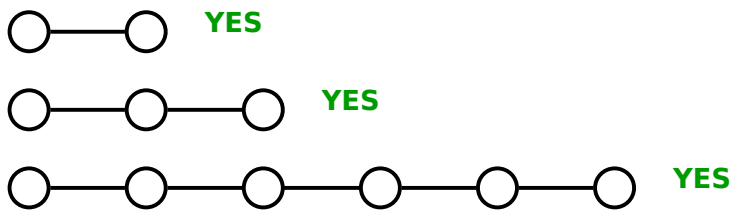


Principle: **grow** a topology in H:
$G_1, \ldots G_k$ s.t. $\forall i < j, G_i \subset G_j$

1. find a diameter
2. mark extremities explored.
3. for each unexplored vertex
   1. append longest chains
   2. mark extremity explored
4. mark vertex explored

Tree

If we know H is a tree



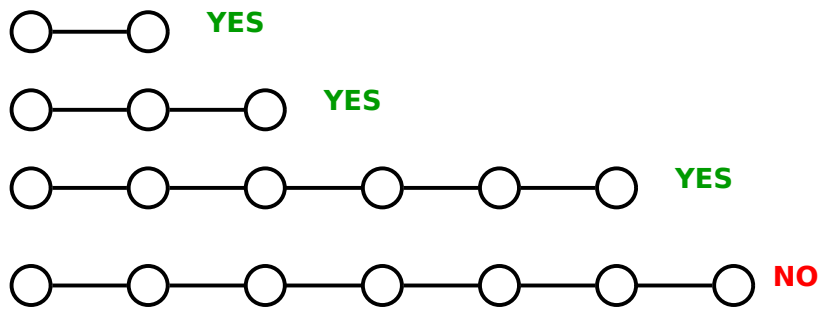
Principle: **grow** a topology in H:
$G_1, \ldots G_k$ s.t. $\forall i < j, G_i \subset G_j$

1. find a diameter
2. mark extremities explored.
3. for each unexplored vertex
   1. append longest chains
   2. mark extremity explored
4. mark vertex explored

Tree

If we know H is a tree

Principle: **grow** a topology in H:
$G_1, \ldots G_k$ s.t. $\forall i < j, G_i \subset G_j$

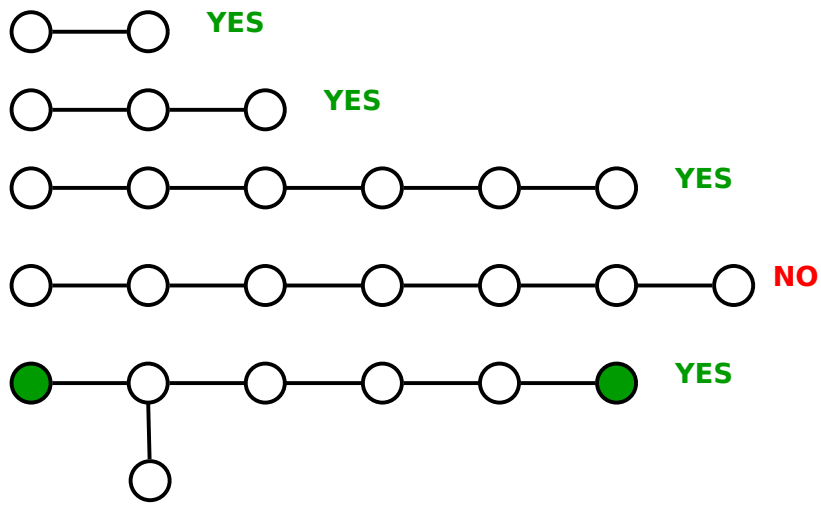


1. find a diameter
2. mark extremities explored.
3. for each unexplored vertex
   1. append longest chains
   2. mark extremity explored
4. mark vertex explored

Tree

If we know H is a tree

Principle: **grow** a topology in H:
$G_1, \ldots G_k$ s.t. $\forall i < j, G_i \subset G_j$



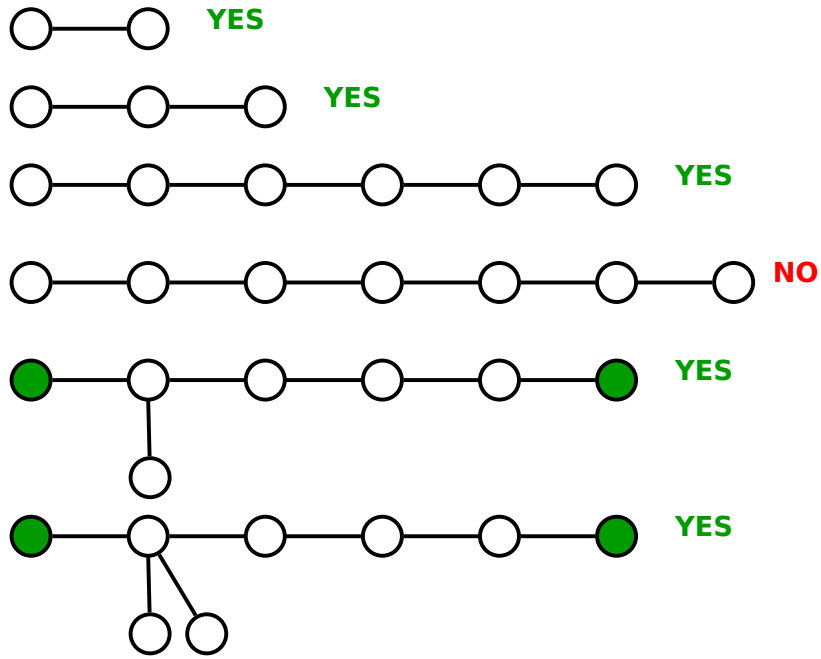
1. find a diameter
2. mark extremities explored.
3. for each unexplored vertex
   1. append longest chains
   2. mark extremity explored
4. mark vertex explored

Tree

If we know H is a tree

Principle: **grow** a topology in H:
$G_1, \ldots G_k$ s.t. $\forall i < j, G_i \subset G_j$

1. find a diameter
2. mark extremities explored.
3. for each unexplored vertex
   1. append longest chains
   2. mark extremity explored
4. mark vertex explored

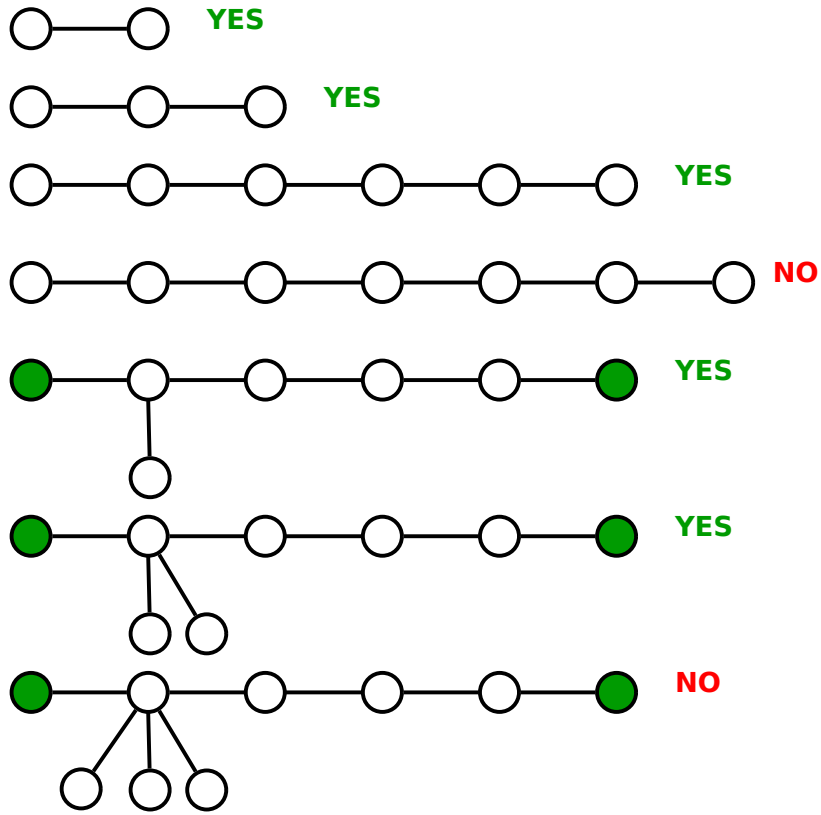


Tree

If we know H is a tree

Principle: **grow** a topology in H:
$G_1, \ldots G_k$ s.t. $\forall i < j, G_i \subset G_j$

1. find a diameter
2. mark extremities explored.
3. for each unexplored vertex
   1. append longest chains
   2. mark extremity explored
4. mark vertex explored

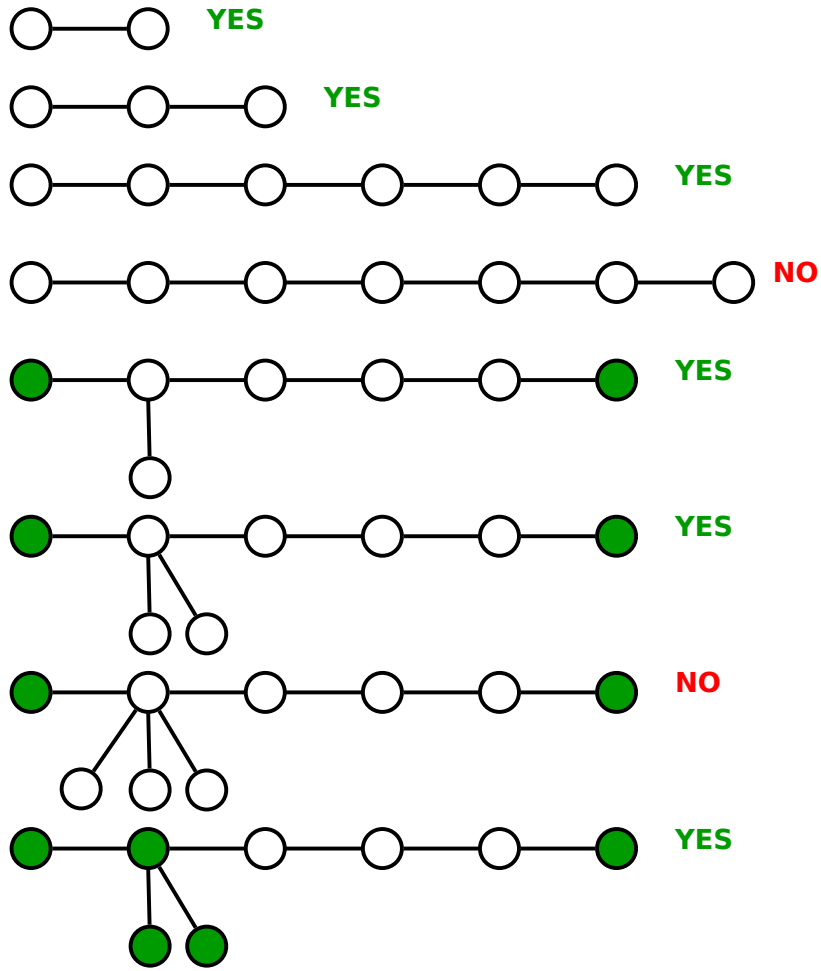


Tree

If we know H is a tree

Principle: **grow** a topology in H:
$G_1, \ldots G_k$ s.t. $\forall i < j, G_i \subset G_j$

1. find a diameter
2. mark extremities explored.
3. for each unexplored vertex
   1. append longest chains
   2. mark extremity explored
4. mark vertex explored



Analysis

- **Correctness**: H is connected, we explore all the possible paths
- **Cost**: each **Yes** we discover at least one more node. Each **No** we mark one node explored $\Rightarrow$

$$\texttt{cost}(\mathit{Tree}) = \theta(n)$$

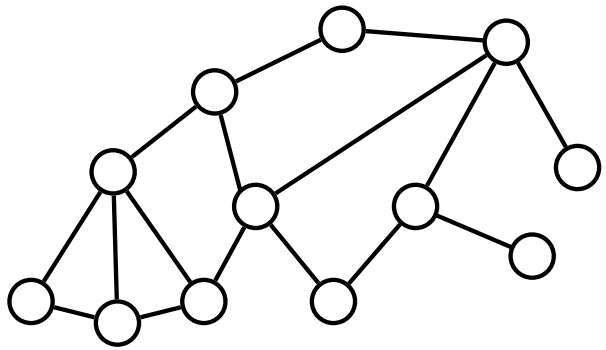
- **Optimality**: $|\text{possible trees of size } n| \approx 2.96^n/n^{5/2}$ ([Plotkin,Rosenthal94])
  $\Rightarrow$at least $log(2.96^n/n^{5/2}) = \Omega(n)$ requests using binary answers

$$\mathit{Tree} \text{ is asymptotically optimal}$$

General graphs

Works for general graphs!

- *Tree* provides a spanning tree on general graphs
- Algorithm sketch
  - Let $T = Tree(H)$
  - $\forall i,j \in V(T)^2$ try $T \cup \{(i,j)\}$



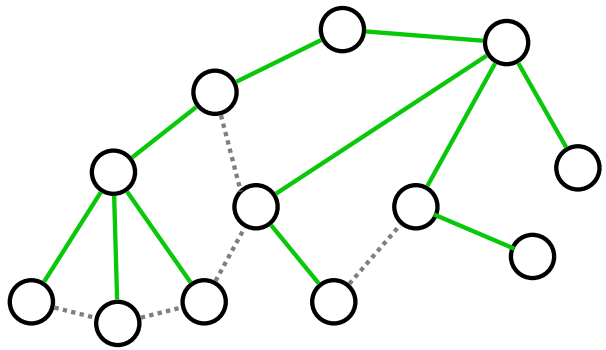
Analysis

- Complexity: n^2 (we test each pair)
- Lower bound:

  $|\{H \text{ of } n \text{ nodes}\}| \geq \frac{2^{\binom{n}{2}}}{n!}$

  $\Rightarrow \Omega(n^2)$ in a binary answer model

  $\Rightarrow$asymptotically optimal



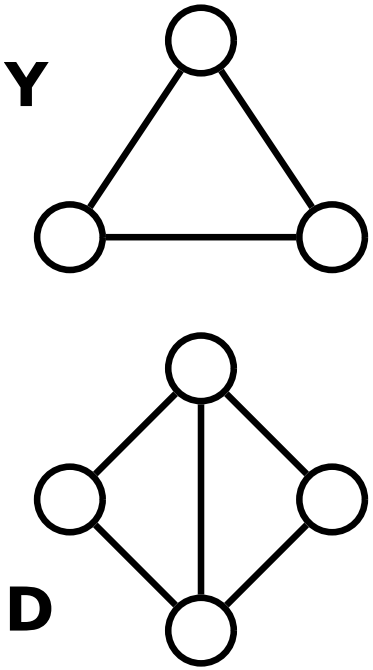
Can we do better ?

E.g. $\texttt{cost}(\mathit{Discovery}) = \theta(|E(H)|)$

Sometimes Yes!

Idea: Compose « little » graph structures

- *Motifs*:minimal bi-connected subgraph that reserves a hole H subgraph
- $\Rightarrow H$ is a **tree** of such motifs !
- Approach: compose motifs.
- However composition is not straigthforward
  $\Rightarrow$need to have composition rules:
  dictionary!
- Key idea: $\mapsto$ on motif set is a poset

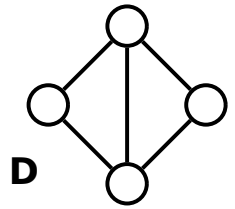


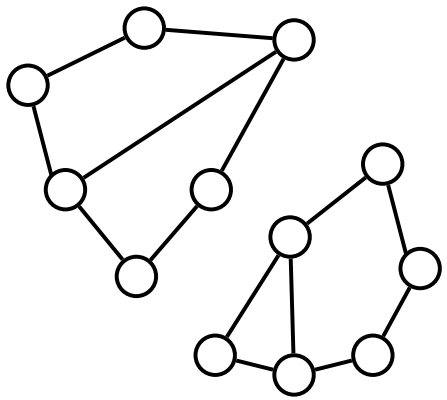
$\Rightarrow$Fast if dictionary is small

Sometimes Yes!

Idea: Compose « little » graph structures

- *Motifs*:minimal bi-connected subgraph that reserves a hole H subgraph
- $\Rightarrow H$ is a **tree** of such motifs !
- Approach: compose motifs.
- However composition is not straigthforward
  $\Rightarrow$need to have composition rules:
  dictionary!
- Key idea: $\mapsto$ on motif set is a poset





$\Rightarrow$Fast if dictionary is small

Evaluation

Evaluation

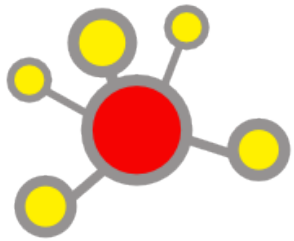
- RocketFuel topologies
- A European power distribution grid

For each topology:

- Identify tree nodes
- Identify relay nodes
- Extract motifs

Tools:

- R
- ggplot2
- R::igraph



Results

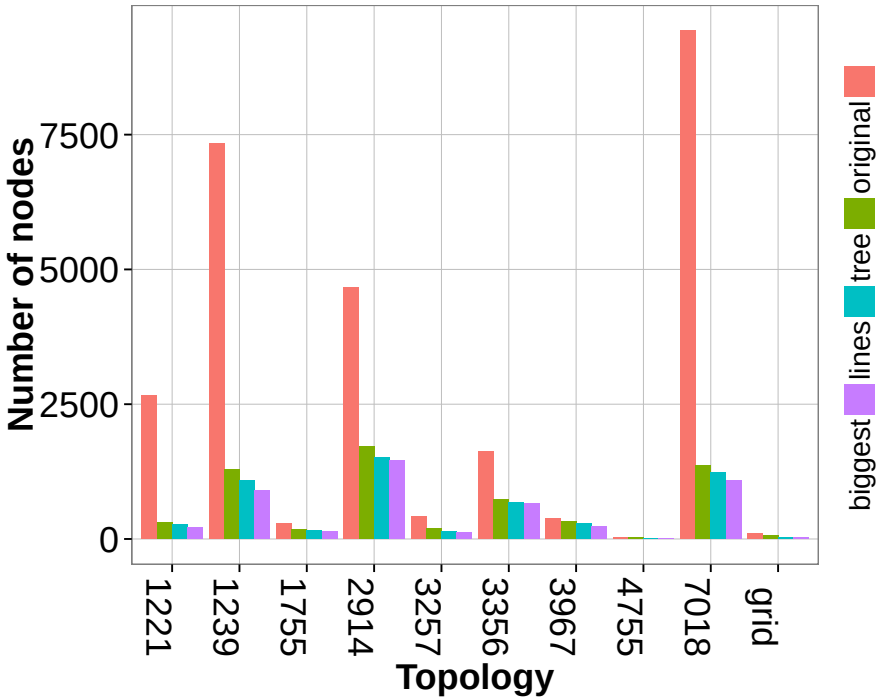


FIGURE: Composition of each topology: Tree, Relay and Motif nodes breakout

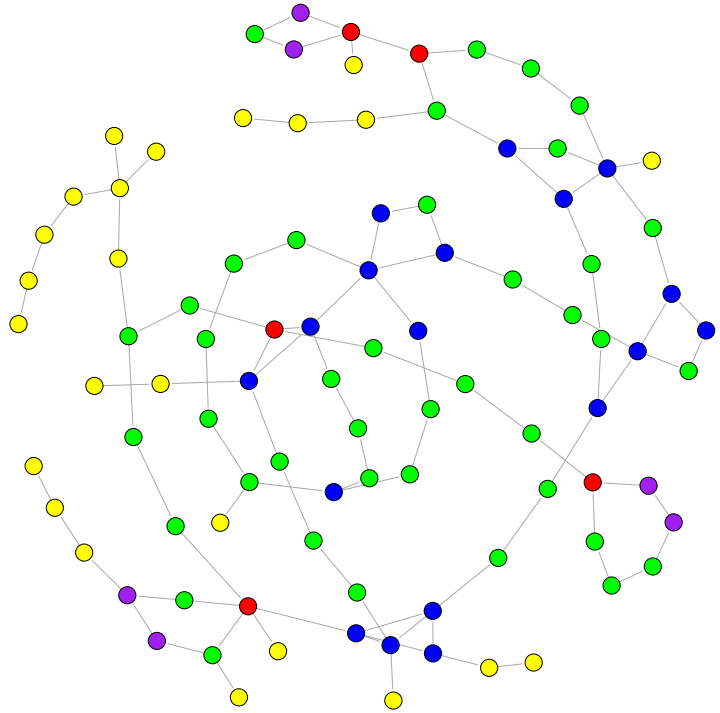


FIGURE: The « grid » topology

Results-2

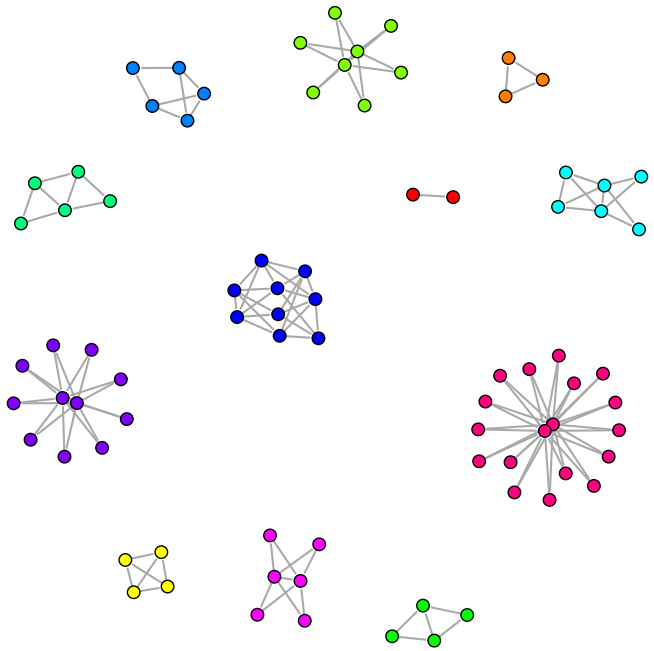


FIGURE: Common motifs found in the topologies

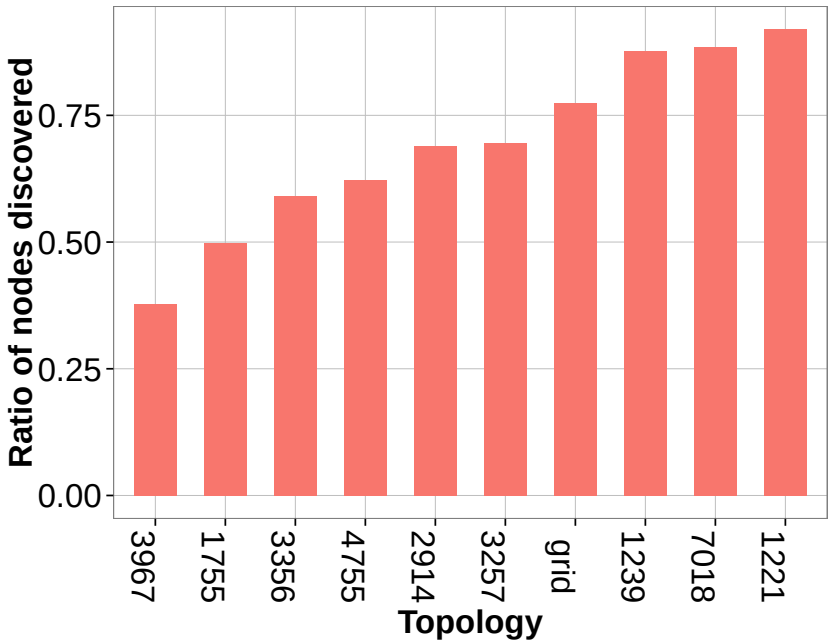


FIGURE: Ratio of the topology discoverable with the dictionnary

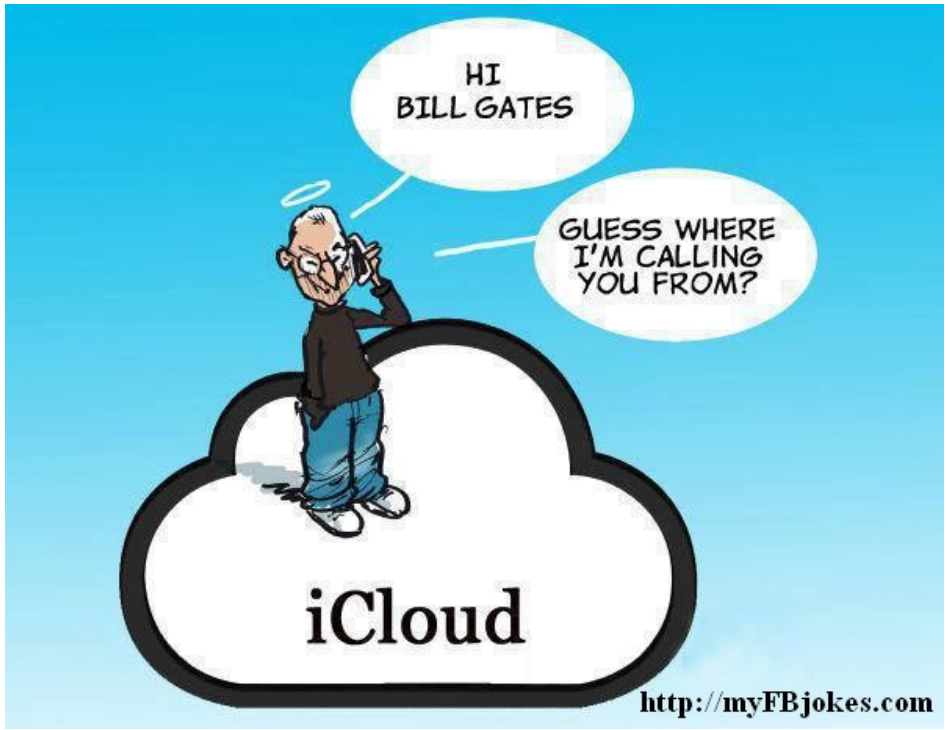
Conclusion

Can information leak through Vnet embedding answers: **Yes !**

- General case: slow
- Small dictionary ⇒fast leak !

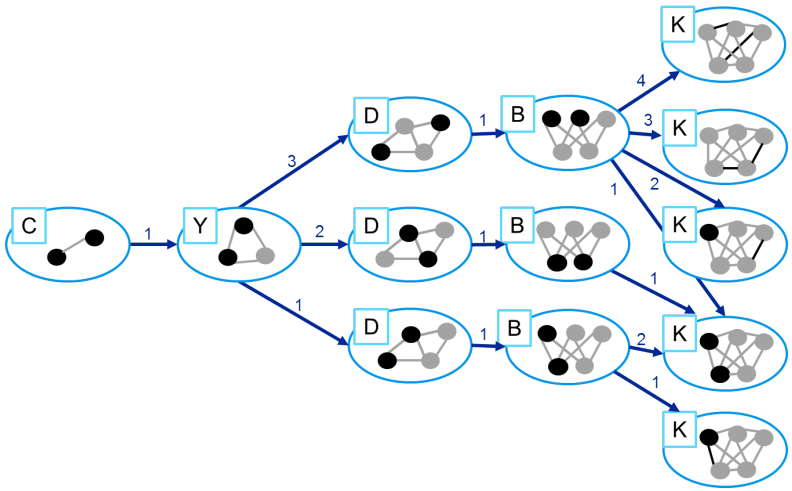
Directions

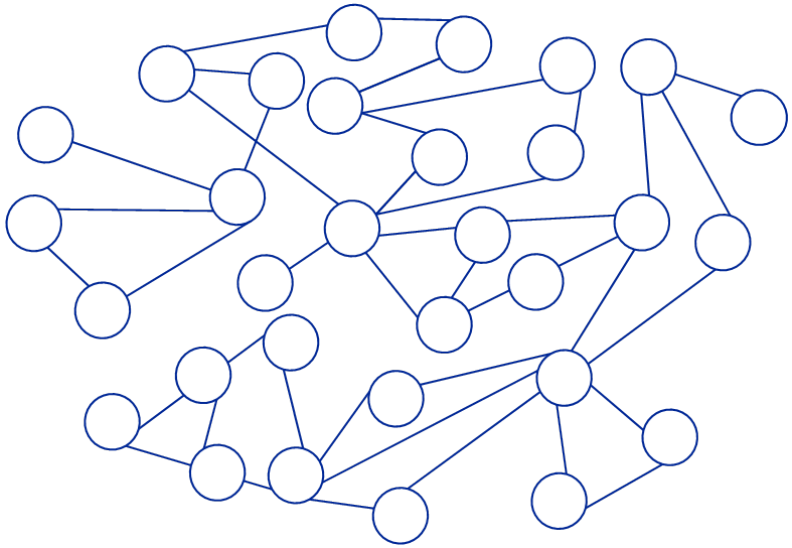
- (Outer)Planar graphs ?
- Only partial info:
  - min-cut
  - bounded stretch ?



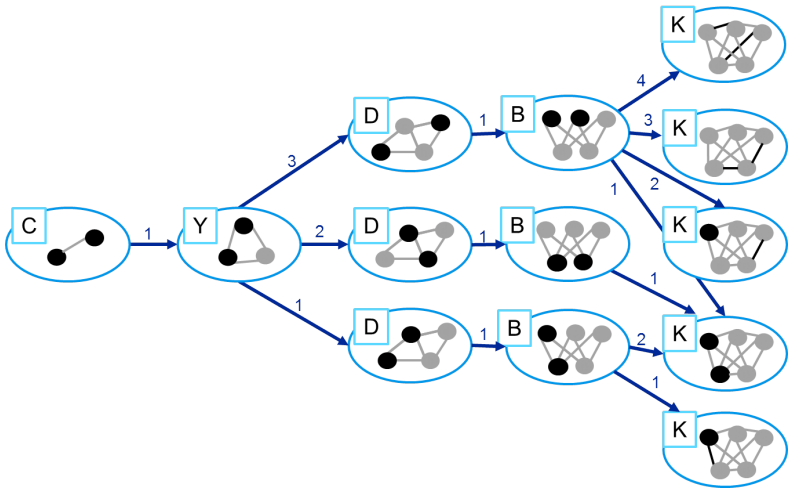
Thanks!

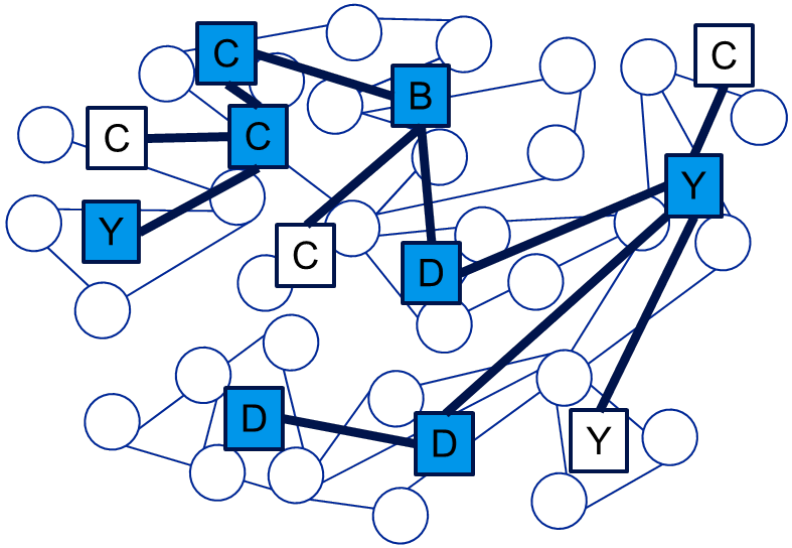
Observations





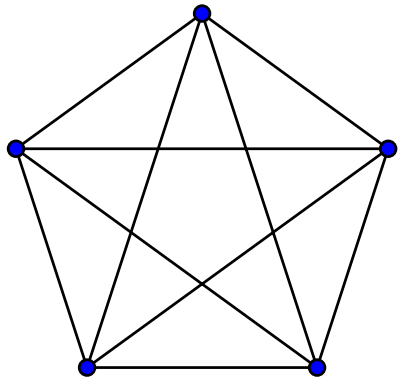
Observations

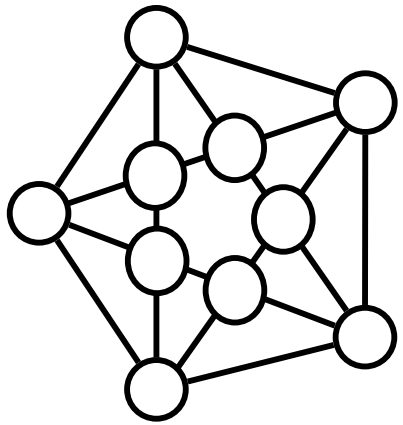




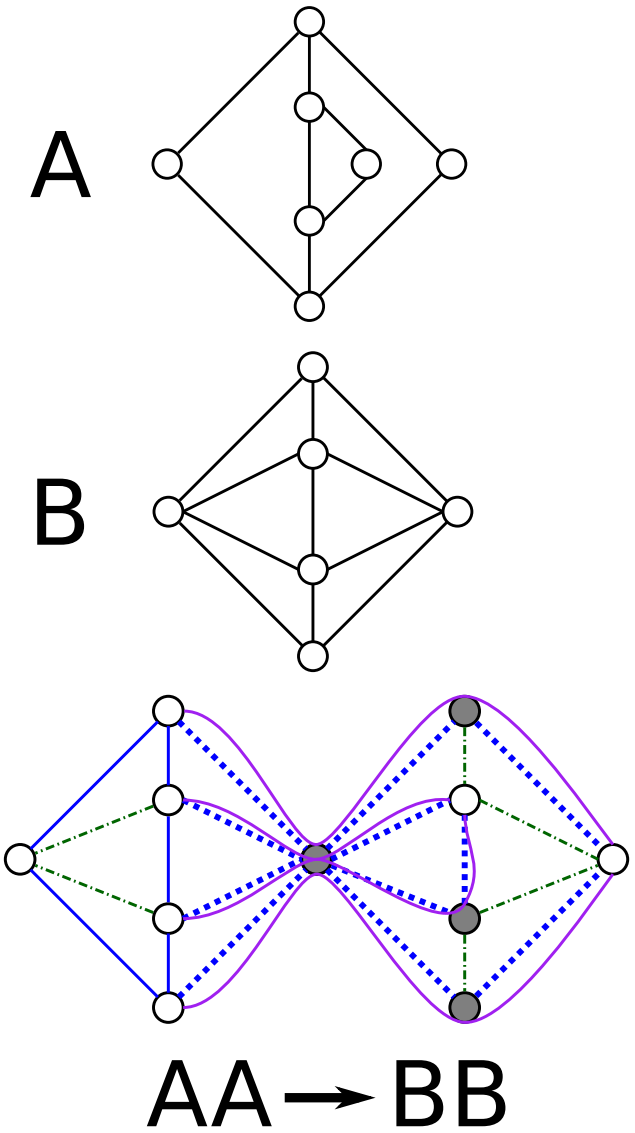
Observations

- Minor differences





Observations



Observations