A Parallel Algorithm for Tracking Dynamic Communities based on Apache Flink

Georgios Kechagias [1]   Grigorios Tzortzis [2]   Dimitrios Vogiatzis [2,3]
George Paliouras [2]

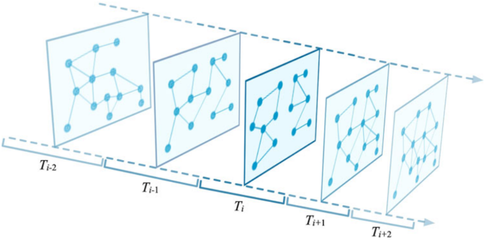
[1]School of Electrical and Computer Engineering
Technical University of Crete
Chania, Greece

[2]Institute of Informatics and Telecommunications
NCSR "Demokritos"
Athens, Greece

[3]The American College of Greece
Deree
Athens, Greece

Social Networks and Community Tracking

▷ Real life social networks are inherently highly dynamic



▷ Community tracking is the problem of locating the instances (i.e. counterparts) of a community in the different timeframes

▷ Common approach e.g. GED Method [1]:

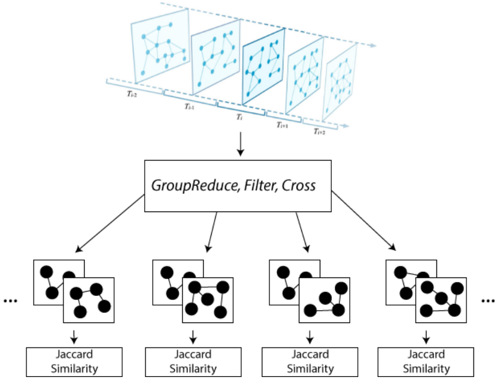
- Compare communities using some similarity measure
- Find their counterparts between consecutive timeframes

[1] Bródka et al. GED: the method for group evolution discovery in social networks.

Key Challenges - Our Objective

- Community Tracking algorithms have time complexity quadratic to the number of communities
- Contemporary real world social networks, contain thousands or even millions of users and communities
- Speed up the community tracking by parallelizing the community comparisons
  - Measure: **Jaccard Similarity**
  - Parallelizing framework: **Apache Flink**
- Evaluate the scalability of the algorithm using real world SN datasets

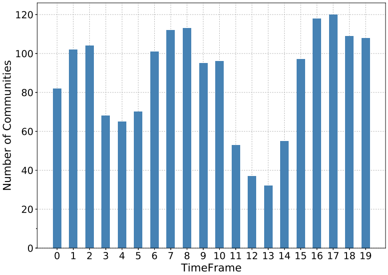
A Parallel Algorithm for Community Tracking

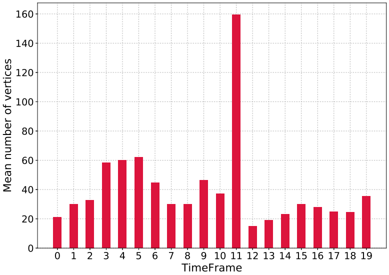


- Apache Flink tasks: *GroupReduce*, *Filter*, *Cross*
- **Parallelism** in Apache Flink is a configuration which defines the splitting of a task into subtasks
- Apache Flink assigns these subtasks to threads for execution

Crimea Dataset Characteristics

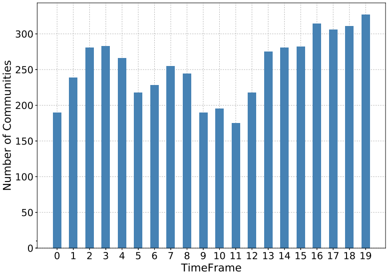
- ▷ 208,841 tweets
- ▷ Crimea crisis on the 18th of March 2014
- ▷ 20 timeframes
- ▷ 32-120 communities per timeframe
- ▷ on average, 15-160 vertices per community

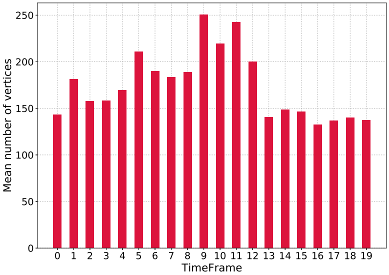




WorldCup Dataset Characteristics

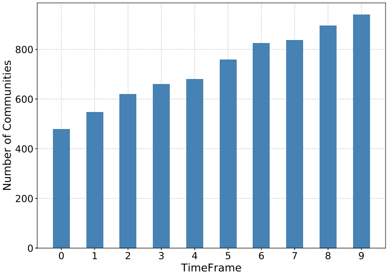
- ▷ 1,112,875 tweets
- ▷ 2014 FIFA World Cup, Between June and July 2014
- ▷ 20 timeframes
- ▷ 175-327 communities per timeframe
- ▷ on average, 132-250 vertices per community

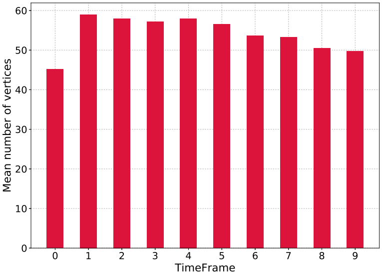




MathExchange Dataset Characteristics

- ▷ 376,030 posts
- ▷ Mathematics Stack Exchange Q&A website, Between 2009 and 2013
- ▷ 10 timeframes
- ▷ 479-940 communities per timeframe
- ▷ on average, 45-58 vertices per community





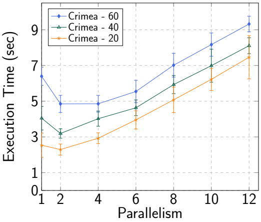
Parallel Algorithm vs GED

▷ The machine used for our experiments has:
- CPU: 12cores at 2.5GHz each
- RAM: 30GB

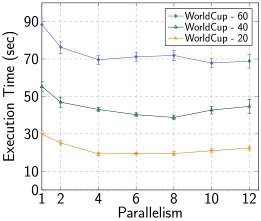
Dataset	Exec. Time (sec)		Difference
	GED	Parallel	
Crimea 20	21.67	2.53	88.3%
WorldCup 20	810.45	29.93	96.3%
MathExchange 10	1670.69	53.0	96.8%

Apache Flink's Parallelism Impact

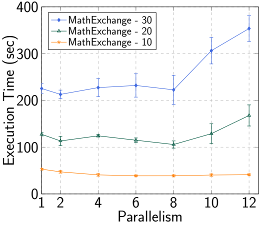
- We artificially enlarged the initial datasets $\times 2$ and $\times 3$ times in order to further evaluate the scalability of our algorithm
- Reminder: Apache Flink Parallelism defines the task splitting



Crimea



WorldCup



MathExchange

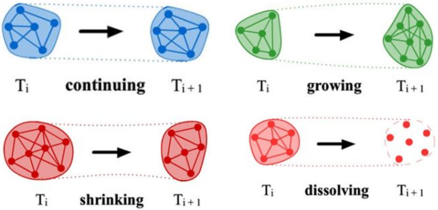
- High Parallelism is only effective when we have sufficiently large datasets
- The performance is increased when we tune appropriately Apache Flink for each individual dataset

Conclusion

- Our parallel method can exploit all available CPUs without any effort due to Apache Flink
- An alternative similarity measure can be easily incorporated
- Community evolutionary events can be calculated at a post-processing step using the output of our algorithm

Future Work

- ▷ Evolution categorization using event labels proposed in the literature



- ▷ Evaluation of more sophisticated similarity measures
- ▷ Extend to streaming using Apache Flink

Thank you for your attention

Questions?