Postmortem Graph Analysis on the Temporal Graph

Md Maruf Hossain, Erik Saule
{mhossa10, esaule}@uncc.edu
UNC Charlotte.

Abstract

Traditional graph analysis that expects a fixed data set is not sufficient for these kinds of temporal graphs. Streaming graph algorithms are more popular to handle this kind of problem. In this poster, we are going to show the performance analysis of *Pagerank* on the temporal graph. Most of the previous research study shows that *incremental* way more popular for the temporal graph.

But the question is if the data shows the offline nature that means if the whole data available at the beginning of the process, and need to perform a series of graph analysis for a list of time interval, should we still choose streaming graph algorithm for the temporal graph analysis? To find the answer, we need to look at another graph analysis, which is called *Postmortem* graph analysis. In the Postmortem graph analysis, one performs graph analysis on multiple subgraphs based on the well-defined time interval. In this study, we are going to show the *Postmortem* graph analysis can provide better Pagerank performance on the temporal graph than streaming graph analysis.

Dummy Temporal Edges

Table: Time interval T1 = (6/1/2021-9/15/2021), T2 = (7/1/2021-10/15/2021) and T3 = (8/1/2021-1/15/2022)

Edges v_1	Edges v_2	Edge Arrival Time	Time Interval T1	T2	T3
1	2	06/21/2021 4:25:27	✓	×	×
3	5	06/25/2021 9:21:23	✓	×	×
4	6	07/11/2021 5:18:27	✓	✓	×
2	3	08/01/2021 4:25:27	✓	✓	✓
2	4	08/11/2021 4:25:27	✓	✓	✓
5	6	09/13/2021 4:25:27	✓	✓	✓
2	7	10/02/2021 4:25:27	×	✓	✓
4	7	10/05/2021 4:25:27	×	✓	✓
5	7	10/06/2021 4:25:27	×	✓	✓
6	7	10/09/2021 4:25:27	×	✓	✓
1	2	11/05/2021 4:25:27	×	×	✓
1	3	11/06/2021 4:25:27	×	×	✓
2	5	11/09/2021 4:25:27	×	×	✓
3	5	11/12/2021 4:25:27	×	×	✓

Naive Pagerank

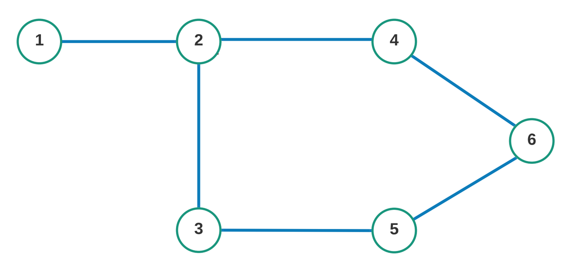

Figure: Graph structure for time interval T1.

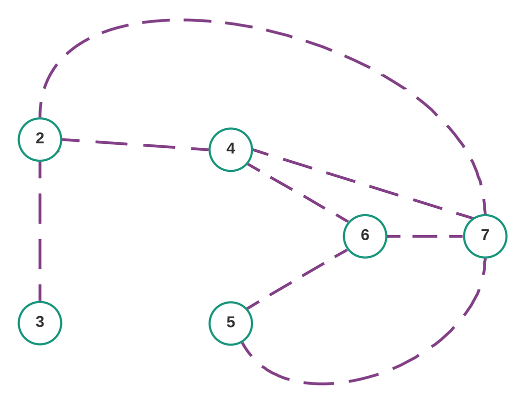

Figure: Graph structure for time interval T2.

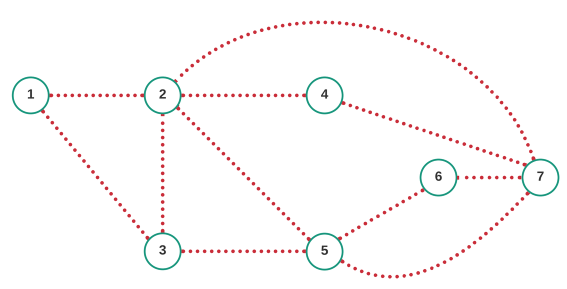

Figure: Graph structure for time interval T3.

Algorithm 1 Naive Pagerank

Input: $G = (V, E), \alpha, \tau, L_c, L_t, U_t$
▷ α teleportation constant.
▷ τ threshold for the difference between two consecutive rank.
▷ L_c maximum loop count to calculate pagerank.
▷ L_t lower time-stamp of the interval
▷ U_t upper time-stamp of the interval

```
G_s(V', E') = extractSubGraph(G, L_t, U_t)    ▷ find the sub-graph where e ∈ E' | L_t ≤ e_t ≤ U_t
δ = (1−α) / |V'|
prevRank(v) ← 1/|V'|, for all v ∈ V'
rank(v) ← 0, for all v ∈ V'
ε = 0
l=0
while l < L_c do
   for u ∈ V' in parallel do
      sum=0
      for v ∈ G → neighbor(u) do
         sum = sum + prevRank[v] / G → outDegree[v]
      end for
      rank[u] = δ + α * sum
   end for
   ε = findDiff(rank, prevRank)
   if ε ≤ τ then
      Break;
   end if
   prevRank = rank
   l = l + 1
end while
return rank
```

Streaming Pagerank

In the streaming system, a batch of edges arrive in the system and based on the `epoch` and `window_size` an edge can insert or delete from the system. And based on the `batch_id` the algorithm will enable to perform *Pagerank*. We choose STINGER [Rie16] to perform the streaming version of *Pagerank*. STINGER also support shared memory parallelization for the dynamic *Pagerank*.

Postmortem Pagerank

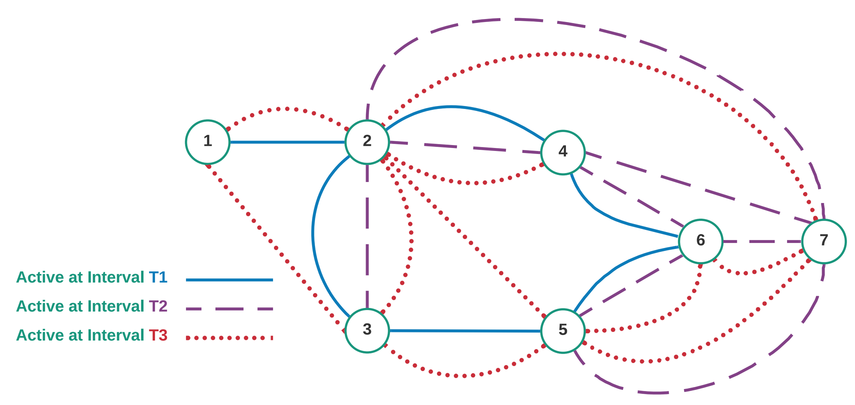

Figure: Graph structure for postmortem analysis.

Algorithm 2 Postmortem Pagerank

Input: $G = (V, E, T), \alpha, \tau, L_c$, intervals
▷ α teleportation constant.
▷ τ threshold for the difference between two consecutive rank.
▷ L_c maximum loop count to calculate pagerank.
▷ V contains all the vertices in non-descending oreder.
▷ T contains time-stamp for all edges.

```
for each interval< L_t, U_t > in intervals do
   activeNodes = 0
   activeOutDeg(v) = 0 for all v ∈ V
   trackNeighbor(v) = −1 for all v ∈ V ▷ to pick a single edge among more than one active edges between any two vertices.
   for u ∈ V do
      deg= 0
      for v ∈ G → neighbor(u) do
         if L_t ≤ T[v] ≤ U_t and trackNeighbor[v] < u then
            deg= deg + 1
            G → setValidNeighbor(indx(v), v)
            trackNeighbor[v] = u
         else
            G → setValidNeighbor(indx(v), −1)
         end if
      end for
      if deg> 0 then
         activeOutDeg[u] = deg
         activeNodes= activeNodes + 1
      end if
   end for
   δ ← (1−α) / activeNodes
   prevRank(v) ← 1 / activeNodes, for all v ∈ V
   rank(v) ← 0, for all v ∈ V
   ε = 0
   l=0
   while l < L_c do
      for u ∈ V' in parallel do
         sum=0
         for v ∈ G → neighbor(u) do
            if G → isActiveNeighbor(v) then
               sum += prevRank[v] / G → activeOutDeg[v]
            end if
         end for
         if sum> 0 then
            rank[u] = δ + α * sum
         end if
      end for
      ε = findDiff(rank, prevRank)
      if ε ≤ τ then
         Break;
      end if
      prevRank = rank
      l = l + 1
   end while
end for
```

Graphs

Table: Graphs and temporal analysis information for pagerank.

Name	Epoch	Window Size
stack-overflow (8/1/2008-3/5/2015)	100	0.8
		0.7
	80	0.6
		0.5
wiki-talk (8/30/2001-1/8/2007)	100	0.8
		0.7
	80	0.6
		0.5

Experimental Settings

1. Processor: Intel SkylakeX.
2. Number of Sockets: 2.
3. Cores: each socket contains 12 cores.
4. Threads: each core has 2 hyper-threads.
5. Cache: L1(32K), L2(1024K) and L3(19712K).
6. Operating System: Linux (ubuntu 16.04.7).

Pagerank on the stack-overflow

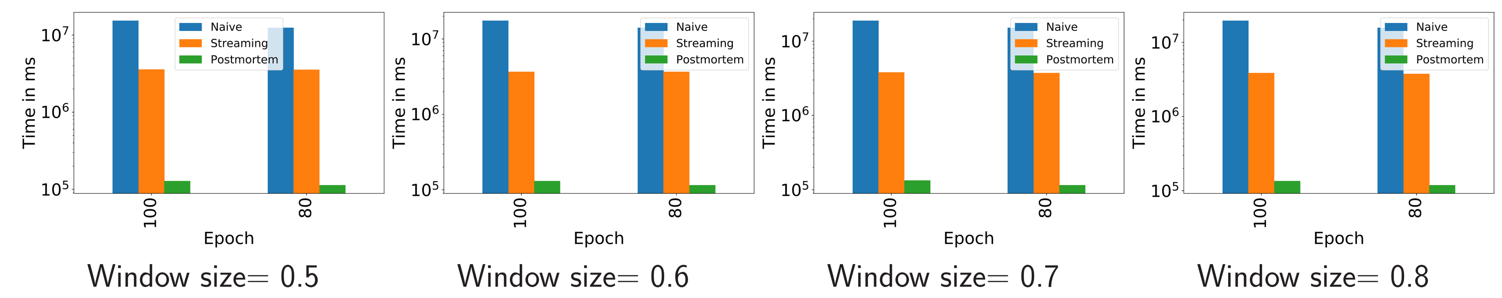

Figure: Pagerank performance of the streaming, naive and postmortem graph analysis on the sx-stackoverflow network for window-size = 0.8.

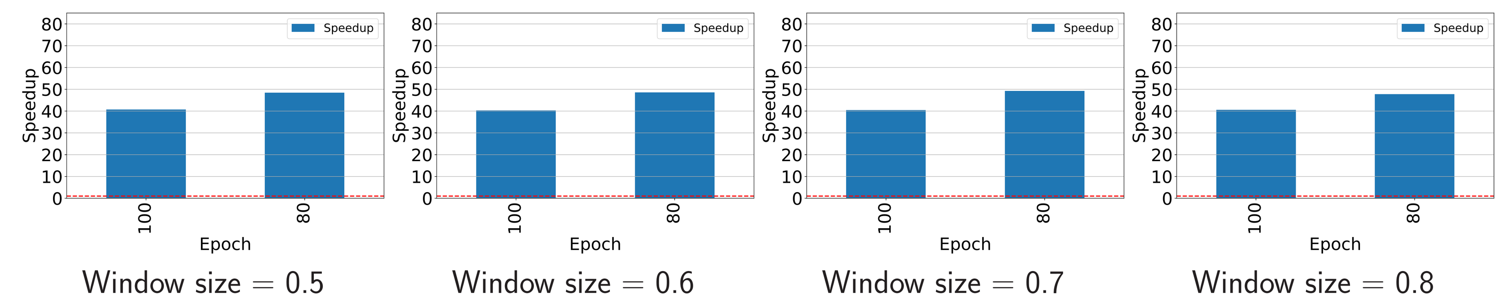

Figure: Pagerank performance speedup for the postmortem against streaming graph analysis on the stack-overflow network.

Pagerank on the wiki-talk

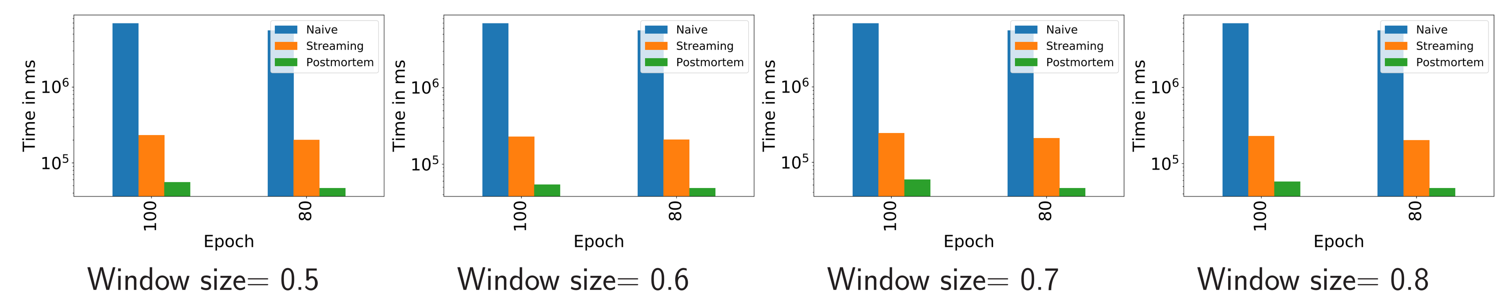

Figure: Pagerank performance of the streaming, naive and postmortem graph analysis on the wiki-talk network for window-size = 0.8.

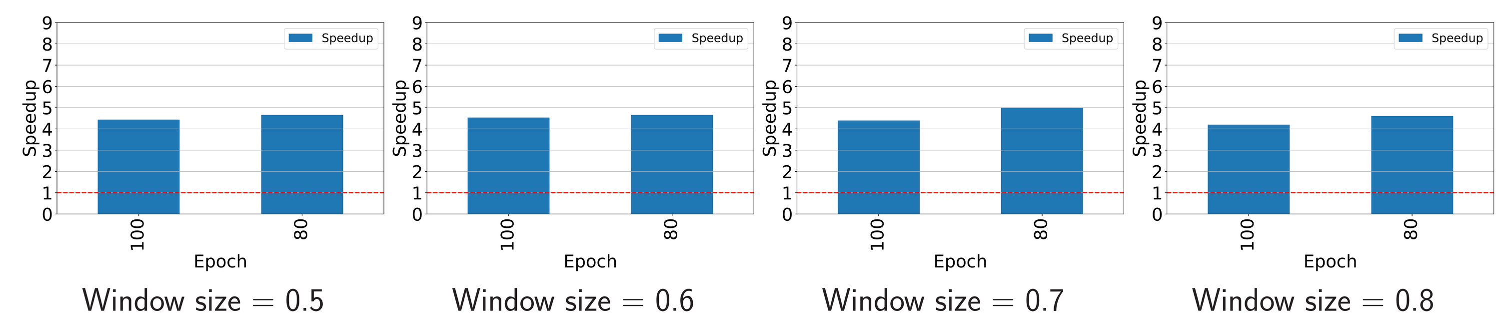

Figure: Pagerank performance speedup for the postmortem against streaming graph analysis on the wiki-talk network.

Summary

This study brings some insight into postmortem graph analysis on temporal graphs. Most of the state-of-art algorithms focus on the incremental version of the streaming model for event-based graphs. For online graph networks, it is obvious to use streaming data. But our study shows that *postmortem* Pagerank can overperform the streaming model on the temporal graph. In this work, we only focus on Pagerank. But in the future, we will explore other graph algorithms such as *Closeness* and *Betweenness Centrality*.

Acknowledgment

This work is supported by grant from the National Science Foundation CCF-1652442 and was made possible by a computing allocation given by TACC through XSEDE.

References

Jason Riedy, *Updating pagerank for streaming graphs*, 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE, 2016, pp. 877–884.