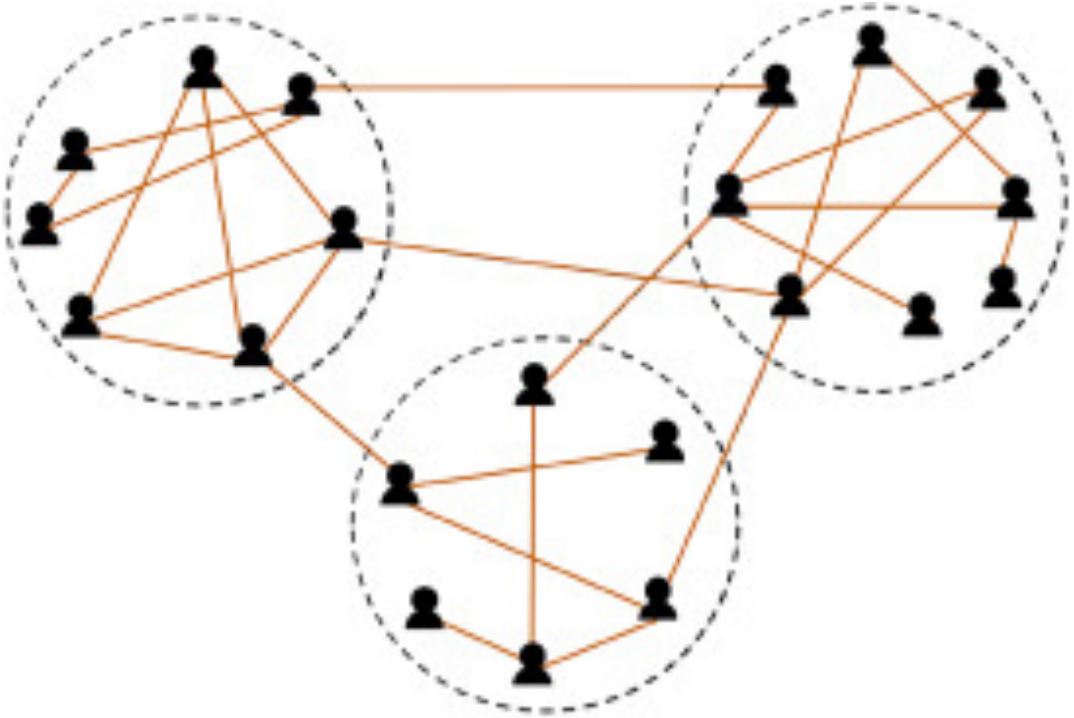
Community detection in signed graphs using label propagation

Sara DÍAZ
Ricardo ROJAS

Motivation

Community detection allows us to **better study datasets as it provides an insight into characteristics** shared by different subgroups that compose them.

Although much work has been done on community detection, **the specific domain of signed graphs has been less unexplored**, which presents an opportunity as these have become more common mainly due to the **importance that social sharing networks and other similar systems have garnered.**



Problem definition

Weighted signed graphs

A signed weighted signed network is defined as

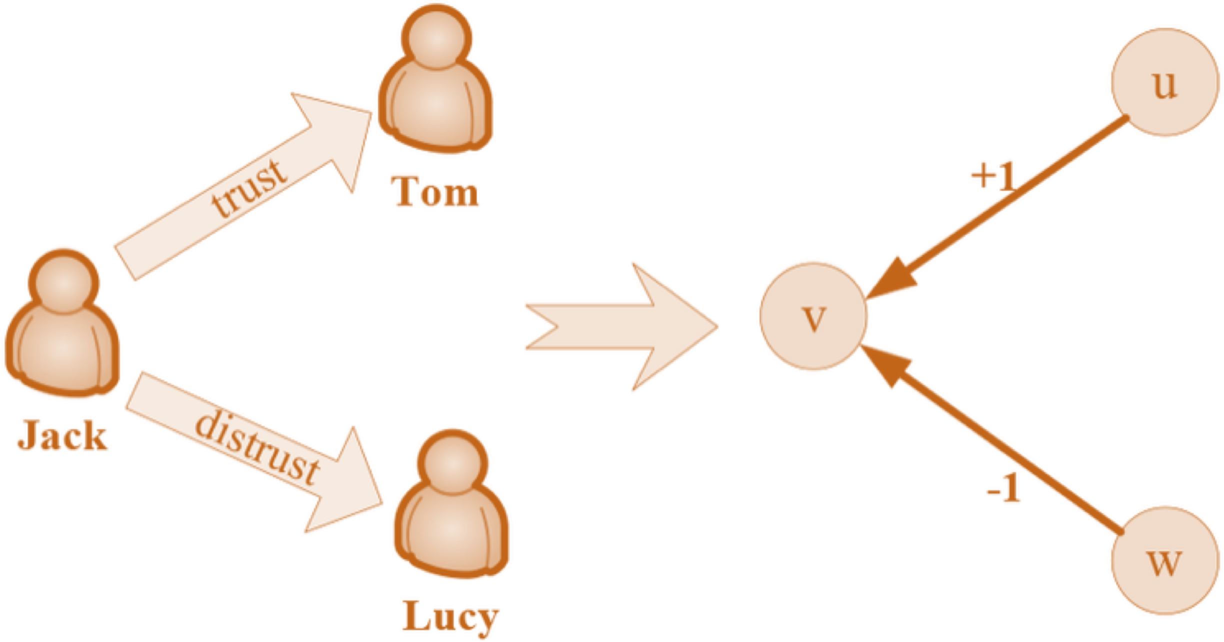
$SN = (V, E^{+}, E^{-})$

where,

$V = v_1, \dots, v_n$ is the set of vertices

$if\ a_{i,j} \in \mathbb{R}$ where (i, j) $\in$ V are entries in the adjacency matrix A:

$E^{+} \implies a_{i,j} > 0$
$E^{-} \implies a_{i,j} < 0$



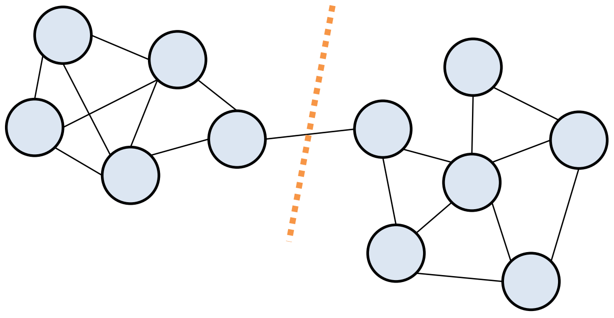
Community detection (1/2)

- Communities are groups of vertices having higher probability of being connected to each other than to members of other groups [Fortunato].

- The original algorithm [Newman] makes **cuts on edges with highest betweenness** *g(e)* of pair of pairs of nodes *s*, *t* as the number of the shortest paths (σ_{st}) that go through an edge *e*.

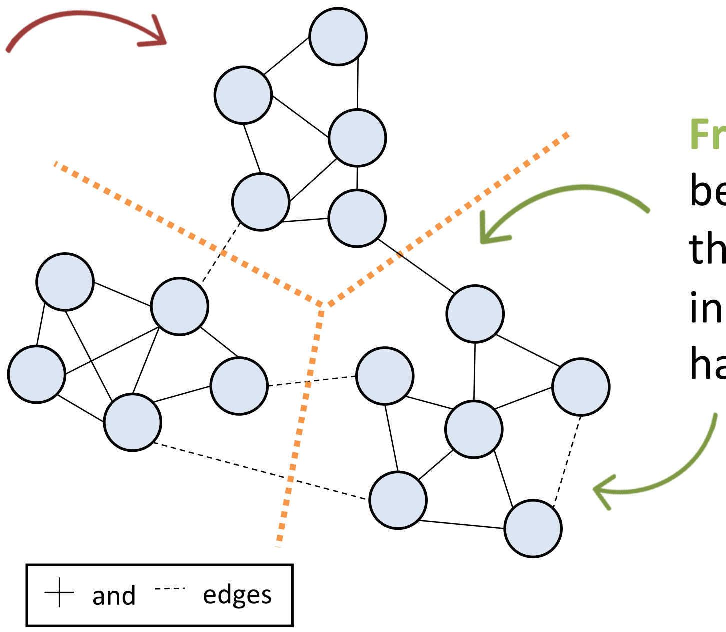
$$g(e) = \sum_{s,t \neq e} \left(\frac{\sigma_{st}(e)}{\sigma_{st}} \right)$$

σ_{st} is the total number of shortest paths from node s to node t and $\sigma_{st}(e)$ are those which pass through e



Community detection (2/2)

Modularity finds communities based on the principle that the inner edge **density** of a community is higher than **between communities**



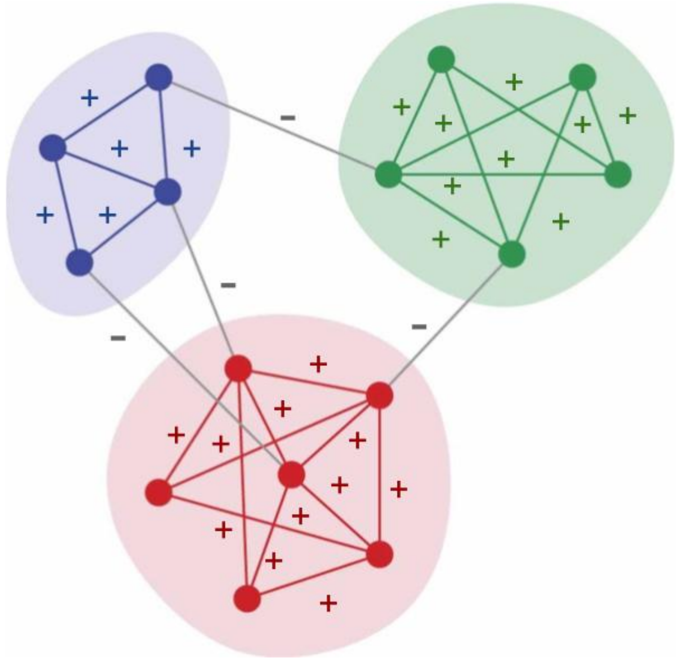
Frustration assumes communities should be **split through negative edges** and that they should **only contain positive edges** inside. Frustration is when this does not happen.

According to [Traag], it is possible to maximize modularity and minimize frustration at the same time in **signed networks** using the **Potts model** [Wu]. The **problem is NP-Hard** and, as such, we must use heuristics to be able to approximate it.

Formal problem definition

Provide communities (C) whose internal (inter) connections belong to E^+ while their connection to other communities (intra) are a part of E^-:

$$C = \begin{cases} if\ a_{i,j} < 0 \Rightarrow (v_i \in c_l) \wedge (v_j \in c_m) \wedge (l \neq m) \\ \quad else\ (a_{i,j} > 0) \Rightarrow (v_i \in c_k) \wedge (v_j \in c_k) \end{cases}$$



$\forall\ i, j \in$ Vertexes (V)

$\forall\ l, m \in$ Communities (C)

Related approaches

Existing algorithms for partitioning [Buluç]

Multilevel Graph Algorithms

Achieve a global solution by local processing, thus parallelizable

Global Algorithms

Directly creates a cut considering the whole graph (generally used on small graphs because of their high complexity)

Iterative Improvement Heuristics

Focused on improving previously generated solutions

Coarsen

Simplify a graph

Graph cuts

K-way partition

Uncoarsen

Return to original

Evolutionary Methods and Further Metaheuristics

Very high-quality communities, but huge runtime

Main algorithms on signed graphs

Multilevel Graph Algorithms

Achieve a global solution by local processing, thus parallelizable

Global Algorithms

Directly creates a cut considering the whole graph (generally used on small graphs because of their high complexity)

Iterative Improvement Heuristics

Focused on improving previously generated solutions

Coarsen

Simplify a graph

Graph cuts

K-way partition

Uncoarsen

Return to original

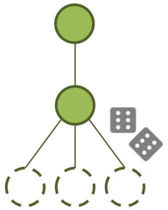
Evolutionary Methods and Further Metaheuristics

Very high-quality communities, but huge runtime

Random walk algorithms [Su et al.]

Initial solution

Based on the belief that every **community center** should be the node that has the **most connections to any other member** of its community.



Algorithm:

- Calculate node degrees
- Find all nodes *s* that have higher degree than its neighbors
- For each node in *S*:
  - Discover initial communities *C* using random walks
- Merge initial communities that are similar

Iterative improvement

Assign unclassified nodes (U) to detected communities depending on the probability of arriving to that community.

Algorithm:

- For every node *u* in *U* until *U* is empty*:*
  - Calculate $p(u_i \rightarrow C_j)$ to get to each community via + and – edges
  - If $p(u_i \rightarrow C_j)^+ > p(u_i \rightarrow C_j)^-$ assign to C_j
  - Else create new community with u_i
  - Remove u_i from U
- Merge communities that are similar

Results

- SRWA got “ground-truth” results along with tabu search algorithms.
- On real-world gene dataset it discovered more communities than other algorithms, but the whole set of communities still needs to be confirmed by biologists.

Improvement areas

- “Ground-truth” datasets are too **small**
- Does not consider **weights**

LabelRank: A Stabilized LPA [Xie et al.]

Different operators to stabilize

LabelRank stores, propagates and ranks labels in each node by using different operators:

- Propagation: Each element Pi'(c) holds the current estimation of probability of node i observing label c, given k(i) neighbors Nb(i).

$$P_i'(c) = \sum_{j \in Nb(i)} P_j(c)/k_i, \forall c \in C,$$

- Explicit Conditional Update: It updates a node only when it is significantly different from its neighbors in terms of labels.

Iterative improvement

- Stop criterion: We determine whether the network reaches a relatively stable state by tracking the number of nodes that update their label distributions.

Algorithm:

- Initialize adjacency matrix
- Repeat
  - Apply operators
- until stop criterion satisfied
- output communities

Results

- Classic LPA only works well on two connection dense real networks
- Stabilized LPA boosts performance in 28.57% on a +10k vertex dataset and 87.1% on a +30k vertex dataset
- SLPA outperforms other algo. like MCL and InfoMap in some cases

Improvement areas

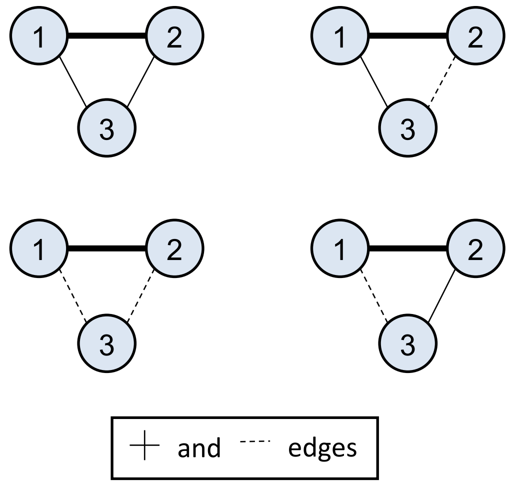
- Evolving networks: streaming

LPA w/ social balance theory [Fang et al.]

Social balance

Based on the principle that if two nodes belong to the same community (i.e. there is a positive edge between them), their relation to a third node should match.

Balanced Unbalanced

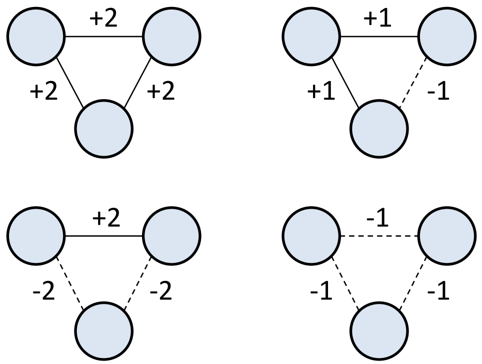


LP with structural balance [Fang]

Label propagation algorithm [Raghavan] with following label update strategy:

$$l_i = \frac{argmax}{l} \sum_{l_j=l,\, v_j \in N_i} K_{ij}$$

Structural balance (K) is set to 1. If the product of all 3 edges it positive, add 1 or -1 to the structural balance, yielding:



Results

- Better solution quality than traditional LP [Raghavan]
  - Average modularity
  - Normalized mutual information
- Decreased convergence time

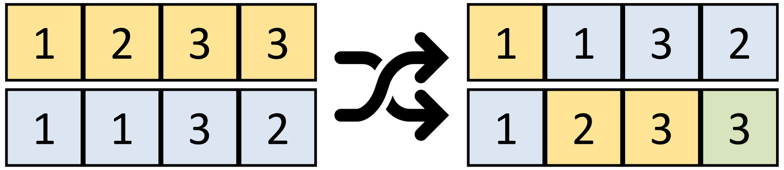
Improvement areas

- Not applied to **weighted** graphs
- Not applied to **real-world** graphs
- Datasets used are very **small**
  - SPN (v:10, e:45)
  - GGSN (v:16, e:116)
  - SMN (v:18, e:158)

Potts model approaches [Wu et al.]

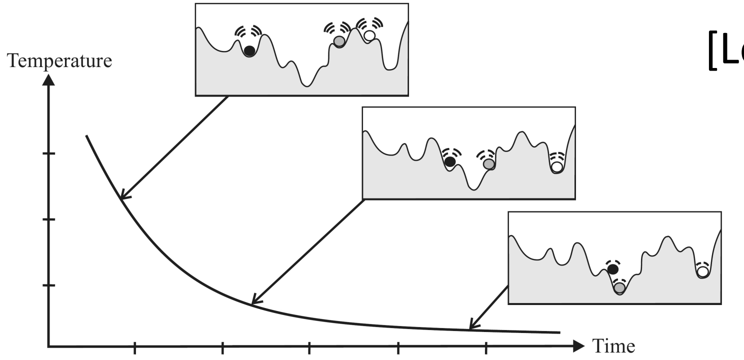
Genetic Algorithms [Amelio][Li]

- **Introduced by:** Holland and Goldberg [Goldberg].
- **Based on:** Biologic evolution and genetics.
- **Description:** They create an initial population of possible solutions and then perform breeding, mutation and selection processes on top of them to create the next generation of solutions.



Simulated Annealing [Traag]

- **Introduced by:** Kirkpatrick et al. [Kirkpatrick]
- **Based on:** Metallurgic process of annealing.
- **Description:** The algorithm allows the initial neighboring solutions to vary widely and gradually reduces the step-size in order to find the global minimum.



[Ledesma]

They are **inconvenient for time-sensitive** real-world applications because of their long execution time.

Technologies comparison

PROCESSING

- Blogel seems to provide better results overall, however it's still a relatively new tool (2014)
- The second option, GraphLab is not an open-source solution
- Giraph has a trajectory and is a well documented solution as it was already being used by Facebook for network analysis in 2013.

Framework	Analytic workload		Online workload	
	PageRank	WCC	SSSP	K-Hop
Giraph	3	3	3	3
GraphLab	1*	2	2	2
Blogel	2	1	1	1
Hadoop/HaLoop	5	5	5	5
GraphX	4	4	4	4

* GraphLab's pageRank approximation, by not using all vertices for computation, reduces the execution time and gives a decent-enough solution

STORAGE

- Neo4j and Dex (now called Sparksee) seem to be able to only process what they define as small graphs: 32k vertices and 256k edges.
- Neo4J provides a good support for different kinds of workloads, it makes it an ideal choice for the pre-processing of the data

Database	Load Workload	Traversal Workload	Intensive Workload	
			Read-only	Read-write
DEX	3	3	1	3
Neo4J	2	1	1	1
Orient DB	4	2	1	1
Titan*	1	4	2	2

Our approach

LPA on weighted signed graphs (1/2)

Implementation based on LPA modifications by [Fang et al. 2016] and [Xie et al. 2013]:

1. Operators to keep on each vertex:
   - Label rank list: $Labelrank_i(l)^T = Labelrank_i(l)^{T-1} + m_{i,j,l} \times a_{i,j} \quad \forall l \in Labels$
   - Conditional update history list:

Label	Frequency
4	16
1	15
1	6
...	...

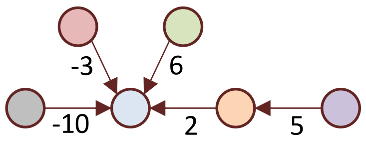
$q = 15$ threshold

2. Pre-clustering: social balance

$$isBalanced(v_i, v_j, v_k) = \begin{cases} 2\ (Yes), & E(v_i, v_j) \times E(v_j, v_k) \times E(v_i, v_k) > 0 \\ 1\ (No), & E(v_i, v_j) \times E(v_j, v_k) \times E(v_i, v_k) < 0 \end{cases}$$

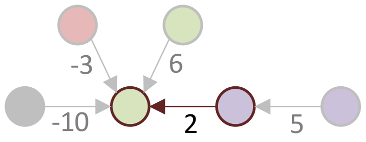
LPA on weighted signed graphs (2/2)

Superstep 0:



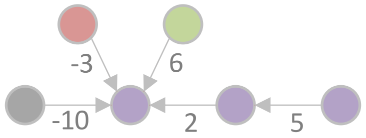
Label Rank
6
2
-3
-10

Superstep 1:

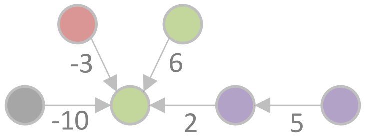


Label Rank
6
2
0
-3
-10

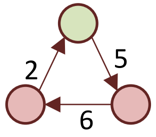
Superstep 2: **Label propagation**



Superstep 2: **Label Rank [Xie]**

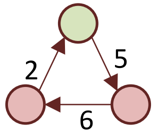


Superstep 0:



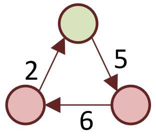
History
1
0

Superstep 3:



History
2
2

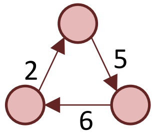
Superstep S:



History
S * (1/3) + 1
S * (2/3)

History threshold = 3

Superstep 4:



Implementation

Pre-processing

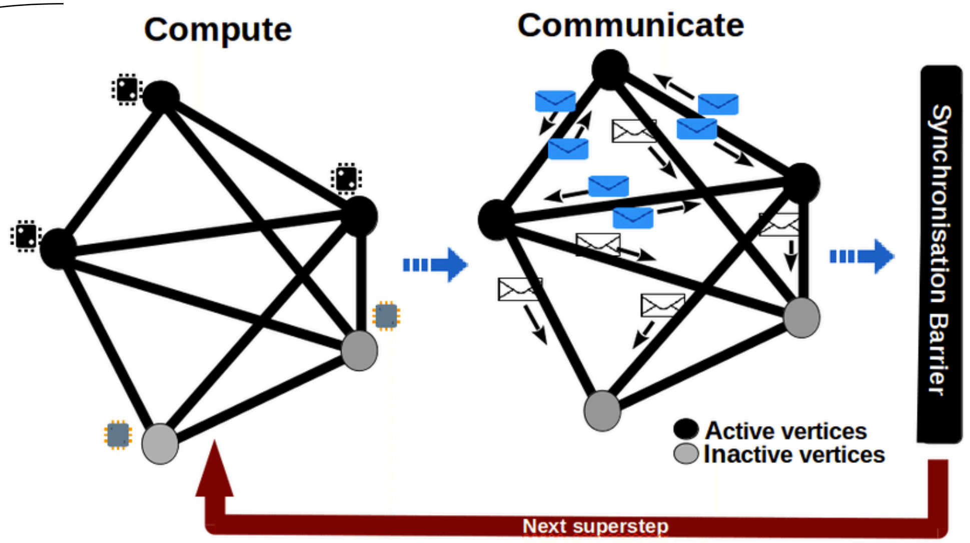
- Neo4J
- Jupyter

Main implementation

- Apache Giraph 1.1.0
- YARN
- MapReduce: 1 worker
- HDFS: Hadoop 2.4.1
- Docker: “uwsampa”

Host OS: MacOS Mojave

Infrastructure: - 1.6 GHz Intel Core i5
- 8 GB 1600 MHz DDR3



Bulk Synchronous Parallel (BSP) vs traditional iterative graph processing. By means of parallelization, it provides a linear complexity of O(max(degree)), but the synchronicity barriers create additional delays.

Dataset statistics

Ground-truth

Attribute	SPN	GGSN	Bitcoin OTC	Bitcoin Alpha	Epinions
Nodes	10	16	5,881	3,783	131,828
Edges	90	116	35,592	24,186	841,372
Positive Edges (%)	60%	50%	89.94%	93.63%	85.30%
Weight range	[-235,254]	[-1,1]	[-10,10]	[-10,10]	{-1|1}
min(Degree)	9	3	1	1	1
min(In-degree)	9	3	0	0	0
min(Out-degree)	9	3	0	0	0
max(Degree)	9	10	1,298	888	3,622
max(in-degree)	9	10	535	398	3,478
max(Out-degree)	9	10	763	490	2,070

Experiments

History threshold

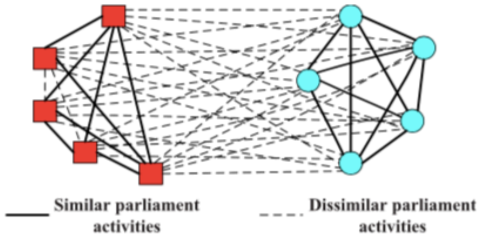
- Since we needed a **class history threshold** to make our algorithm converge and merge communities, we need to test if there exists an **optimal value** for any given graph or if the hyperparameter needs to be adjusted.

Testing

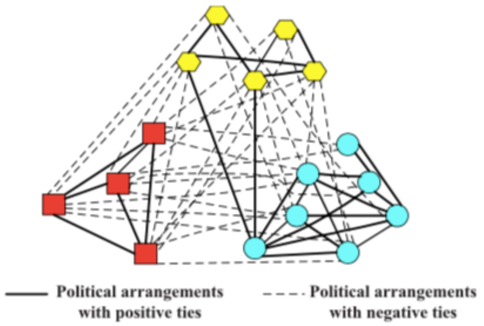
- In order to understand our implementation's performance, we had to evaluate it considering two different criteria:
  - Average normalized **mutual information** (Groundtruth)
  - Average execution time
- Running the algorithm in all 5 datasets to see performance on:
  - **Different sized** graphs
  - **Natural and ground truth** graphs
- No pre-processing (social-balance) is included

Social balance

- Running the algorithm in 2 datasets used by [Fang]
  - Slovene Parliamentary Network (SPN)

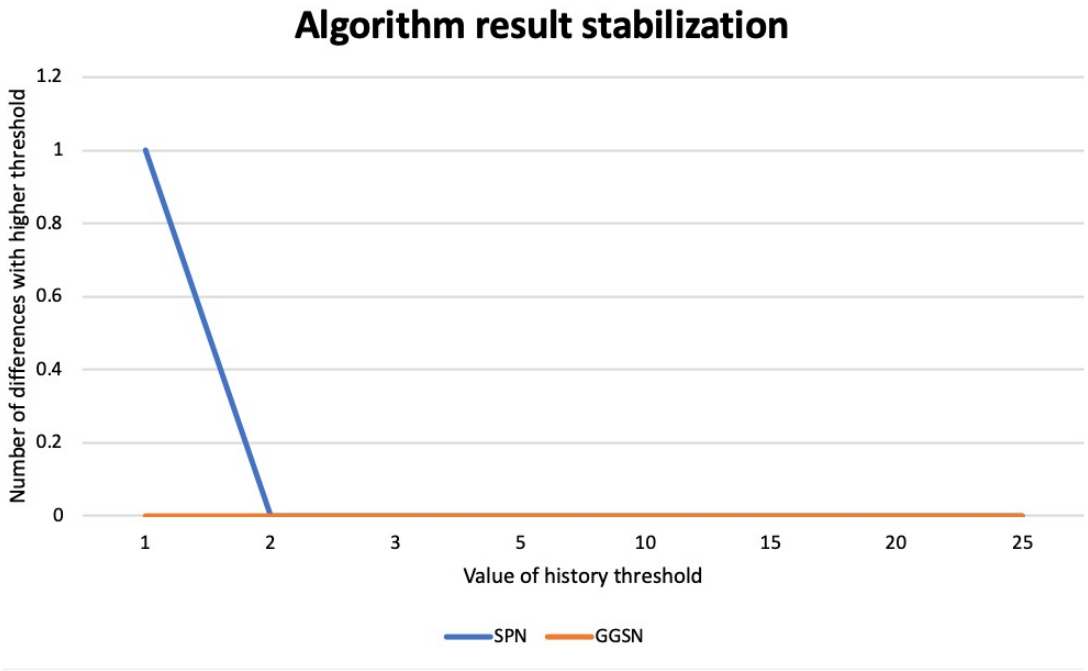


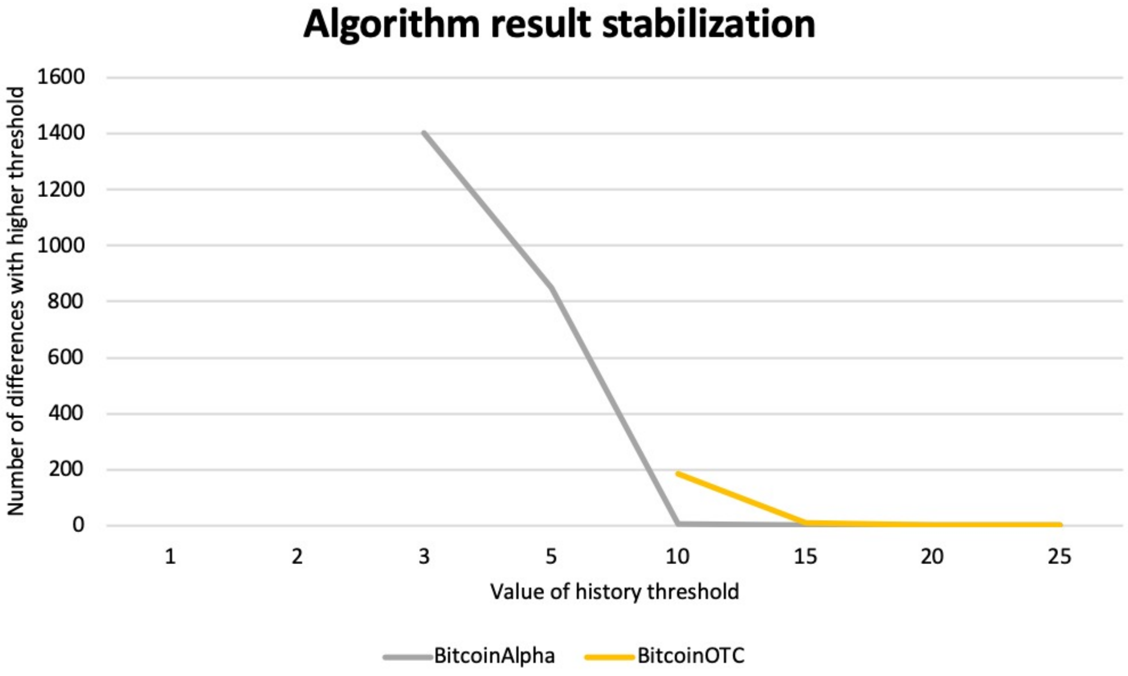
  - Gahuku-Gama Subtribes Network (GGSN)



Results and takeaways

Experiment 1: History update threshold





Experiment 2: Testing no-preprocessing

Statistics	SPN	GGSN	BitcoinAlpha	BitcoinOTC	Epinions
Nodes	10	16	3,783	5,881	131,828
Edges	45	116	24,186	35,592	841,372
Positive Edges	40%	50%	94%	90%	85%
Weight range	[235,-254]	{1,-1}	[10,-10]	[10,-10]	{1,-1}
min(Degree)	9	3	1	1	1
min(In-Degree)	9	3	0	0	0
min(Out-Degree)	9	3	0	0	0
max(Degree)	9	10	888	1,298	3,622
max(In-Degree)	9	10	398	535	3,478
max(Out-Degree)	9	10	490	763	2,070
RESULTS	**SPN**	**GGSN**	**BitcoinAlpha**	**BitcoinOTC**	**Epinions**
History Threshold	2	1	15 --> (10-15)	20 --> (15-20)	NA
Supersteps	6	4	54	88	NA
Setup	○ 0.10	○ 0.10	○ 0.14	○ 0.12	NA
Graph creation	○ 0.42	○ 0.46	○ 1.34	○ 1.39	NA
Execution	○ 0.92	○ 0.63	◑ 9.11	◑ 15.81	NA
Shutdown	● 9.11	● 9.09	◑ 9.26	◔ 9.42	NA
Total time	● 10.55	● 10.27	● 19.85	● 26.73	NA
# of communities	2	3	249	345	NA

Experiment 3: Including social balance

Statistics	SPN	SPN (SB)	GGSN	GGSN (SB)
Nodes	10	10	16	16
Edges	45	45	116	116
Positive Edges	40%	40%	50%	50%
Weight range	[235,-254]	[470,-508]	{1,-1}	{2,-2}
RESULTS				
Supersteps	6	6	4	6
Setup	○ 0.10	○ 0.10	○ 0.10	○ 0.17
Graph creation	○ 0.42	○ 0.43	○ 0.46	○ 0.55
Execution	○ 0.92	○ 0.97	○ 0.63	○ 1.12
Shutdown	● 9.11	● 9.15	● 9.09	● 9.03
Total time	● 10.55	● 10.66	● 10.27	● 10.87

Analysis (1/2)

Experiment 1:

- We discovered that our **hyperparameter** (history threshold) needs to be properly adjusted for every given graph. The bigger the size of the graph, the greater the value of the parameter.

- Since the algorithm would **need to be run several times** to find an optimal setting for the hyperparameter, our algorithm makes more sense to be classified as a **metaheuristical algorithm**

Analysis (2/2)

Experiment 2:

- It was capable of achieving the ground-truth communities in all iterations in a decent amount of execution time
- For algorithms of the size of Epinions our infrastructure was not capable of executing the algorithm, a more robust one is needed
- The shutdown time in MapReduce takes in some cases 50% of the total time of execution

Experiment 3:

- As we presumed, the social balance pre-processing has no positive effect in the results and actually causes a large execution time

Conclusions

- The implementation shows promising results in terms of robustness and accuracy; for the ground-truth datasets as the expected solution was garnered and it was maintained through all iterations
- The proposed algorithm cannot be classified as anything other than a metaheuristical algorithm because of the existence of a hyperparameter
- Social balance pre-processing proved futile for a faster convergence
- For larger datasets (+100k nodes), a more robust infrastructure than the one proposed is needed
- Docker is practical for environment setup but proved to be problematic for debugging
- Giraph's reliance on MapReduce leads to large shutdown times that are problematic for smaller datasets

Thank You!

https://www.overleaf.com/project/5dc93c9c2e14f60001bcd322

Sources

- Alessia Amelio and Clara Pizzuti. Community mining in signed networks: a multiobjective approach. In Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining, pages 95–99. ACM, 2013.
- Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders and Christian Schulz. Recent advances in graph partitioning. In Algorithm Engineering, pages 117–158. Springer, 2016.
- Fa-Yueh Wu. The potts model. Reviews of modern physics, vol. 54, no. 1, page 235, 1982.
- Ledesma, S., Ruiz, J., & Garcia, G. (2012). Simulated annealing evolution. In *Simulated Annealing-Advances, Applications and Hybridizations*. IntechOpen.
- Lei Fang, Qun Yang, Jiawen Wang and Weihua Lei. Signed network label propagation algorithm with structural balance degree for community detection. In International Conference on Smart Homes and Health Telematics, pages 427–435. Springer, 2016.
- M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. Phys. Rev. E, vol. 69, page 026113, Feb 2004.
- Santo Fortunato and Darko Hric. Community detection in networks: A user guide. Physics Reports, vol. 659, page 1–44, Nov 2016.
- Vincent A Traag and Jeroen Bruggeman. Community detection in networks with positive and negative links. Physical Review E, vol. 80, no. 3, page 036115, 2009.
- Yadong Li, Jing Liu and Chenlong Liu. A comparative analysis of evolutionary and memetic algorithms for community detection from signed social networks. Soft Computing, vol. 18, no. 2, pages 329–348, 2014.
- Yansen Su, Bangju Wang, Fan Cheng, Lei Zhang, Xingyi Zhang and Linqiang Pan. An algorithm based on positive and negative links for community detection in signed networks. Scientific reports, vol. 7, no. 1, page 10874, 2017.