ISIMA
INSTITUT D'INFORMATIQUE
Université Clermont Auvergne

SYNTHETIC GRAPHS GENERATOR

Presented by Eloise Yollande Molé Kamga

yollandemole@gmail.com

Supervisor: Pr. Engelbert Mephu

University Clermont Auvergne

20 june 2018

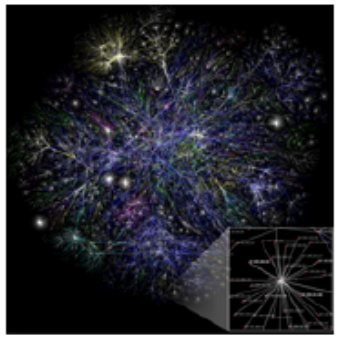
Summary

Summary

Introduction

- Graphs can be used to represent various objects and relationship between them.
- Several algorithms have been created to analyze these large graphs.
- So we need large real graphs to test these algorithms.
- **It is difficult to represent a real network in its entirety[10].**
- Hence the implementation of real synthetic large graph generator.

Réseau social



Réseau internet

Réseau de communication

Réseau urbain

Questions

Q1

How to design a real graph?

Q2

How to design a real static and synthetic graph generator?

Summary

Real graph properties

- Most graphs in nature although of different origins have common characteristics:
  - **high density**: the average degree does not depend on the size of the graph;
  - **small diameter**, usually in the order of the **logarithm of the number of vertices**;
  - a degree distribution in **power law**[7][2] (we refer to graph without scale).

Static graph generator approaches

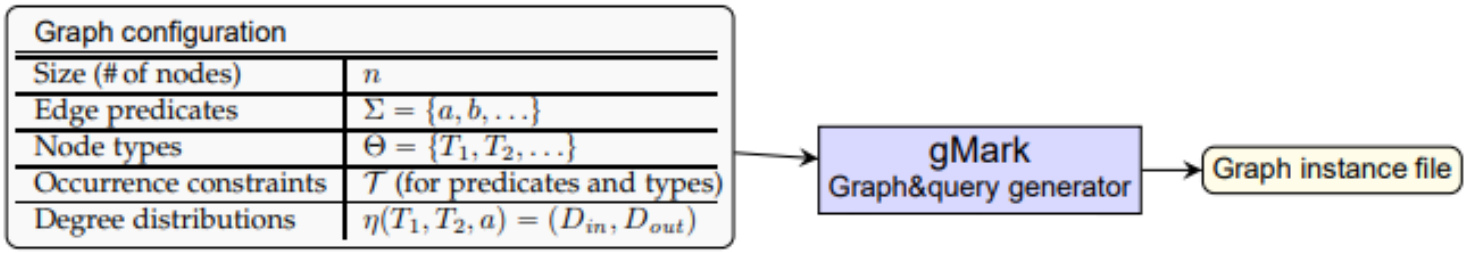
- **Random**: in this model the edges are placed at random on all possible edges.
  - Erdos and Rényi model [8].
- **Configuration model** [5]: it is based on an initial configuration set of the graph.
  - Gmark [3].
- **Models that begin from a known graph.**
  - Preferential attachment method: new entrants prefer to attach themselves to the most connected existing ones.
    - Barabasi and Albert model [4].
  - Construction of a regular graph: with a probability p one disconnects an end of a link to reconnect it to a chosen node, one examines the graph by varying P from 0 to 1.
    - Watts and strogatz model [13] - small-word graph.

Focus

Focus

The generation model that interests us here is configuration model. Gmark is an example of this model.

Gmark: schema-driven static graph generator



- **Approach**
  1. For each triplet of distribution degree put in input:
     - create for edge *a*, D_{out} node(s) of each type element T_1 called *src*;
     - create for edge *a*, D_{in} node(s) of each type element T_2 called *dest*;
     - connect the nodes *src* and *dest*($Min(|src|, |dest|)$ times): $(src, a, dest)$.

Questions

Q3

The time dimension is often forgotten when representing graphs, hence the concept of evolving graphs. How to build a scalable graph generator?

Q4

What are the properties that can define the temporality of graphs?

Summary

Types of evolution

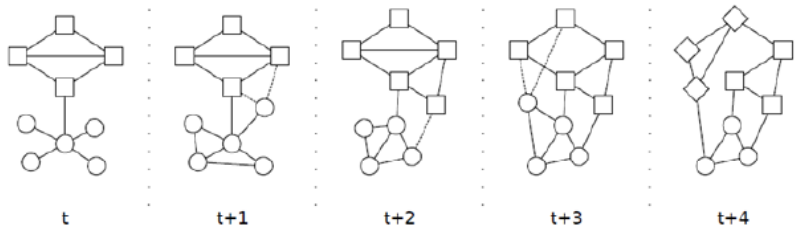
Definition

Evolving graph
Let $G = (V, E)$ a graph directed with V nodes set and E edge set whose extremities are in V. Let $S_G = \{G_1, G_2, ..., G_T\}$, a set of G subgraphs.
The system $S = (G, S_G)$ is called "evolving graph".

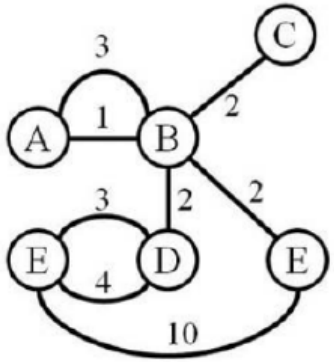
- Types of evolution in evolving graph:
  - add or remove nodes or edges;
  - updating labels.
- Evolution dynamic in step form.

Representation of evolving graphs

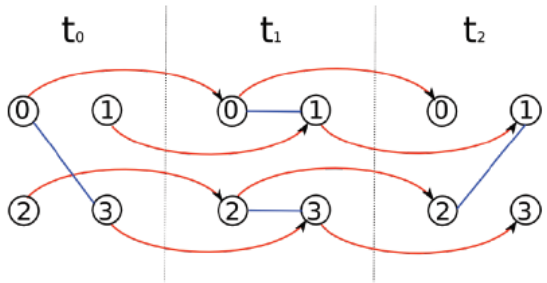
1. Snapshots. [11]



2. Labelled form. [9][6]



3. Connected snapshots set. [14]



Evolving graph metrics

- According to Yoann Pigné's thesis on the modeling and decentralized processing of dynamic graphs, we can quote evolving graphs as properties:
  - **age**: difference between the current date and the date of last appearance;
  - **cumulative age**: total durations of the time intervals during which it is present in the graph;
  - **volatility**: ratio between the number of appearances and the cumulative age. Gives a qualitative indication of the life of the observed element;
  - **volatility of pattern**: average value of the volatility of the elements of pattern;
  - **average age**: inverse of the volatility;
  - **refresh rate**: ratio between the number of changes in the structure and the number of elements in the original structure.

Question

Q5

What are the evolving graphs generator approaches?

List of evolving graphs generator approaches

- **Random**: is to randomly generate a set of snapshot.
- **Incremental approach**: is to deduct the snapshot from the time t_{i+1} using snapshot t_i.
  - e.g EGG[1].
- **Non-incremental approach**: consists in analyzing the evolution of a network and building the corresponding evoling graph.

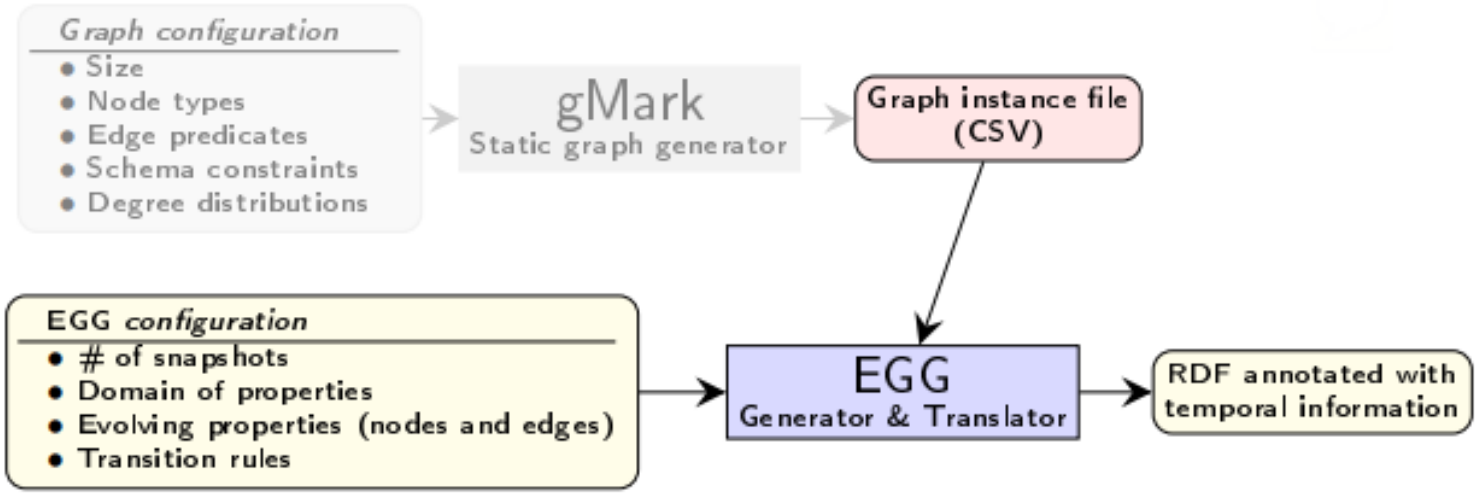
  Here the snapshot of time t_i can be obtained without knowing the snapshot of time t_{i-1}.
  - e.g Evogen[12].
- **Is there a hybrid approach?** Which combine non-incremental and incremental approach depend of periods.

Example of evolving graphs generation algorithms

Example

We will present EGG and Evogen.

Evolving graph generator(EGG): inputs



- **EGG[1] Configuration**: $\delta = (S, I, \sigma, map, \phi)$.
  - I: # of periods we want to generate.
  - σ : finite set of dynamic properties.
  - *map*: function that maps edge predicates and node types to their properties.
  - ϕ : defines for each property, its domain, its evolution parameters, and its influence on other properties.

EGG: evolving properties examples

- Validity of the elements.
- Property values.
- Evolution of a property.
- Relationship between this property and other properties.

```
    "weather":{
            "elements_type":"city",
            "domain":{
                    "type":"qualitative",
                    "order":"false",
                    "v":"true",
                    "values":["sunny","rainy","cloudy"],
                    "distribution":{"type":"uniform"}
            },
            "duration":1,
            "evolution":{
                    "e":"true",
                    "relation":"true",
                    "staticity":0.5,
                    "successors":{"sunny":["sunny","cloudy"]}
```

```
        "rules":[],
        "rulese":[
                {"if":{"prop":"weather","change":["cloudy","sunny"]},
                    then":{"prop":"qAir","sens":"up"}},
                {"if":{"prop":"weather","change":["rainy","sunny"]},"
                    then":{"prop":"qAir","sens":"up"}},
                {"if":{"prop":"weather","change":["sunny","cloudy"]},
                    then":{"prop":"qAir","sens":"down"}}
        ]
}
```

EGG: approach

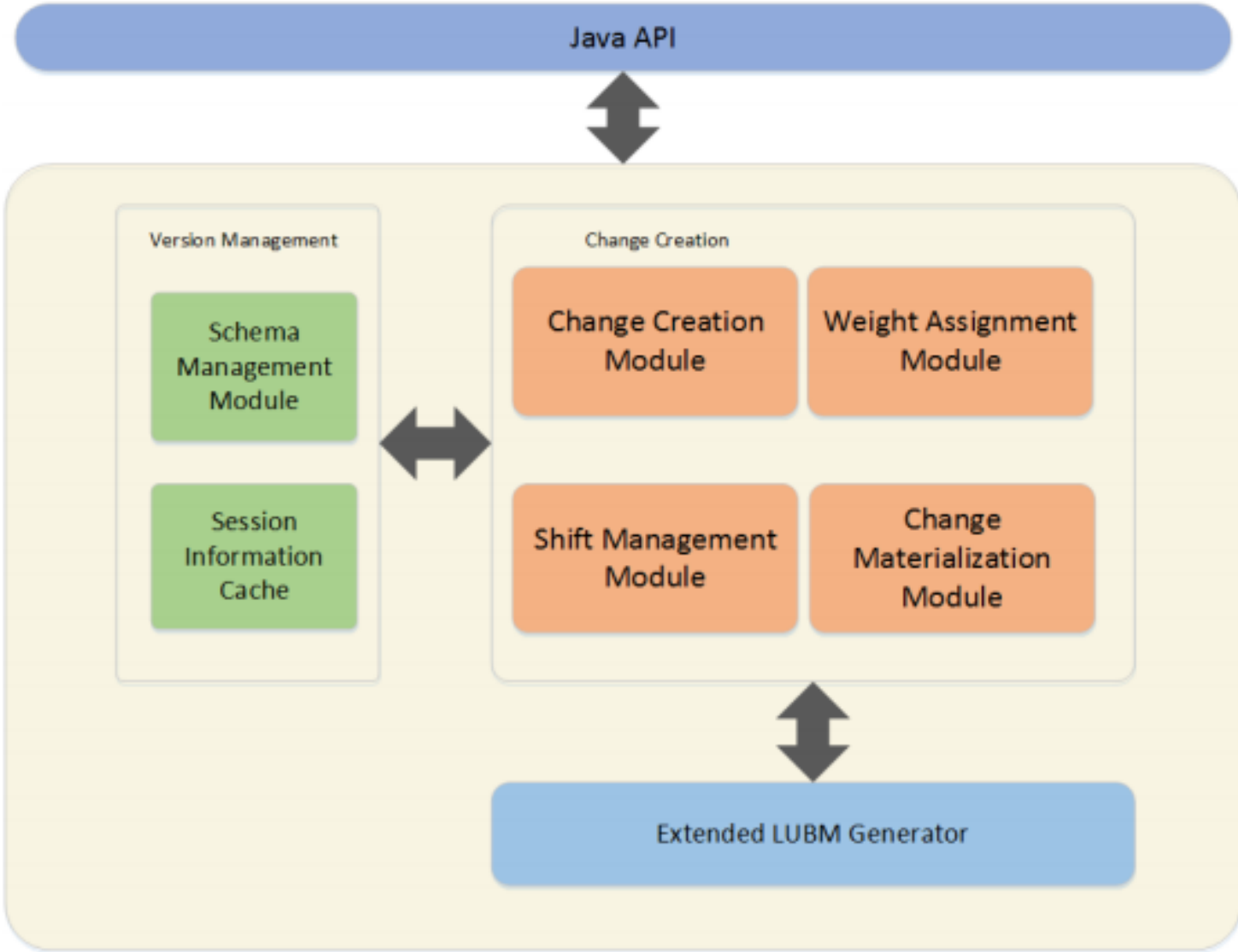
- **Approach**
  1. Generate a static graph SG through gMark.
  2. Sort topologically properties dependencies.
  3. Generate the first valid graph $G0$.(Subgraph of SG).
  4. Generate evolving properties for $G0$.
  5. For each snapshot in $(1, n)$.
     - Generate the valid graph G_i.
     - Generate the evolving properties for G_i.
- **Outputs**: set of snapshots in RDF graph form annotated by temporal information.

Evogen

1. Evogen [12] is a generator of RDF graph versioned.
2. Inputs:
   - **number of versions**: number of snapshots;
   - **shift**: captures the modification of its size through different versions;
   - **monotonicity**: captures whether or not a dataset with an incremental or decremental shift changes monotonically in a given time period.
3. Approach:

   multiply the global rate of evolution (shift) by the identifier of each version to obtain the size of the graph.
   - **Change creation module**.
   - **Version management module**.

Evogen architecture



Questions

Q6

The evolving graphs produced by EGG respect the input properties but they do not respect the shape of the application domain because depending on the domain they must have a certain shape. What are the forms of graphs by domain?

Q7

How to represent the shape of each application domain in our evolving graphs?

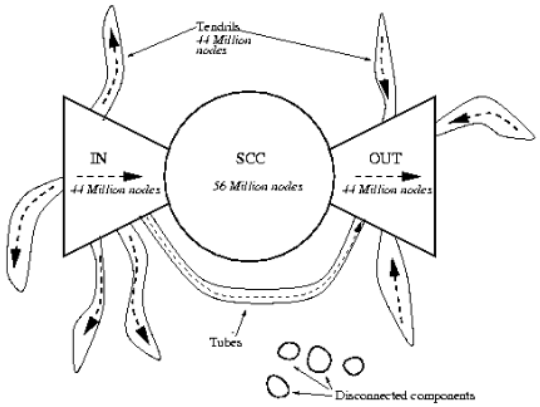
Summary

Network shape type according to the domain

- **Social network**: triangle shape.

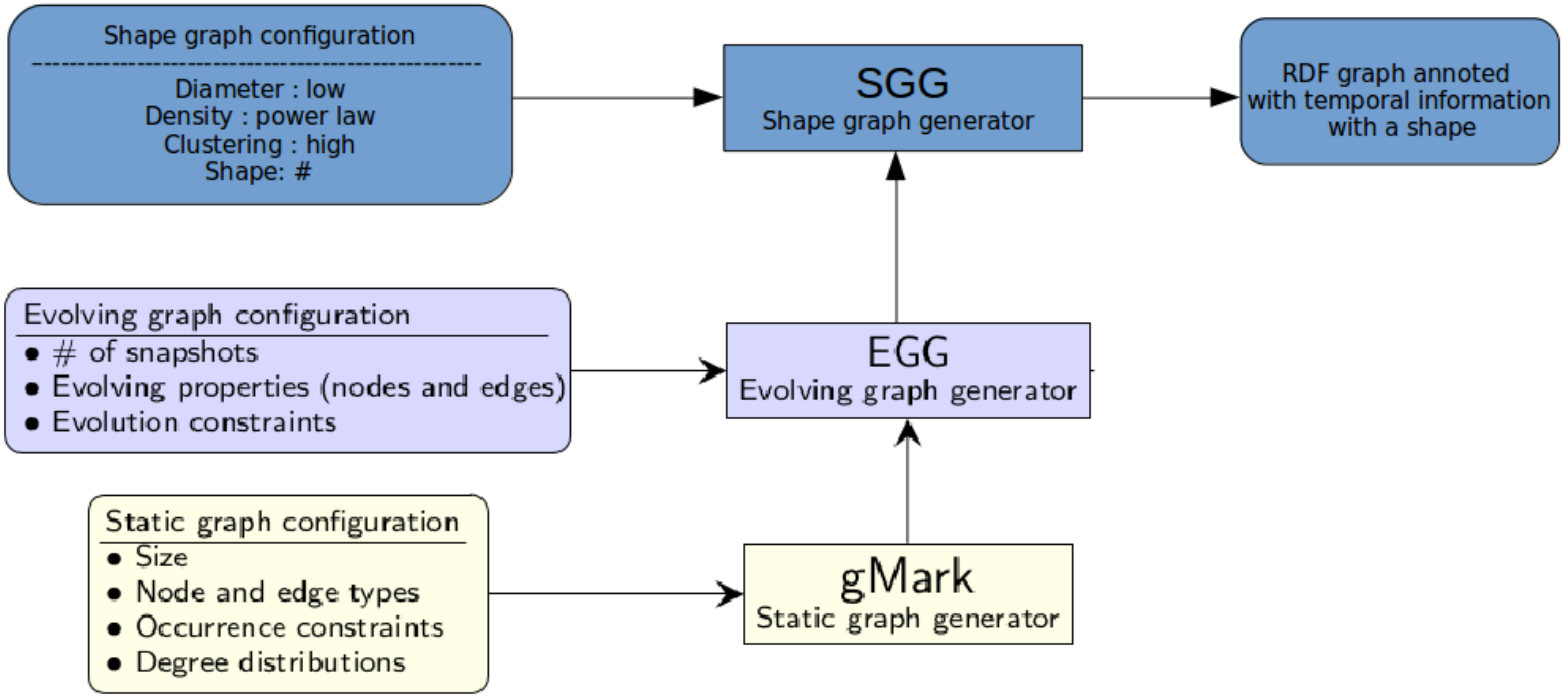
  In a real social network my friend's friend is my friend.

- **World Wide Web network**: bow tie shape.



- **Urban network**: rectangle shape.

Problem-solving approaches



- To assign shapes to evolving graphs we have identified three solutions tracks:
  - generate the shape only at the first snapshot and used incremental approach for deduct others snapshots;
  - after each snapshot is generated, add the corresponding shape;
  - build the shape during the generation process (during graphs construction).

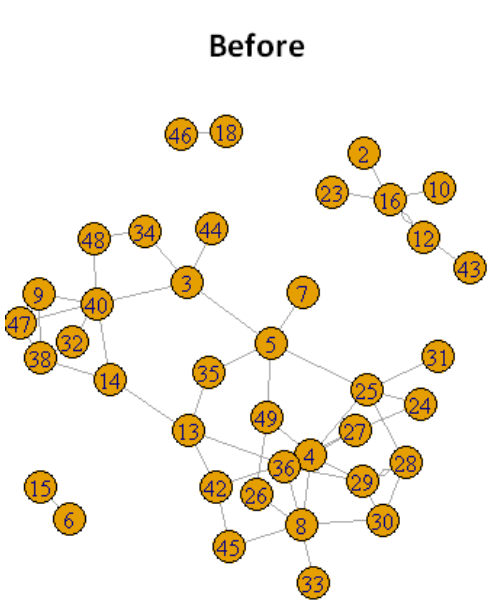
Algorithm that gives the triangle shape

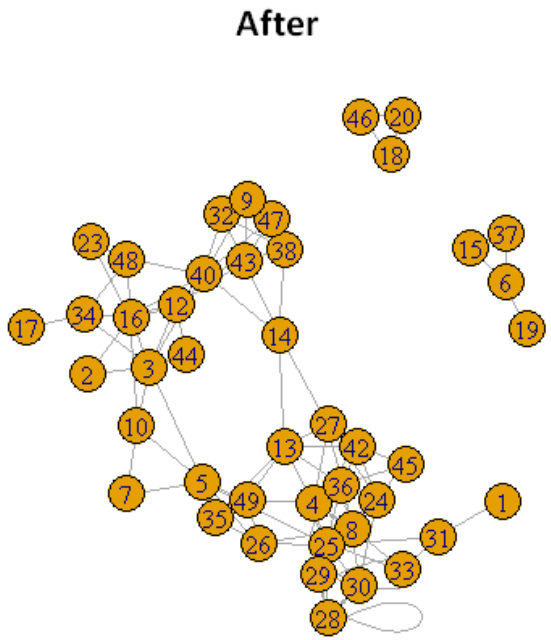
- Tools used for our experimentation: **python** and **R**.
- **Approach**

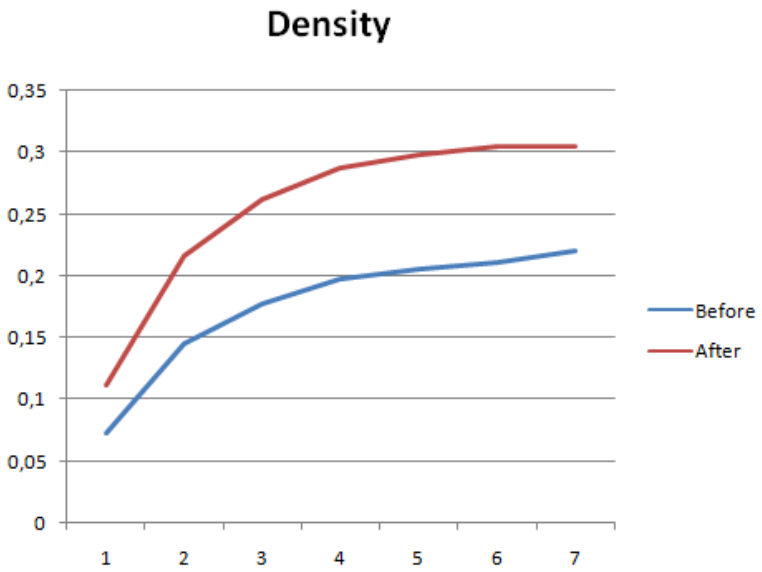
```
def Triangle(graph_src,graph_dest):
  graph_add_src=[]
  graph_add_dest=[]
  i = 0
  while i < len(graph_src):
    n = 0
    while len(graph_dest) > n:
        if graph_src[i] == graph_dest[n]:
            if graph_dest[i] != graph_src[n]:
                list.append(graph_add_src,graph_src[n])
                list.append(graph_add_dest,graph_dest[i])
        n = n+1
    i = i+1
  resul = numpy.array([graph_add_src,graph_add_dest])
  return resul
```

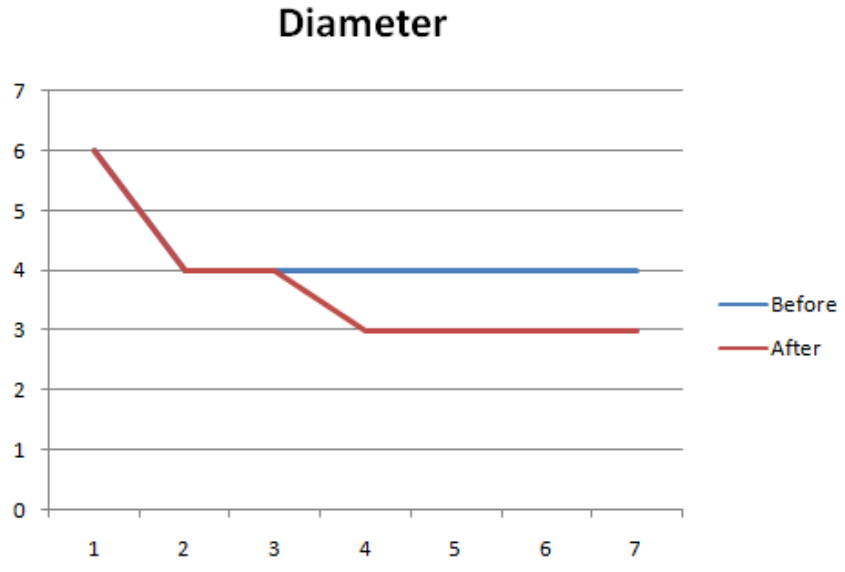
- We evaluated diameter and density.

Results before and after taking into account the shape









Summary

Conclusions and future works

- Conclusions

  When generating graphs it is necessary to:

  - take into account the temporal dimension;
  - take into account the shape of the graphs of the domain you are in.

- Future works:
  - eliminate loops and evaluate execution time for large graphs.
  - find a way to stay in shape over time.
  - finalize implementation of interface and comparative study of metrics.
  - implement the shape for the others domains.
  - to study some community detection and frequent subgraphs extraction algorithms in evolving graphs.

- Links
  - https://github.com/graphMark/gmark
  - https://github.com/karimalami7/EGG

Bibliography I

Karim Alami, Radu Ciucanu, and Mephu Nguifo Engelbert.

EGG: A Framework for Generating Evolving RDF Graphs.

In *ISWC 2017 (International Semantic Web Conference), system demo*, Vienna, Austria, October 2017.

L. A. Amaral, A. Scala, M. Barthelemy, and H. E. Stanley.

Classes of small-world networks.

Proc Natl Acad Sci U S A, 97(21):11149–11152, October 2000.

Guillaume Bagan, Angela Bonifati, Radu Ciucanu, George Fletcher, Aurélien Lemay, and Nicky Advokaat.

gMark: Schema-Driven Generation of Graphs and Queries.

IEEE Transactions on Knowledge and Data Engineering, November 2016.

Bibliography II

Albert-Laszlo Barabasi and Reka Albert.

Emergence of scaling in random networks.

Science, 286(5439):509–512, 1999.

Edward A Bender and E.Rodney Canfield.

The asymptotic number of labeled graphs with given degree sequences.

Journal of Combinatorial Theory, Series A, 24(3):296 – 307, 1978.

Binh-Minh Bui-Xuan, Afonso Ferreira, and Aubin Jarry.

Computing shortest, fastest, and foremost journeys in dynamic networks.

Technical Report RR-4589, INRIA, October 2002.

Bibliography III

Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman.

Power-law distributions in empirical data.

SIAM Rev., 51(4):661–703, November 2009.

P. Erdős and A Rényi.

On the evolution of random graphs.

In *PUBLICATION OF THE MATHEMATICAL INSTITUTE OF THE HUNGARIAN ACADEMY OF SCIENCES*, pages 17–61, 1960.

Afonso Ferreira.

On models and algorithms for dynamic communication networks: The case for evolving graphs.

01 2002.

Bibliography IV

Jean-Loup Guillaume, Matthieu Latapy, and Damien Magoni.

Relevance of Massively Distributed Explorations of the Internet Topology: Qualitative Results.

Computer Networks, 50(16):3197–3224, 2006.

Wei Liu, Andrey Kan, Jeffrey Chan, James Bailey, Christopher Leckie, Jian Pei, and Ramamohanarao Kotagiri.

On compressing weighted time-evolving graphs.

In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 2319–2322, New York, NY, USA, 2012. ACM.

Bibliography V

Marios Meimaris.

Evogen: a generator for synthetic versioned RDF.

In *Proceedings of the Workshops of the EDBT/ICDT 2016 Joint Conference, France, March 15.*, 2016.

Duncan J. Watts and Steven H. Strogatz.

Collective dynamics of 'small-world' networks.

Nature, 393(6684):440–442, June 1998.

Klaus Wehmuth, Artur Ziviani, and Eric Fleury.

A unifying model for representing time-varying graphs.

CoRR, abs/1402.3488, 2014.

Thank you for your attention!