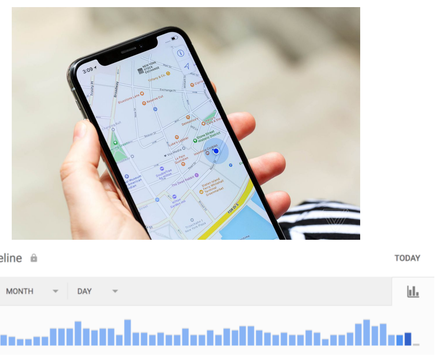
Distributed Processing Model for Temporal Graphs

Swapnil Gandhi and Yogesh Simmhan

Introduction

- Graphs are widely considered to be the natural means of representation for many networks
- But real-world networks are often evolving with links being added/removed and properties updated over time

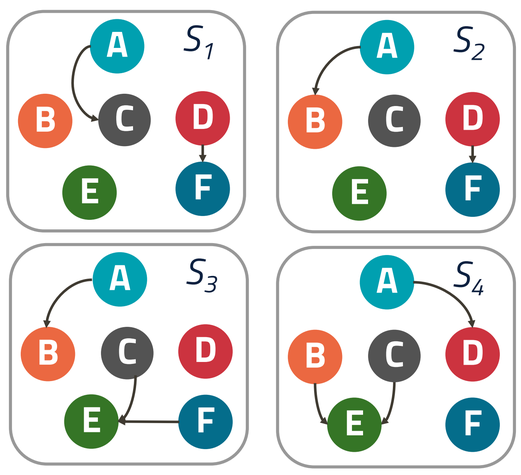
(a) Paper Map



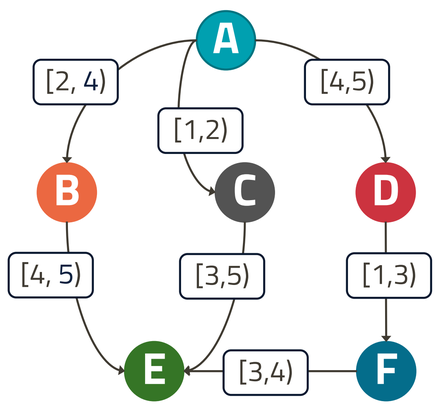
(b) Google Maps

- Temporal Graphs are dynamic networks which contain existence information for all vertices & edges at every point in time

  Examples : Social, Citation/Collaboration, Sensor & Transit Network, Human Connectomes, Internet-of-Things …



(c) Multi-Snapshot Graph



(d) Temporal Graph

- Challenges:

1. Lack of *unifying abstraction* to operate on temporal graphs

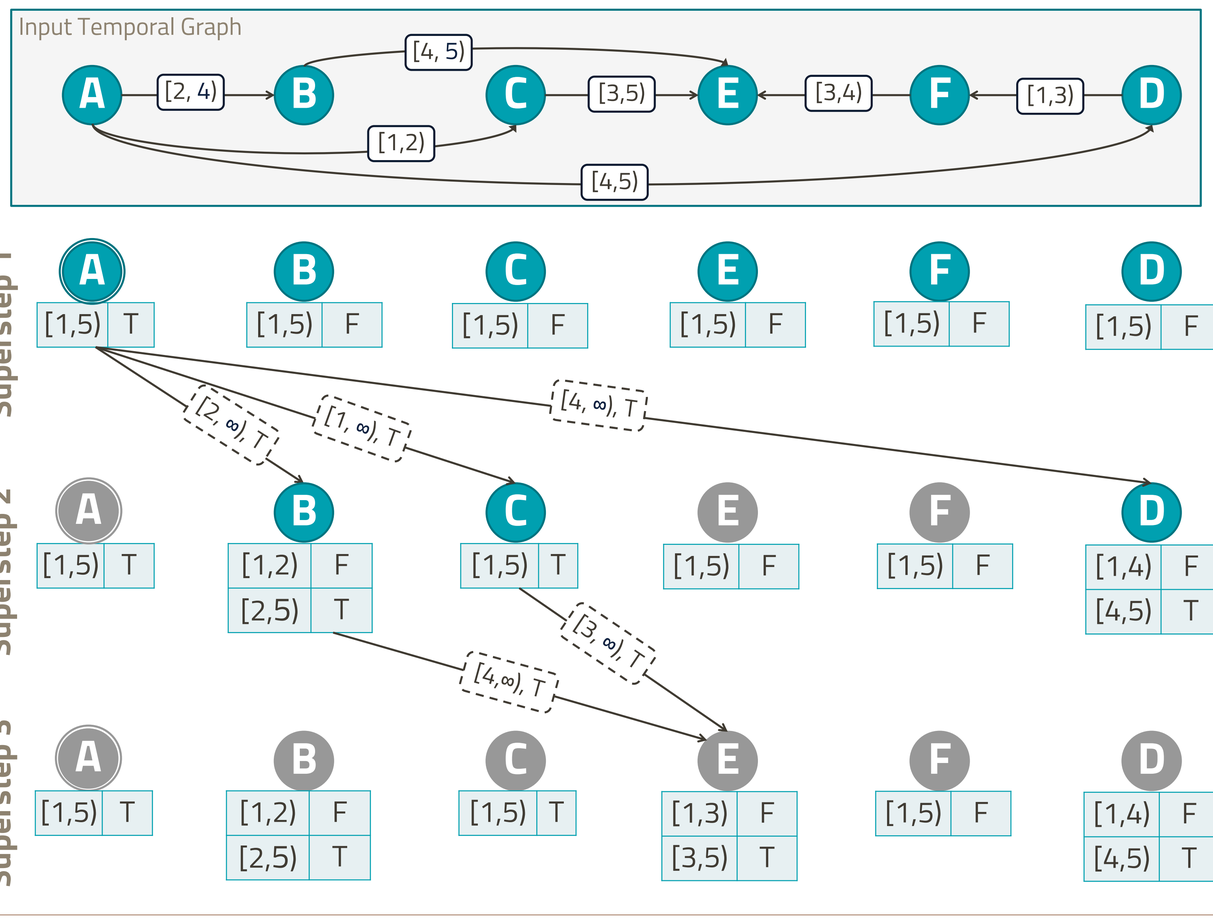
   Existing abstractions either work only for time-dependent algorithms or time-agnostic algorithms but not for both classes

2. Existing abstractions either *do not scale* on distributed systems or are *not even designed* for distributed execution

   Specialized native time-dependent algorithms are designed for single-threaded shared-memory execution

This Work : *First scalable, distributed and general-purpose programming abstraction for processing arbitrarily large temporal graphs*

Temporal Reachability



Interval-Centric Computation

- ICM extends vertex-centric computation model with Interval as a first-class citizen
  - Vertex (and edge) attributes, computation state and messages have *time-validity* and may have associated computation
- User writes program from the perspective of a single interval
  - Infrastructure lifts computation for arbitrary-sized graph

Compute method is executed at every active interval of a vertex in each superstep. Compute can inspect/modify state associated with active interval and has access to all interval messages received from previous superstep.

```
1  void init(Vertex v) {
2     v.setState(v.interval, false);
3  }
4
5  void compute(Vertex v, Interval t, boolean
       vstate, Message[ ] msgs) {
6     if(getSuperstep() == 1 && isSource(v)) {
7        v.setState(t, true); return;
8     }
9     if(!msgs.isEmpty()) v.setState(t, true);
10 }
11
12 Message scatter(Edge e, Interval t, boolean
       vstate){
13    return new Message(e, new
          Interval(t.start, ∞), true);
14 }
```

Initializes state of vertex v for its entire lifespan

Scatter permits user to send message with interval validity.
The function has read-only access to vertex state associated with interval and edge properties

Time-Warp

- Allows user-logic to consistently operate over multiple vertex sub-intervals in-parallel
  - Reminiscent of Temporal Aggregate
  - Uses one-pass algorithm
  - Support for online aggregation

State

τ_s	S
[0,5)	s_1
[5,9)	s_2
[9,10)	s_3

Messages

τ_m	M
[0,4)	m_1
[2,7)	m_2
[5,7)	m_3
[5,9)	m_4
[9,10)	m_5

Time Warp $\mathbb{X}_{S\times M}$

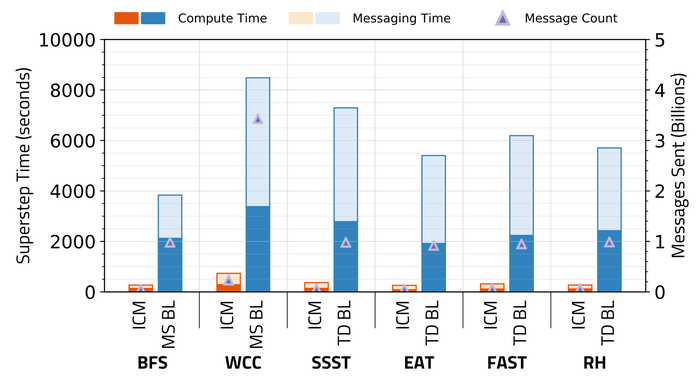
TW	τ	S	$\mathbb{M}$
w_1	[0,2)	s_1	m_1
w_2	[2,4)	s_1	m_1, m_2
w_3	[4,5)	s_1	m_2
w_4	[5,7)	s_2	m_2, m_3, m_4
w_5	[7,9)	s_2	m_4
w_6	[9,10)	s_3	m_5

Advantages :

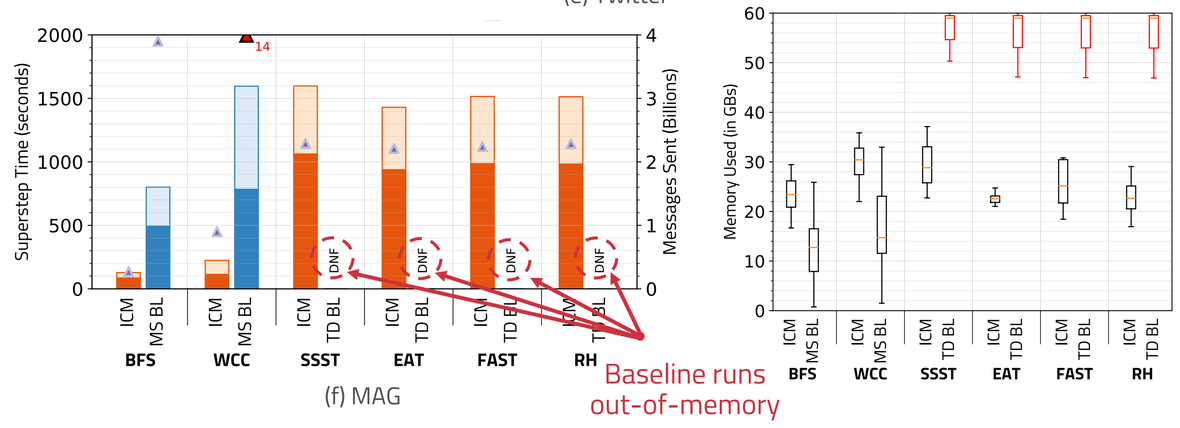
- Transparently performs temporal alignment, re-partitioning, replication and grouping
- Minimizes compute calls; coalesces interval-state and outgoing messages

Experimental Evaluation

Dataset	Snapshots	Avg. Lifespan V	Avg. Lifespan E	ICM Graph $\|V_{ICM}\|$	ICM Graph $\|E_{ICM}\|$	Multi-Snapshot Graph $\sum\|V\|$	Multi-Snapshot Graph $\sum\|E\|$	Transformed Graph $\|V_{TD}\|$	Transformed Graph $\|E_{TD}\|$
Twitter	30	29.5	28.4	43.9M	2.01B	1.3B	60B	219M	8.3B
MAG	219	20.9	12.75	116M	1.02B	3,4B	13B	2.6B	9.6B



(e) Twitter



(f) MAG

Longer Vertex and edge lifespan ➲ Better performance

Summary & On-Going Work

- First scalable API for processing temporal graph
  - Leverages existing solutions & makes it practical
- On-going work towards adding support for real-time streaming updates and temporal graph partitioning

Distributed Research on Emerging Applications & Machines Lab
Department of Computational and Data Sciences
dream-lab.in | Indian Institute of Science, Bangalore