Models and Algorithms for Dynamic Graphs

Mathilde Vernet
Supervised by: Yoann Pigné, Eric Sanlaville
Collaboration: Stefan Balev, Maciej Drozdowski, Frédéric Guinand
mathilde.vernet@univ-lehavre.fr

LITIS, team RI2C
Université Le Havre Normandie

LIP6 Seminar
Paris
October 22, 2020

Why dynamic graphs ?

- Time can be an important variable
- Static graphs not sufficient

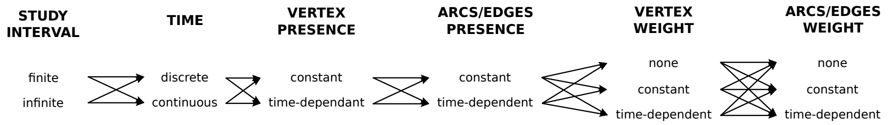
Various terminology

- temporal networks
- dynamic networks
- time varying graphs
- evolving graphs
- temporal graphs
- dynamic graphs
- link streams

Various application fields

- Transportation networks
  - *Roads temporarily unavailable*
- Communication networks
  - *Sensor networks*
- Social networks
  - *Evolving relationships*

Various models



Dynamic graph

- Succession of static graphs: $G = (G_i)_{i \in \mathcal{T}}$, where:
  - $\mathcal{T} = \{1, \ldots, T\}$ is the study interval
  - T is the time horizon
  - $G_i = (V, E_i)$ is a t-graph

Example

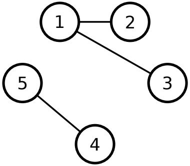


Figure: G_1

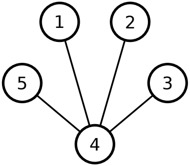


Figure: G_2

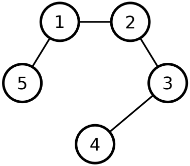


Figure: G_3

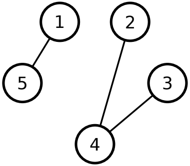
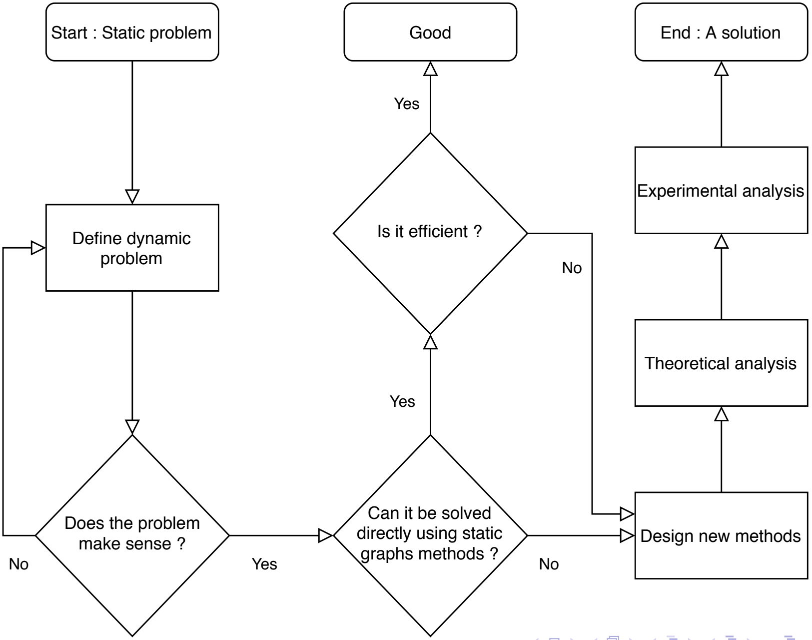


Figure: G_4


Start : Static problem
Good
End : A solution
Yes
Experimental analysis
Define dynamic problem
Is it efficient ?
No
Theoretical analysis
Yes
Does the problem make sense ?
Can it be solved directly using static graphs methods ?
Design new methods
No
Yes
No

Different solving methods

1. Solve T independent problems, one on each t-graph
2. The problem is equivalent to a problem on a static graph
3. A method specific to dynamic graphs is necessary

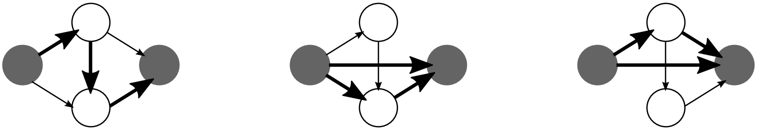
Different solving methods

1. Solve T independent problems, one on each t-graph
2. The problem is equivalent to a problem on a static graph
3. A method specific to dynamic graphs is necessary

Category 1

Example: maximum flow on dynamic graph without travel time or storage
Solution: Get maximum flow on each t-graph, sum them

Example



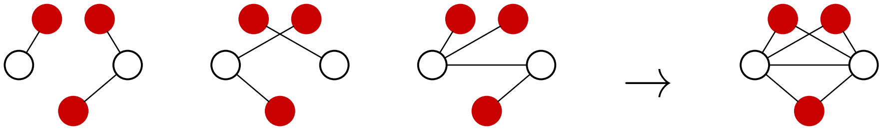
Different solving methods

1. Solve T independent problems, one on each t-graph
2. The problem is equivalent to a problem on a static graph
3. A method specific to dynamic graphs is necessary

Category 2

Example: Maximum independent set
Solution: Equivalent to maximum independent set on a static graph which is the union of the t-graphs

Example



Different solving methods

1. Solve T independent problems, one on each t-graph
2. The problem is equivalent to a problem on a static graph
3. **A method specific to dynamic graphs is necessary**

Category 1

Example: maximum flow on dynamic graph without travel time or storage
Solution: Get maximum flow on each t-graph, sum them

Category 2

Example: Maximum independent set
Solution: Equivalent to maximum independent set on a static graph which is the intersection of the t-graphs

Category 3

Example: Persistent connected components
Solution: Design new algorithm

Minimum cost flow:

- Dynamic graph: no storage, no travel time
- Minimum cost flow problem
- Look for efficient algorithm
- Avoid time-expanded graph

Maximum flow:

- Dynamic graph: infinite storage, no travel time
- Maximum flow problem
- Look for efficient algorithm
- Avoid time-expanded graph

Persistent Connected Components

- Connected component in a dynamic graph?
- How can it be identified ?

Steiner problem

- Steiner problem in a dynamic graph?
- How can it be identified ?

How to define connectivity in dynamic graphs?

Journey-based definitions

- There is a journey both ways between each pair of vertices in the component *(Bhadra and Ferreira 2003)*
- The journey has bounded length *(Gómez-Calzado et al. 2015)*
- Connected on any time window of given length *(Huyghues-Despointes, Bui-Xuan, and Magnien 2016)*

Without journeys

- Connection on intersection of graphs *(Casteigts et al. 2015)*
- Period of time on which the graph remains connected *(Akrida and Spirakis 2019)*

Persistent Connected Component (PCC)

Set K of vertices of size k that remain connected for l consecutive time steps

$p = (K, k, l)$

- K: set of vertices
- k: size of set K
- l: length (# time steps)

A PCC $p = (K, k, l)$ is considered maximal

- $\nexists\ p' = (K \cup \{u\}, k+1, l)$, vertex $u \notin K$, on the same time interval
- $\nexists\ p' = (K, k, l+1)$ on the same time interval

Differences to literature

Literature	$\neq$	*Us*
Mostly based on journeys	$\neq$	Instantaneous and lasting connexion
OR		*AND*
Connexion through same path	$\neq$	Connexion through different paths
OR		*AND*
Continuous time	$\neq$	Discrete time

$p_1 = (\{1, 5\}, 2, 3)$

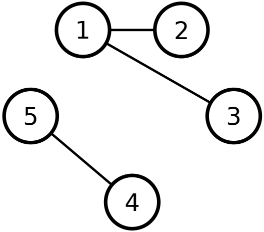


Figure: G_1

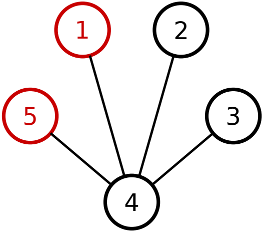


Figure: G_2

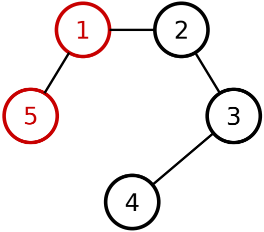


Figure: G_3

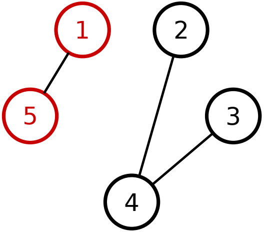


Figure: G_4

p_2=({4, 5}, 2, 3)

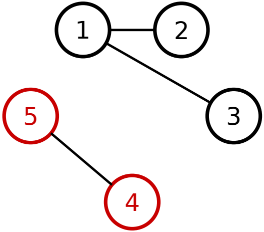


Figure: G_1

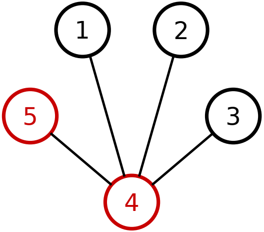


Figure: G_2

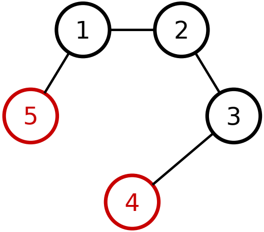


Figure: G_3

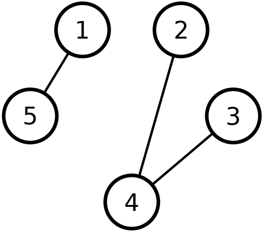


Figure: G_4

p_3=({1, 2, 3}, 3, 3)

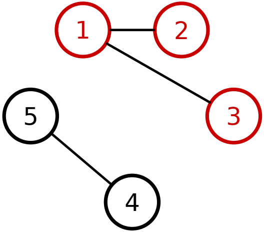


Figure: G_1

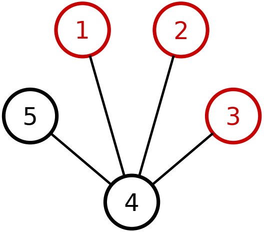


Figure: G_2

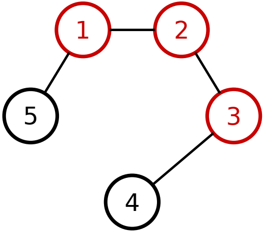


Figure: G_3

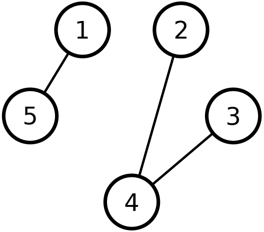


Figure: G_4

p_4=({2, 3, 4}, 3, 3)

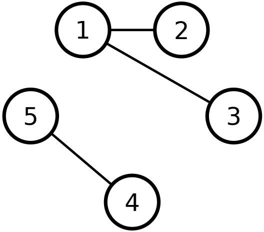


Figure: G_1

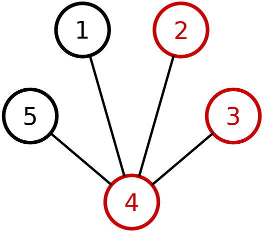


Figure: G_2

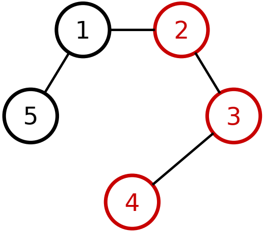


Figure: G_3

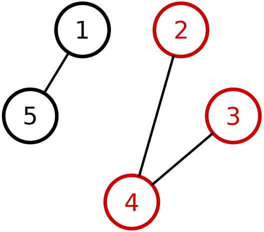


Figure: G_4

$p_5 = (\{1, 2, 3, 4, 5\}, 5, 2)$

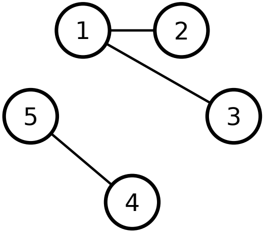


Figure: G_1

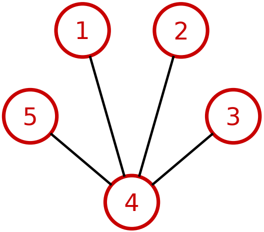


Figure: G_2

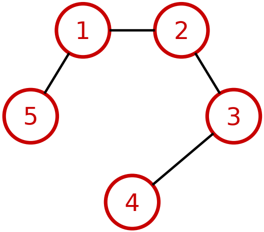


Figure: G_3

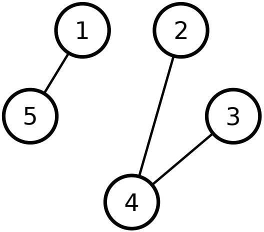


Figure: G_4

$p_6 = (\{2, 3\}, 2, 4)$

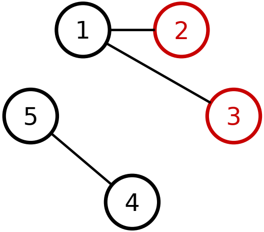


Figure: G_1

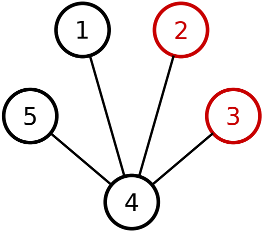


Figure: G_2

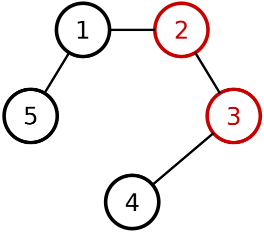


Figure: G_3

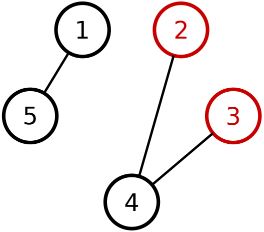


Figure: G_4

Dominant PCC

$p = (K, k, l)$ is dominant $\Leftrightarrow \forall\ p' = (K', k', l') \neq p$:

- $k > k'; l \geq l'$

OR

- $l > l'; k \geq k'$

Goal

- Retrieve dominant PCCs
- Keep one PCC for a given size k and length l

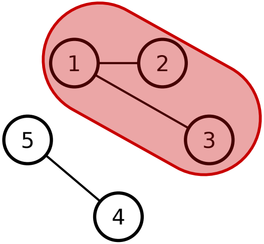


Figure: G_1

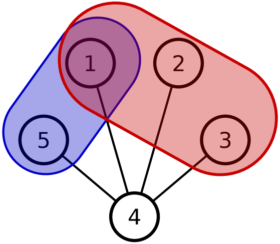


Figure: G_2

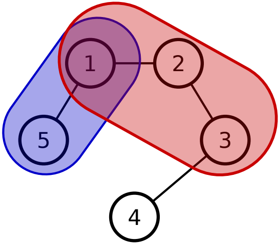


Figure: G_3

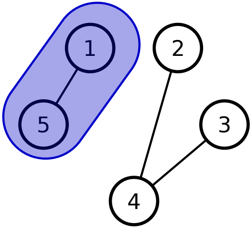


Figure: G_4

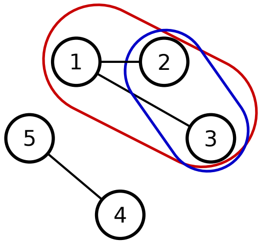


Figure: G_1

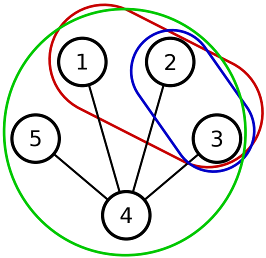


Figure: G_2

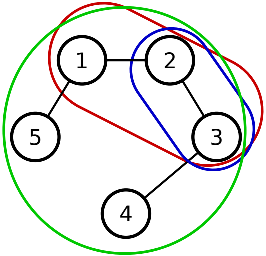


Figure: G_3

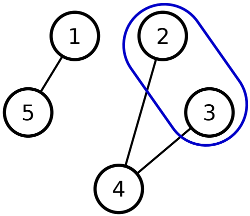


Figure: G_4

PersIstent Connected CompoNent InCremental Algorithm (PICCNIC)

- At each time step $t \in \mathcal{T}$:
  1. Compute classical connected components in G_t
  2. Compare with PCC alive at $t-1$
  3. Extract finished PCC and still alive ones
  4. Remove dominated PCCs

Remarks on PICCNIC

- Incremental Algorithm
- Exact at each time step
- Polynomial complexity

Complexity

- One iteration: $O(n^2)$ (number of PCCs kept from $t-1$ to t is bounded by n)
- T iterations
- **Total complexity:** $O(n^2 \cdot T)$

Why bounded number of PCCs kept?

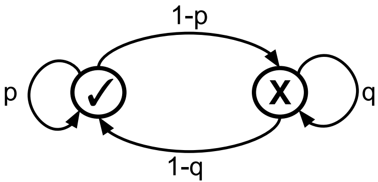
- $p = (K, k, l)$ and $p' = (K', k', l')$ two PCCs
- $K \cap K' \neq \emptyset$ and $K \nsubseteq K'$ at time step t
  $\Rightarrow K \cup K'$ is a connected component in G_t
  $\Rightarrow$ The union is considered and kept
  $\Rightarrow$ PCCs kept are disjoint or included
  $\Rightarrow$ No more than n PCCs are kept

Parameters

- 4 underlying graph types:
  - Random
  - Regular
  - Scale-free
  - Random Geometric
- Average degree 4, 8, 12
- Varying n from 100 to 4500
- Fixed T to 1000
- 10 instances

Dynamicity

- Markov chain for each edge presence



Complexity

- Algorithm complexity: $O(n^2 \cdot T)$
- With fixed T: $O(n^2)$

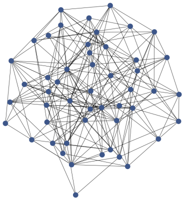


Figure: Random Graphs

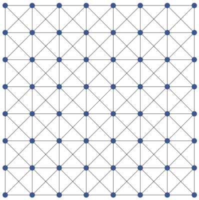


Figure: Regular Graphs

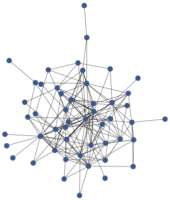


Figure: Scale-free Graphs

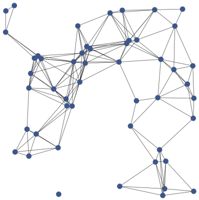


Figure: Random Geometric Graphs

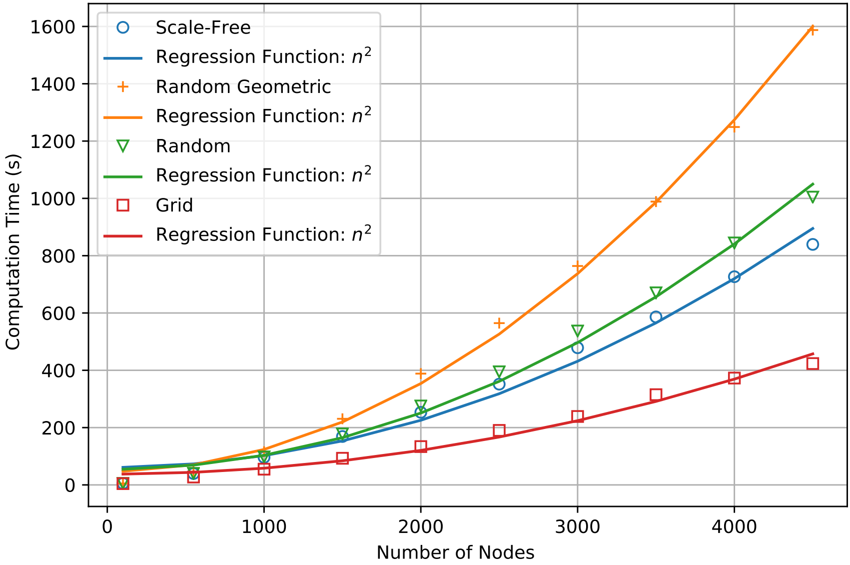


Figure: Computation Time (Average presence probability 90%, average degree 4)

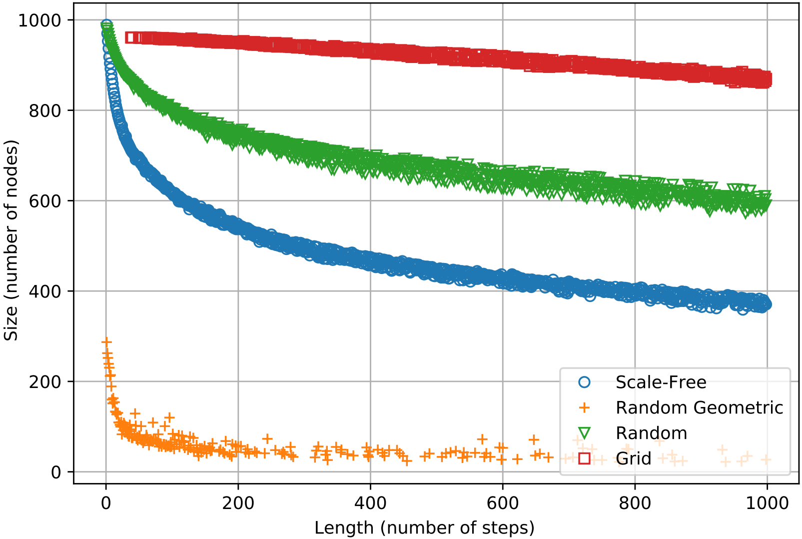


Figure: Representation of non-dominated PCCs size and length (Average presence probability 90%, average degree 4)

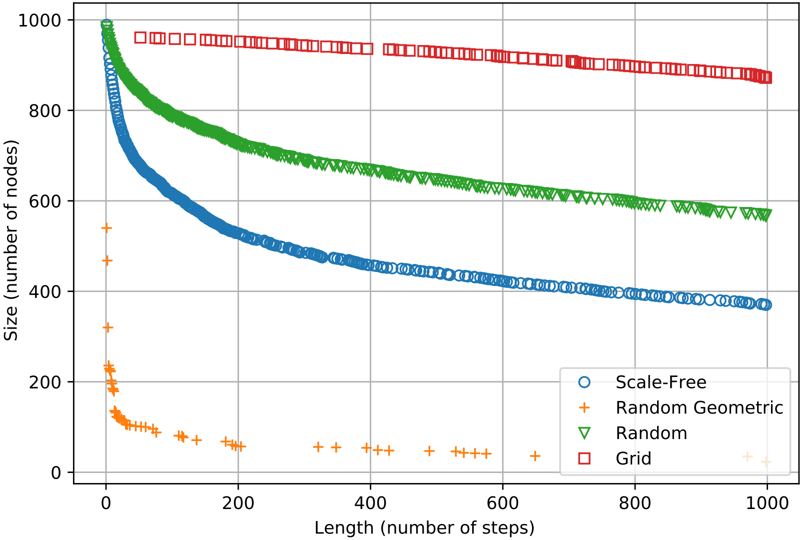


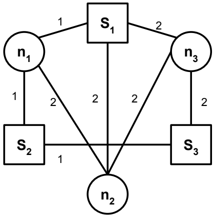
Figure: Pareto front of non-dominated PCCs for one instance (Average presence probability 90%, average degree 4)

Context

- (Static) graph $G = (V, E)$
- Edge weight $w_{(i,j)} \geqslant 0 \ \forall (i,j) \in E$
- Terminal set $S \subset V$

Goal

- Find a tree with minimum weight containing all vertices from S

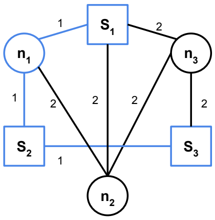


Context

- (Static) graph $G = (V, E)$
- Edge weight $w_{(i,j)} \geqslant 0 \ \forall (i,j) \in E$
- Terminal set $S \subset V$

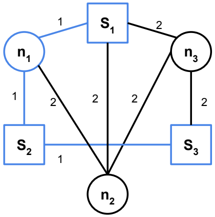
Goal

- Find a tree with minimum weight containing all vertices from S



Context

- (Static) graph $G = (V, E)$
- Edge weight $w_{(i,j)} \geqslant 0 \ \forall (i, j) \in E$
- Terminal set $S \subset V$



Goal

- Find a tree with minimum weight containing all vertices from S

Decision problem

- Is there a subgraph of G containing all vertices from S with total weight lower to K?

Complexity proof

- NP-complete
- Polynomial transformation from *exact cover by 3-sets*

Context

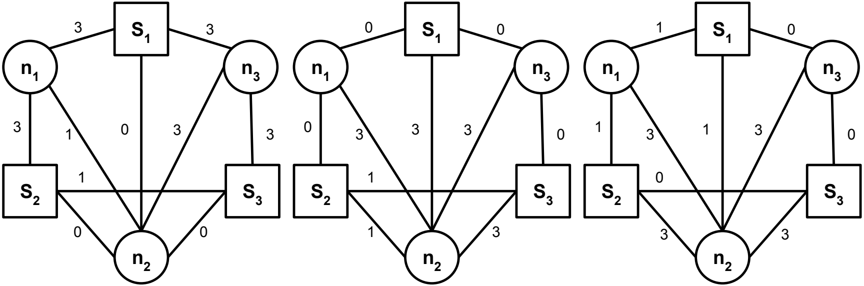
- Dynamic graph: $G = (V, E)$
- Study interval of G: $\mathcal{T} = \{1, \ldots, T\}$
- Edges have time-dependent weight: $w_{(i,j),t} \geqslant 0 \ \forall (i,j) \in E, t \in \mathcal{T}$
- Terminal set $S \subset V$

Questions

- What is a Steiner Tree in a dynamic graph ?
  - A "dynamic tree" containing all vertices of S with minimum total weight on $\mathcal{T}$
- How is that tree ?
- Can special cases be identified ?

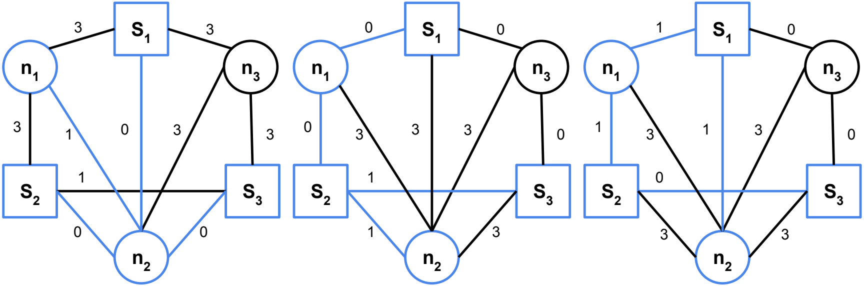
Possibility 1 : Fully Connected Set

- Find $V' \subset V$ such that $S \subset V'$
- Find spanning tree of V' of minimum weight



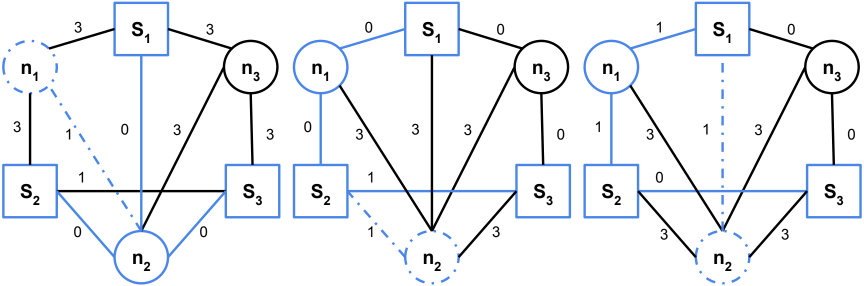
Possibility 1 : Fully Connected Set

- Find $V' \subset V$ such that $S \subset V'$
- Find spanning tree of V' of minimum weight



Possibility 1 : Fully Connected Set

- Find $V' \subset V$ such that $S \subset V'$
- Find spanning tree of V' of minimum weight

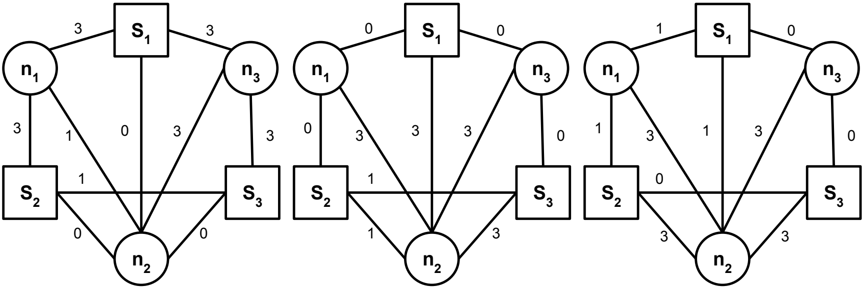


Problem

- As long as the terminals are connected, what is the point of connecting V'?

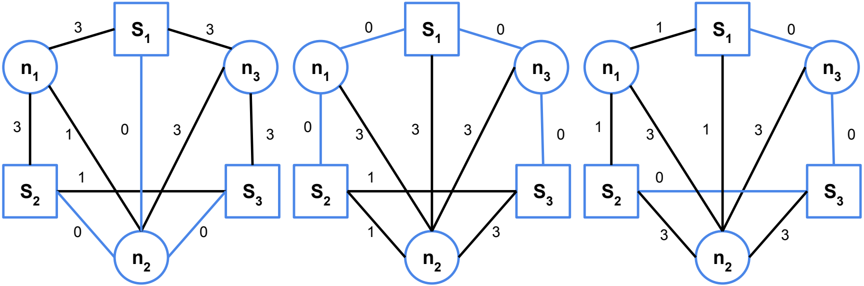
Possibility 2: Partially connected Set

- Find $V' \subset V$ such that $S \subset V'$
- Minimize the weight of edges connecting S



Possibility 2: Partially connected Set

- Find $V' \subset V$ such that $S \subset V'$
- Minimize the weight of edges connecting S



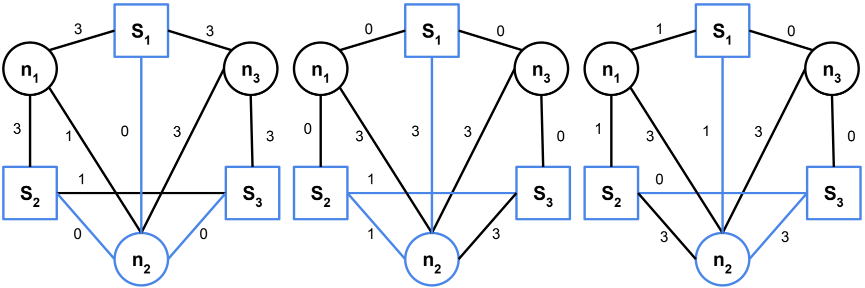
Problem

- Take $V' = V$ and look for Steiner tree at each time step ignoring vertices from $V' \setminus S$

Goal

Find V' with $S \subset V' \subset V$ and $E'_t \subset E_t \ \forall t \leqslant T$ such that

- All vertices of S in same connected component in $G'_t = (V', E'_t)$
- Cardinality of V' is minimum
- $\sum_e \sum_t w_{e'_t}$ with $e'_t \in E'_t$



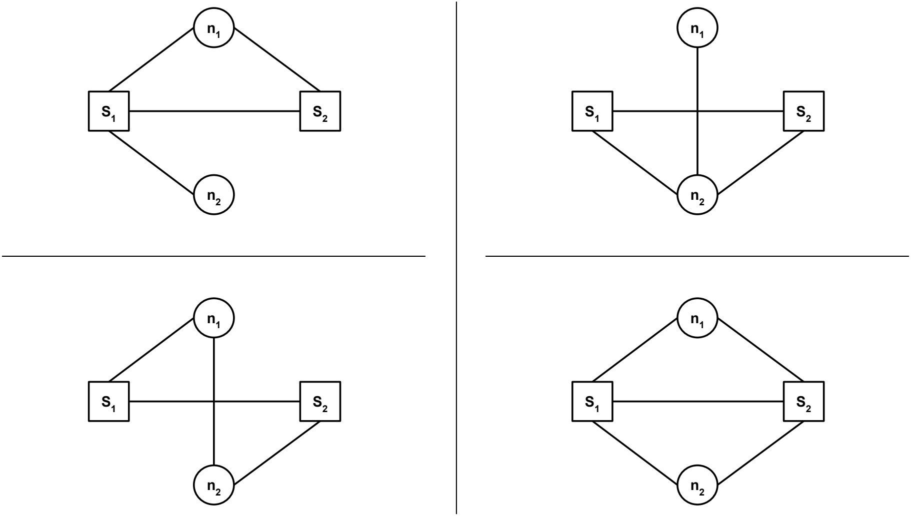
Definition

- No weight on edges
- $|S| = 2$: Connect optimally two vertices
- Minimize number of vertices keeping the terminals connected

Remarks

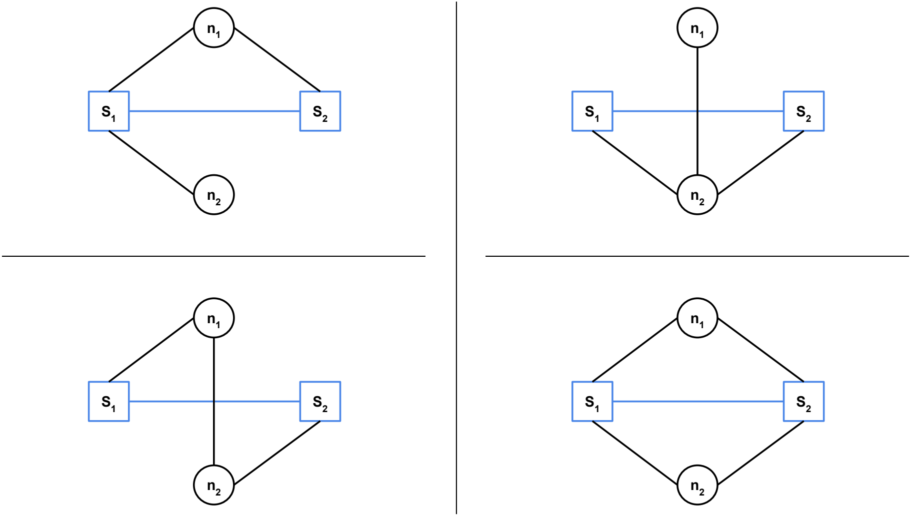
- Polynomial in static graphs
- NP-complete on dynamic graphs

Example 1

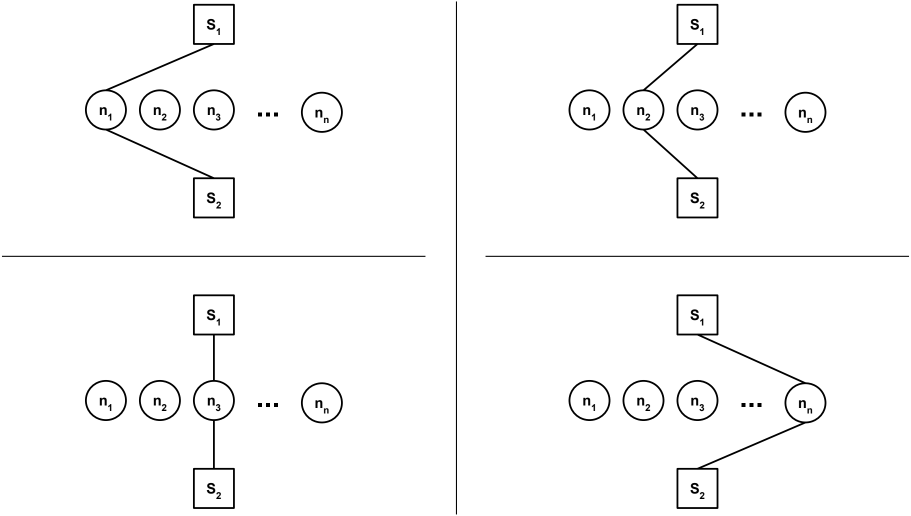


Example 1

No extra vertex is necessary

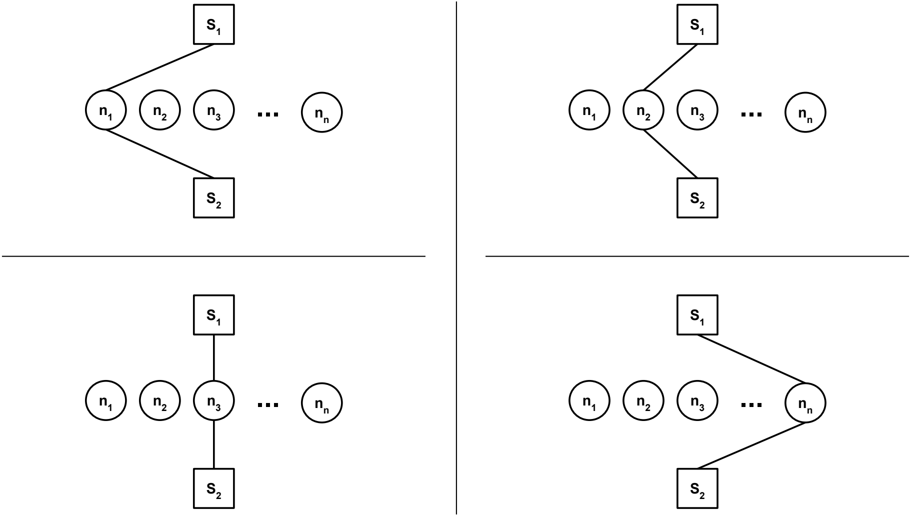


Example 2

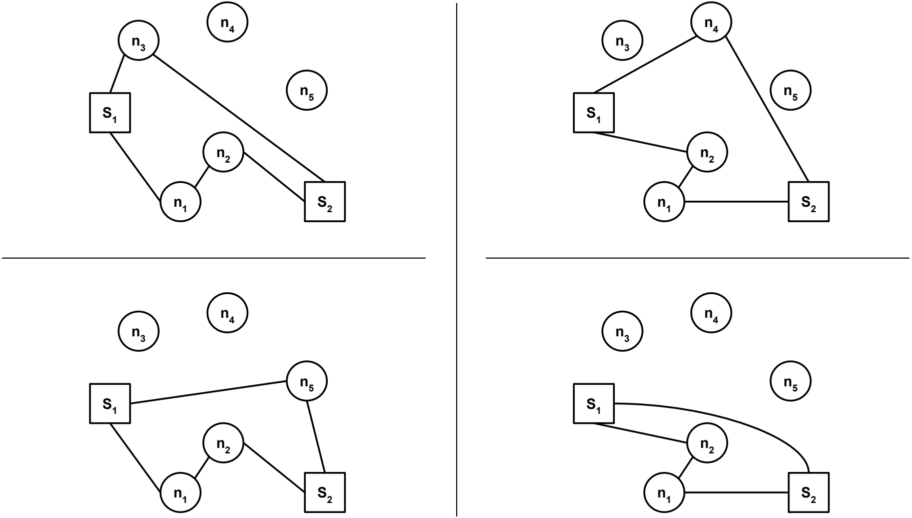


Example 2

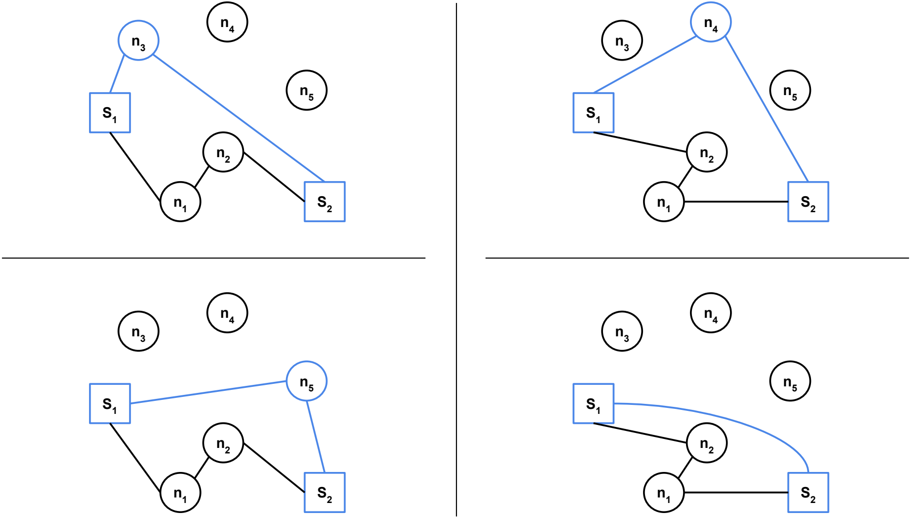
All vertices of the graph are necessary



Example 3

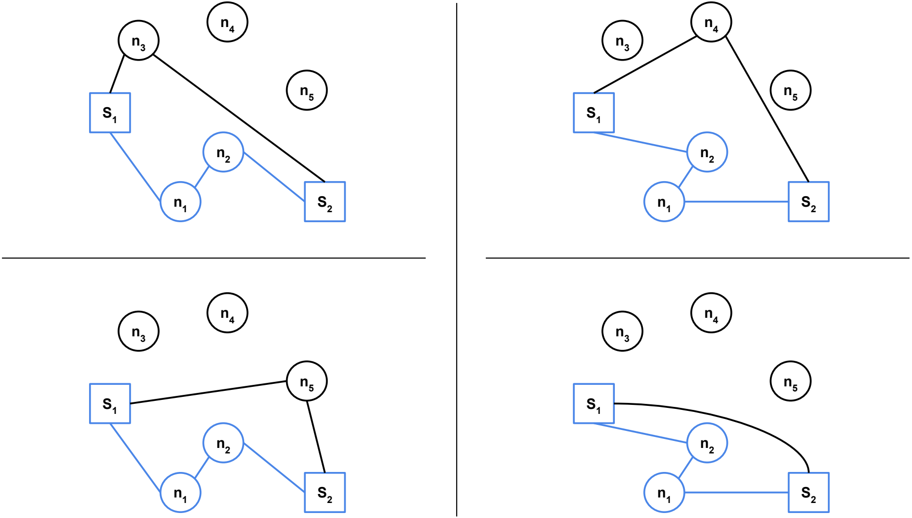

n4
n3
n5
S1
n2
n1
S2
n4
n3
n5
S1
n2
n1
S2
n4
n3
n5
S1
n2
n1
S2
n4
n3
n5
S1
n2
n1
S2

Example 3


n4
n3
n5
s1
n2
n1
s2
n4
n3
n5
s1
n2
n1
s2
n4
n3
n5
s1
n2
n1
s2
n4
n3
n5
s1
n2
n1
s2

Example 3

The shortest path is not a good idea



NP-completeness proof

- Polynomial transformation from the Vertex Cover Problem

Reminder: Vertex Cover

- Graph $G = (V, E)$
- Vertex Cover Set $V_c \subset V$ such that $\forall (u, v) \in E$, $u \in V_c$ or $v \in V_c$

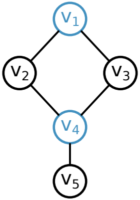
For a given integer $k \geq 0$, is there a set V_c of size k ?

Transformation

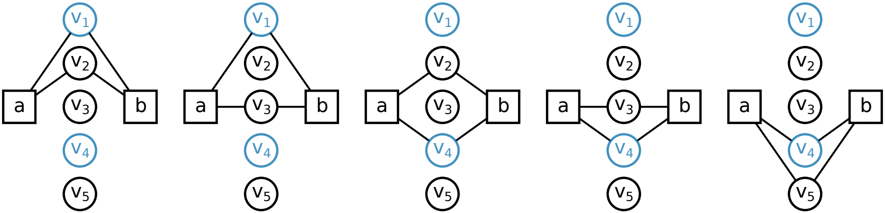
- $\forall u \in V$, there is a vertex u in the dynamic graph G^{DYN}
- G^{DYN} has two extra vertices a and b
- $\forall e = (u, v) \in E$, there is a time step i_e in G^{DYN} and $G^{DYN}_{i_e}$ has 4 edges; (a, u), (u, b), (a, v), (v, b)

Example of transformation

Vertex Cover instance:



Corresponding instance on dynamic graph:



Minimum cost flow:

- Time expanded graph not used
- Polynomial optimal algorithm
- Theoretically and practically efficient
- *Vernet et al. 2020* in DAM

Persistent Connected Components

- Extension of connected components to dynamic graphs
- Polynomial online and optimal algorithm to obtain dominant PCCs
- Definition of PCC extended to:
  - Directed graphs, eternal PCC, interrupted components
- Under review *(Vernet, Pigné, and Sanlaville 2020)*

Maximum flow:

- Look for new method without time expanded graph
- Bound on maximum flow value

Steiner problem

- Extension of Steiner Problem to dynamic graphs
- Special case proven to be NP-complete
- Working paper *(Balev et al. 2020)*

General questions

- Formal problem classification
  - from problem definition point of view
  - from the algorithmic point of view

Steiner problem

- Exact algorithms efficient in specific cases
- Approximation algorithms

Connected components

- Persistent connected components
  - Enumerative algorithm for maximal PCCs
- Eternal connected components
  - Experimental analysis of the algorithm

Maximum flow

- How tight is our bound ?

Thank you for your attention

Mathilde Vernet
Supervised by: Yoann Pigné, Eric Sanlaville
Collaboration: Stefan Balev, Maciej Drozdowski, Frédéric Guinand
`mathilde.vernet@univ-lehavre.fr`

LITIS, team RI2C
Université Le Havre Normandie

LIP6 Seminar
Paris
October 22, 2020

- Akrida, Eleni C and Paul G Spirakis (2019). "On Verifying and Maintaining Connectivity of Interval Temporal Networks". In: *Parallel Processing Letters* 29.02, p. 1950009.
- Balev, Stefan et al. (2020). "On the complexity of the Dynamic Steiner Tree Problem". Rapport interne.
- Bhadra, Sandeep and Afonso Ferreira (2003). "Complexity of connected components in evolving graphs and the computation of multicast trees in dynamic networks". In: *International Conference on Ad-Hoc Networks and Wireless*. Springer, pp. 259–270.
- Casteigts, Arnaud et al. (2015). "Efficiently Testing T-Interval Connectivity in Dynamic Graphs". In: *International Conference on Algorithms and Complexity*. Springer, pp. 89–100.
- Gómez-Calzado, Carlos et al. (2015). "A connectivity model for agreement in dynamic systems". In: *European Conference on Parallel Processing*. Springer, pp. 333–345.

Huyghues-Despointes, Charles, Binh-Minh Bui-Xuan, and Clémence Magnien (2016). “Forte Δ-connexité dans les flots de liens”. In: *ALGOTEL 2016-18èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*.

Vernet, Mathilde, Yoann Pigné, and Eric Sanlaville (2020). “A Study of Connectivity on Dynamic Graphs: Computing Persistent Connected Components”. Article soumis. URL: https://hal.archives-ouvertes.fr/hal-02473325.

Vernet, Mathilde et al. (2020). “A theoretical and experimental study of a new algorithm for minimum cost flow in dynamic graphs”. In: *Discrete Applied Mathematics*.