Path and Walk Problems in Temporal Graphs

Anuj Jain

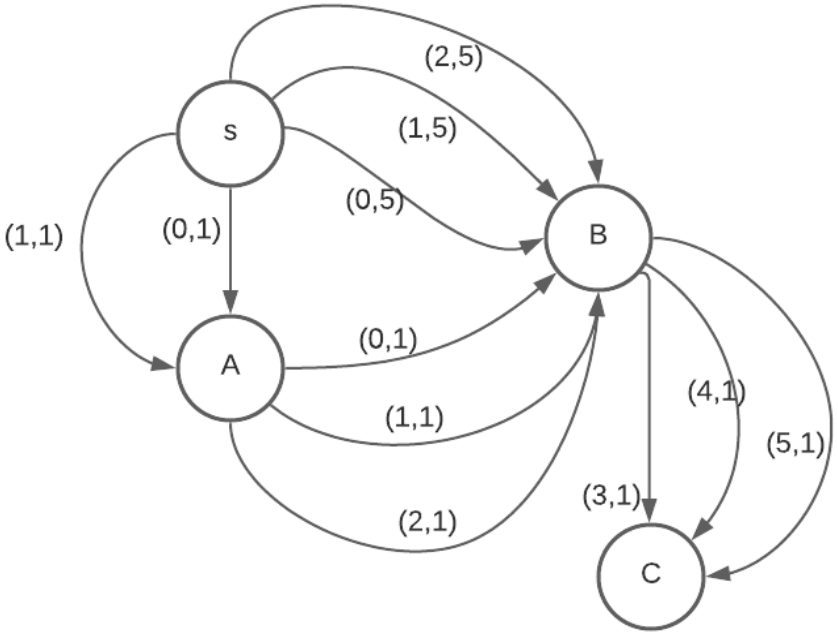
University of Florida, Gainesville and Adobe Systems Inc., USA

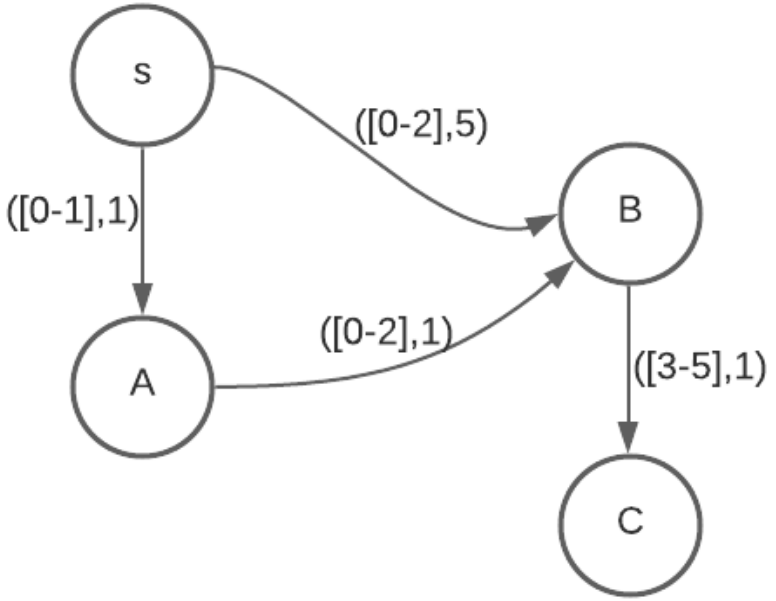
Joint work with Prof. Sarataj Sahni, University of Florida, Gainesville

Temporal Graphs

- Graphs that change over time.
- Vertices and Edges may appear/disappear or change their properties over time.
- Formally, temporal graph $G_T = \langle G_1, G_2, \ldots G_l \rangle$ sequence of static graphs G_i
- Each G_i represents a static graph at a time instant i
- Alternatively, G_i represents a static graph over a time interval $[t_i, t_{i+1}]$

Contact Sequence v/s Interval Temporal Graphs

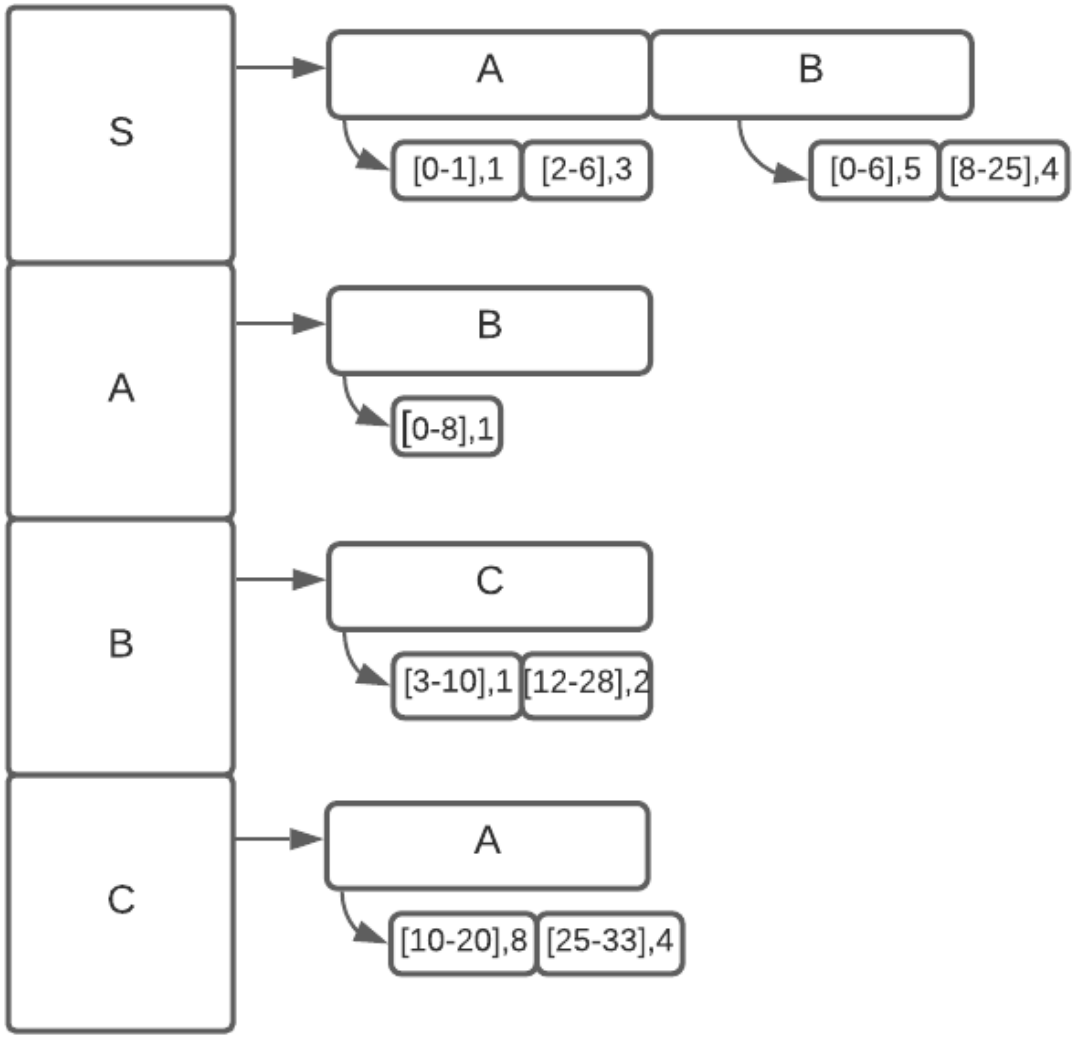




Practical Applications of Temporal Graphs

- Analyzing information spread on social networks
- Modeling disease spread during epidemics
- Finding optimal paths in ad-hoc wireless networks and computer networks
- Finding optimal flight routes from source to destination
- Finding optimal travel path in road networks

Data structure for Interval Temporal Graphs



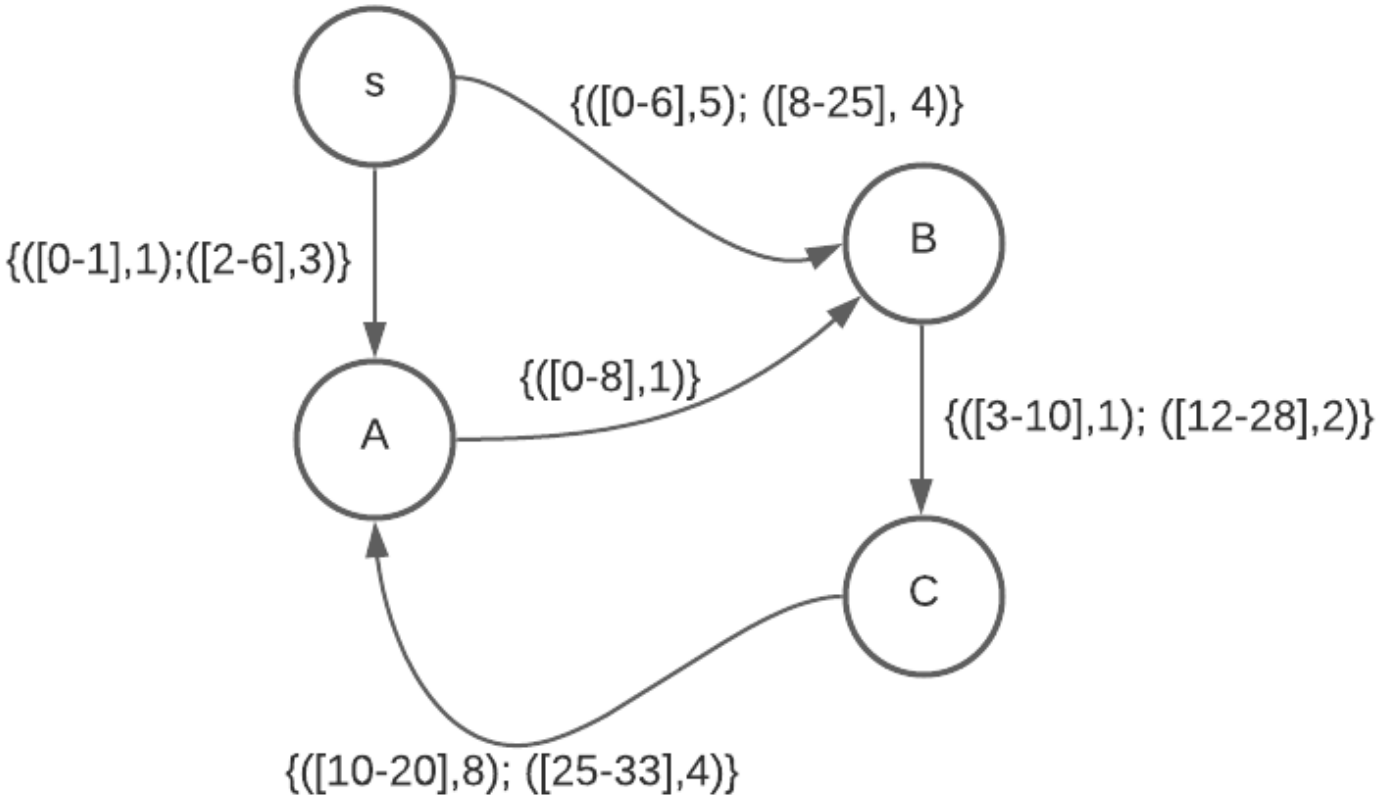
Optimal Paths in Temporal Graphs

Let S(s,v) be the set of all time respecting paths from ‘s’ to ‘v’. A path in S(s,v) is

- Foremost – Arrives earliest at the vertex ‘v’
- Min-hop – Arrives at ‘v’ with minimum number of hops
- Shortest – Arrives at ‘v’ with minimum travel time

Examples of Optimal Paths

- Optimal Paths from s to C:
  - Foremost Path: (s,0,A,1,B,3,C) arriving C at time 4
  - Min-hop Path: (s,0,B,5,C) arriving C with 2 hops at time 6



Our Foremost and Min-hop Algorithms on ITGs

- Handle variable travel times per interval while *Xuan's assumes travel times are same across all intervals for an edge
- Used different data structure than used by Xuan to represent ITG
- Foremost algorithm is a natural extension to Xuan's for variable travel time per interval
- Min-hop algorithm
  - Fundamentally different Algorithm than used by Xuan
  - For benchmarking,
    - Fixed a bug in Xuan's algorithm that affected correctness
    - Significantly improved the efficiency of Xuan's alg – original algorithm took excessive time
- Benchmarked against **Wu's algorithm as well that works only on Contact Sequence model

*Xuan, Ferreira, Jarry "Evolving graphs and least cost journeys in dynamic networks"

**Wu, Cheng, Ke, Huang, Huang, Wu "Efficient algorithms for temporal path computations"

Koblenz Datasets

- Temporal graphs of different social networks
- Have a fairly low activity ranging from 1 to 3.67
- Travel duration on every edge (λ) was set to 1

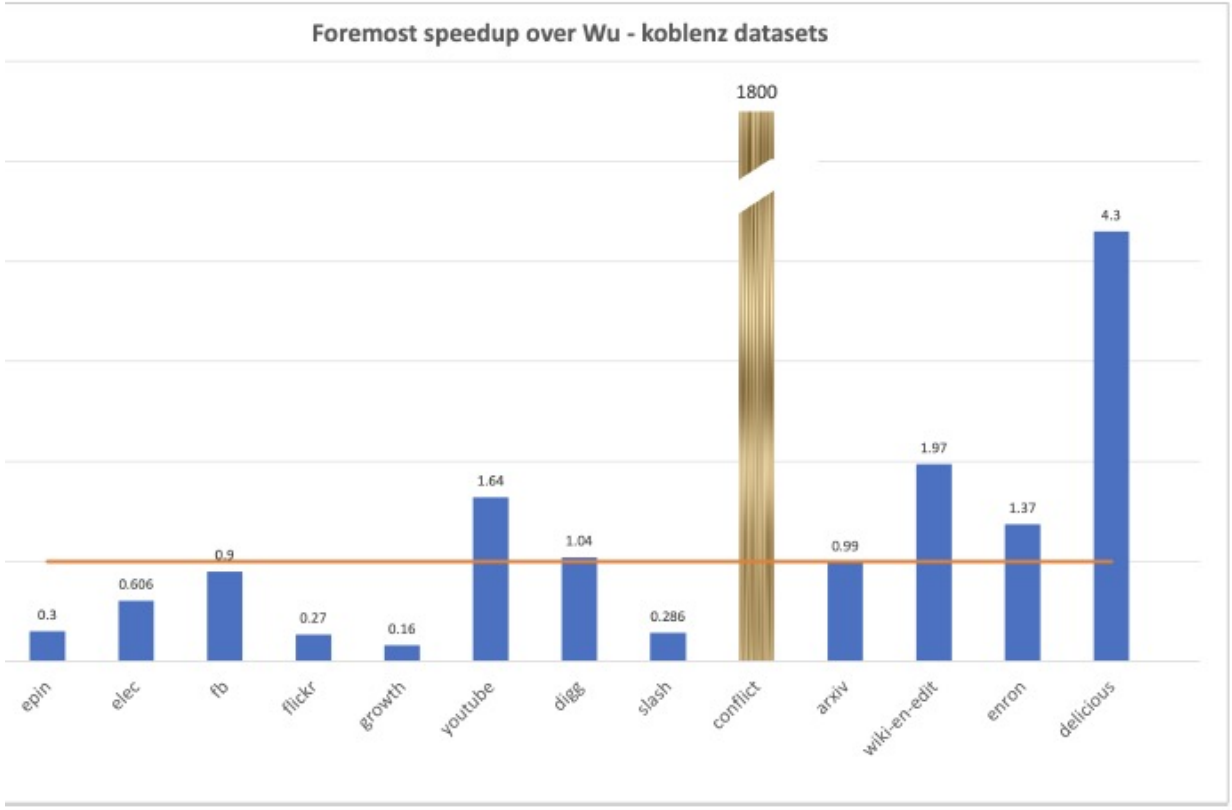
Dataset	$\|V\|$	$\|E_s\|$	$cs-edges$	Activity
epin	131.8K	840.8K	841.3K	1
elec	7,119	103.6K	103.6K	1
fb	63.7K	817K	817K	1
flickr	2,302.9K	33,140K	33,140K	1
growth	1,870.7K	39,953K	39,953K	1
youtube	3,223K	9,375K	9,375K	1
digg	30.3K	85.2K	87.6K	1.02
slash	51K	130.3K	140.7K	1.07
conflict	118K	2,027.8K	2,917.7K	1.43
arxiv	$28K$	3,148K	4,596K	1.45
wiki-en-edit	42,640K	255,709K	572,591K	2.23
enron	87,274	320.1K	1,148K	3.58
delicious	4,512K	81,988K	301,186K	3.67

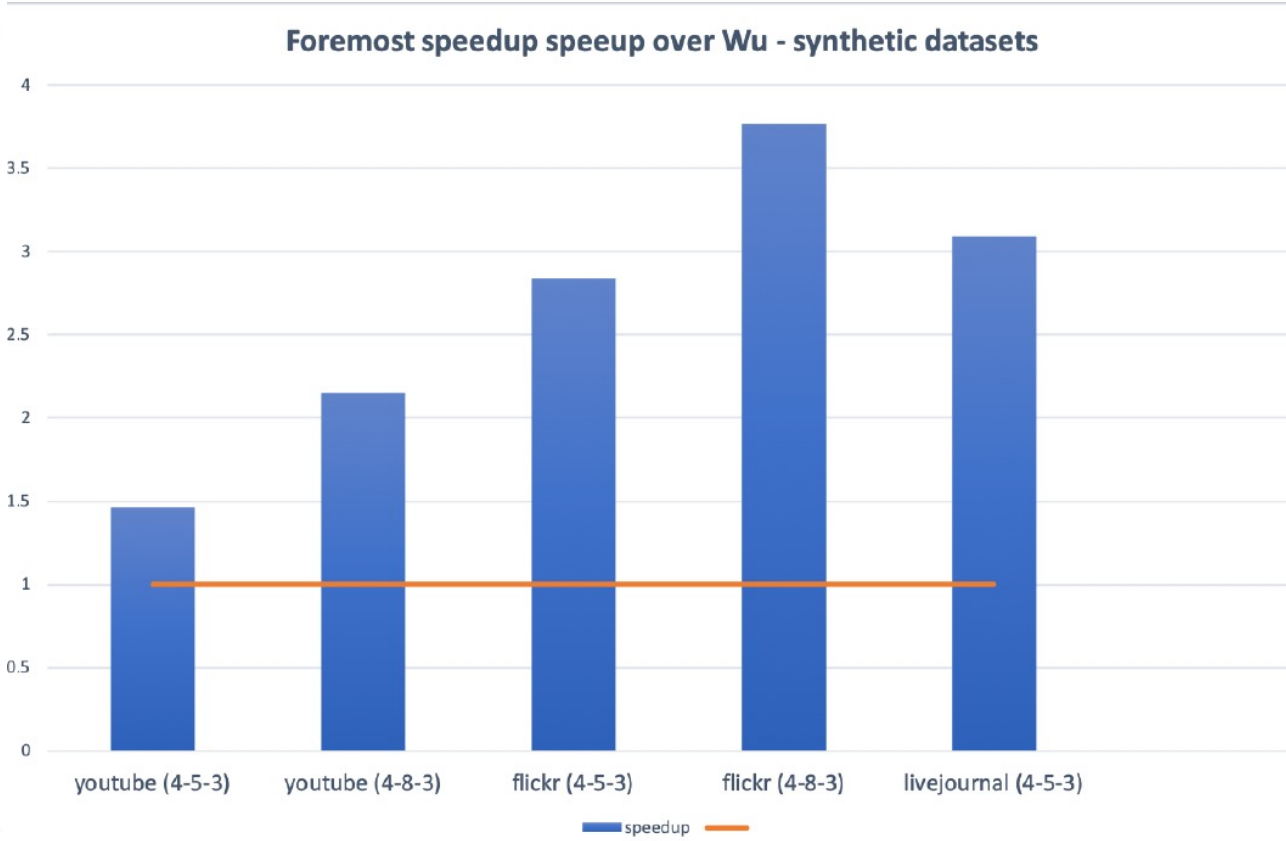
Synthetic datasets

- Synthetic datasets generated based on the social network graphs of youtube, flickr and livejournal
- Temporal parameters introduced on each connection for number of Intervals, length of interval and travel time
- Values assigned to each of these parameters based on normal distribution around a certain mean

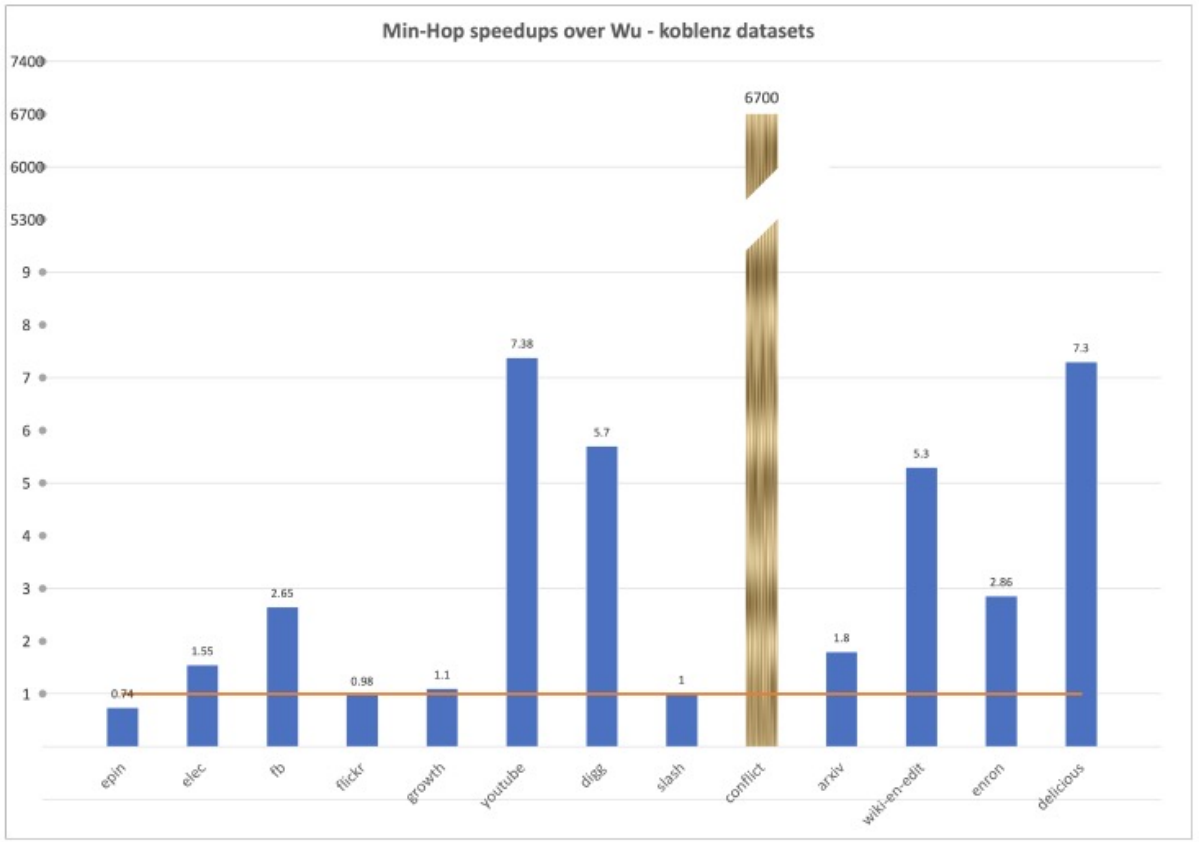
Graphs with $\mu_I = 4, \mu_D = 5, \mu_T = 3$				
Dataset	$\|V\|$	$\|E_s\|$	cs-edges	Edge Activity
youtube	1,157.8K	4,945K	105,039K	21.2
flickr	1,861K	22,613.9K	480,172K	21.24
livejournal	5,284K	77,402.6K	1,643,438K	21.3
Graphs with $\mu_I = 4, \mu_D = 8, \mu_T = 3$				
youtube	1,157.8K	4,945K	159,103.7K	32.1
flickr	1,861K	22,613.9K	727,405.9K	32.1

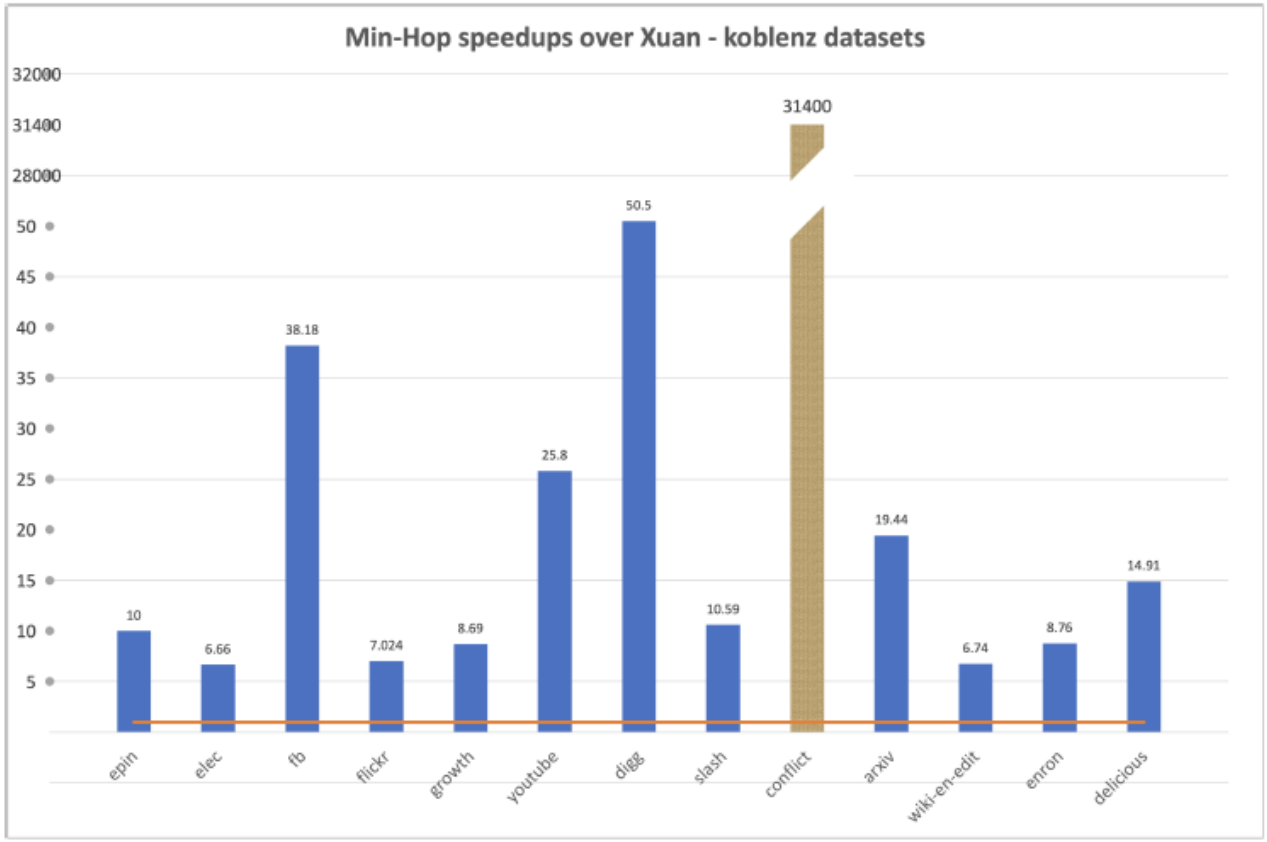
Foremost Comparisons with Wu et al.



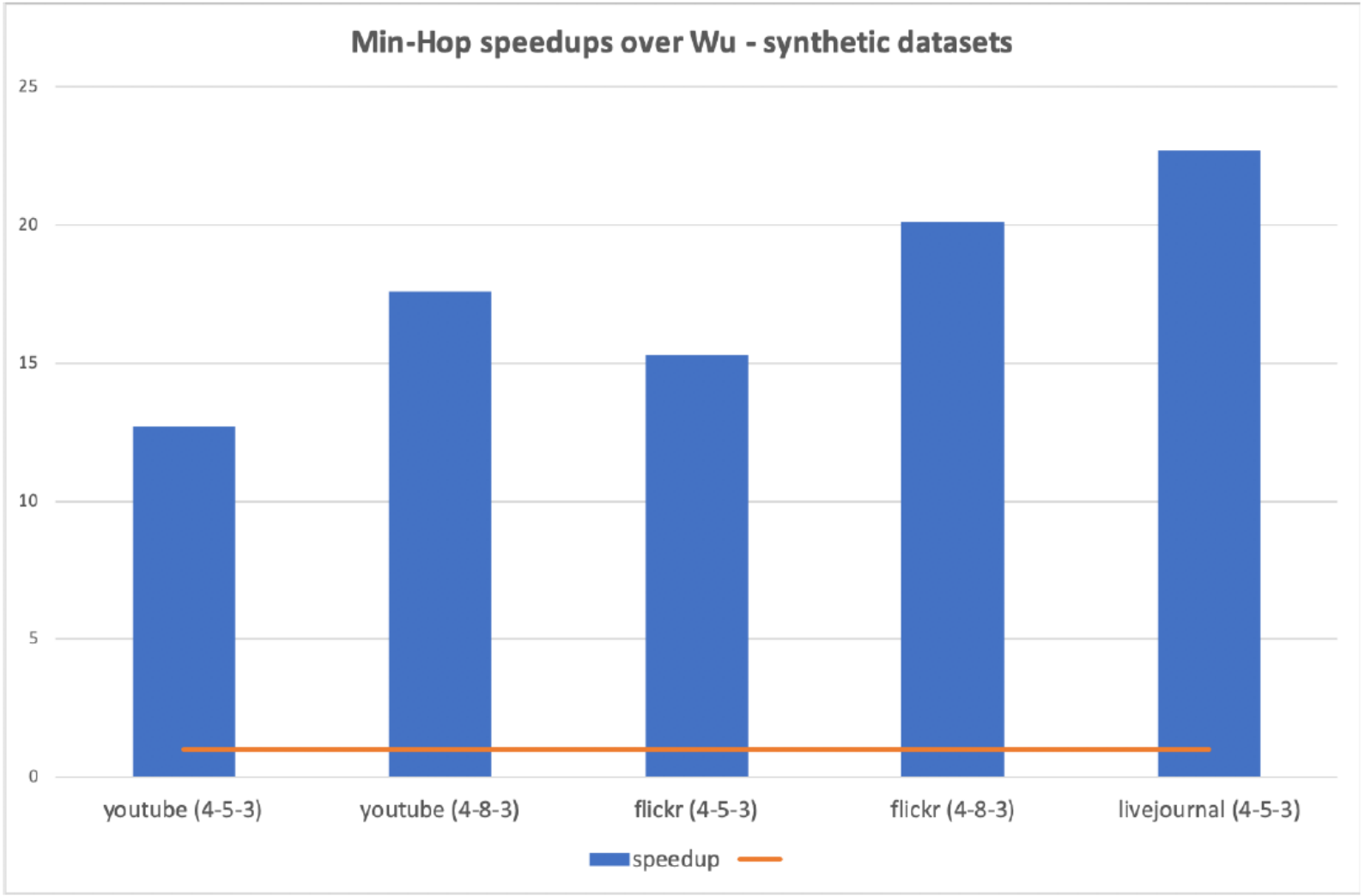


Min-Hop Speedups - Koblenz Networks





Min-hop Speedups- Synthetic Datasets



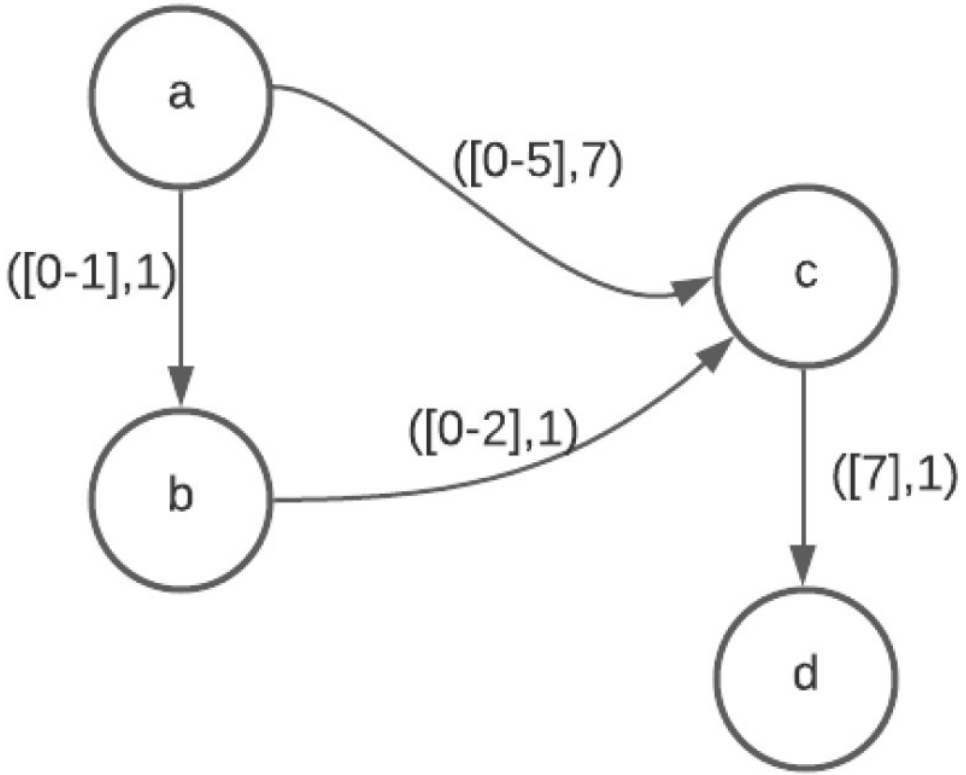
Problems with dual optimization criteria

- Min-Hop Foremost Paths (*mhf*) – Polynomial Algorithm for *mhf* paths
- Min-Wait Foremost Walks (*mwf*) – Proved *mwf* problem is NP-hard for ITGs. Pseudopolynomial Algorithm for *mwf* walks
- Min-Cost Foremost Paths (*mcf*) – Proved *mcf* problem is NP-Hard for ITGs. Pseudopolynomial algorithm for *mcf* would be similar to the Algorithm for *mwf* walks

Example *mhf* paths in ITG

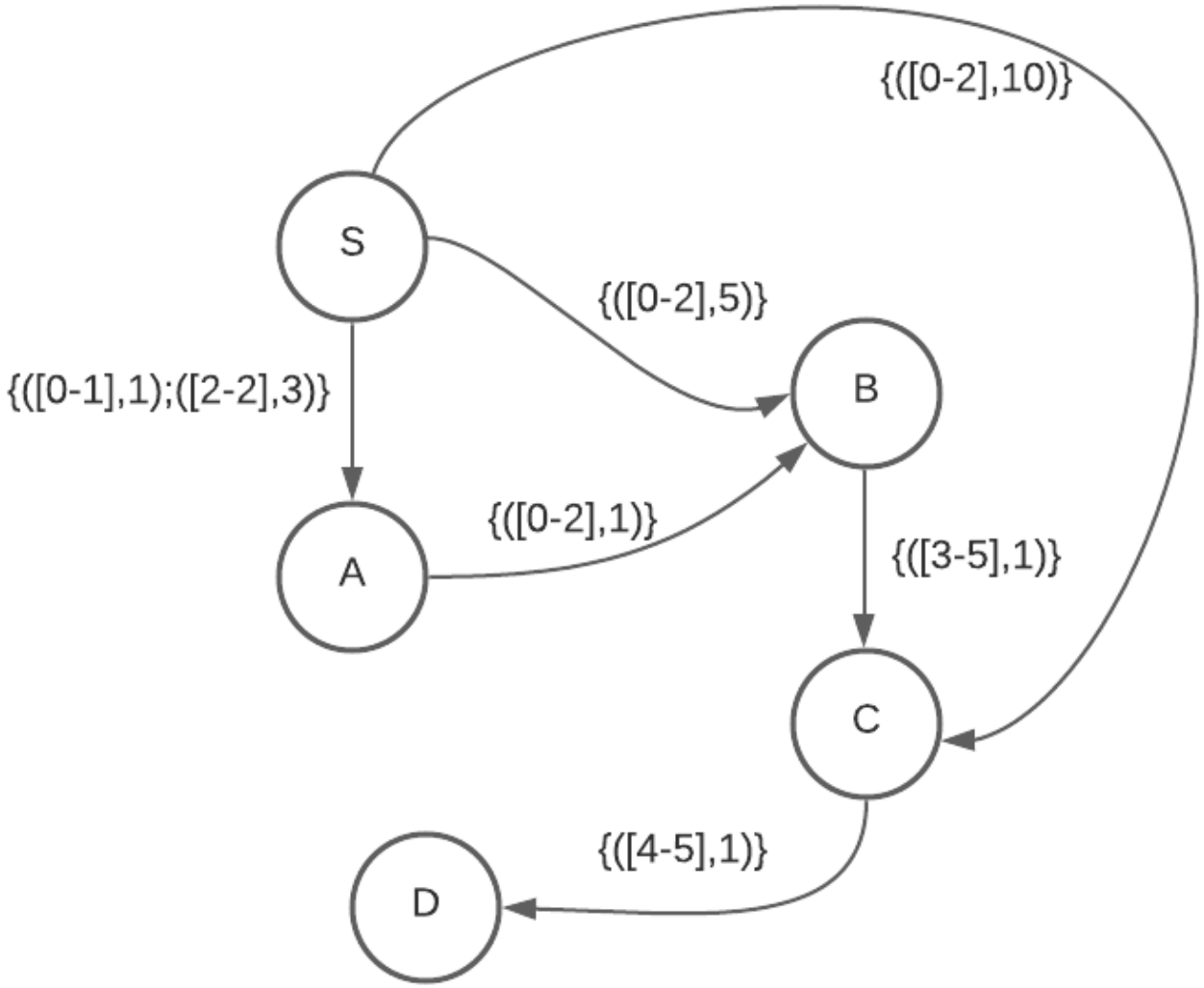
- Foremost Paths from a to d
  - P1= <a,0,c,7,d> arrives at 8
  - P2=<a,0,b,1,c,7,d> arrives at 8

- P1 is the only *mhf* path from a to d



mhf Hop by Hop Algorithm

- In hop 1 paths to (A,B,C) as (S,A) (S,B) & (S,C) are discovered with arrival times (1,5,10) respectively
- In hop 2 new path to B (S,A,B) & to C (S,B,C) are discovered with earlier arrival times of (2,6) at (B,C) respectively, so paths and arrival times updated.
- In hop 3, the 2-hop path to B is extended and new path to C (S,A,B,C) is discovered with an earlier arrival time of 4.
- Finally, 3-hop path to C is extend and path to D (S,A,B,C,D) is discovered arriving at 5
- *Mhf* paths to all vertices from s have been discovered.

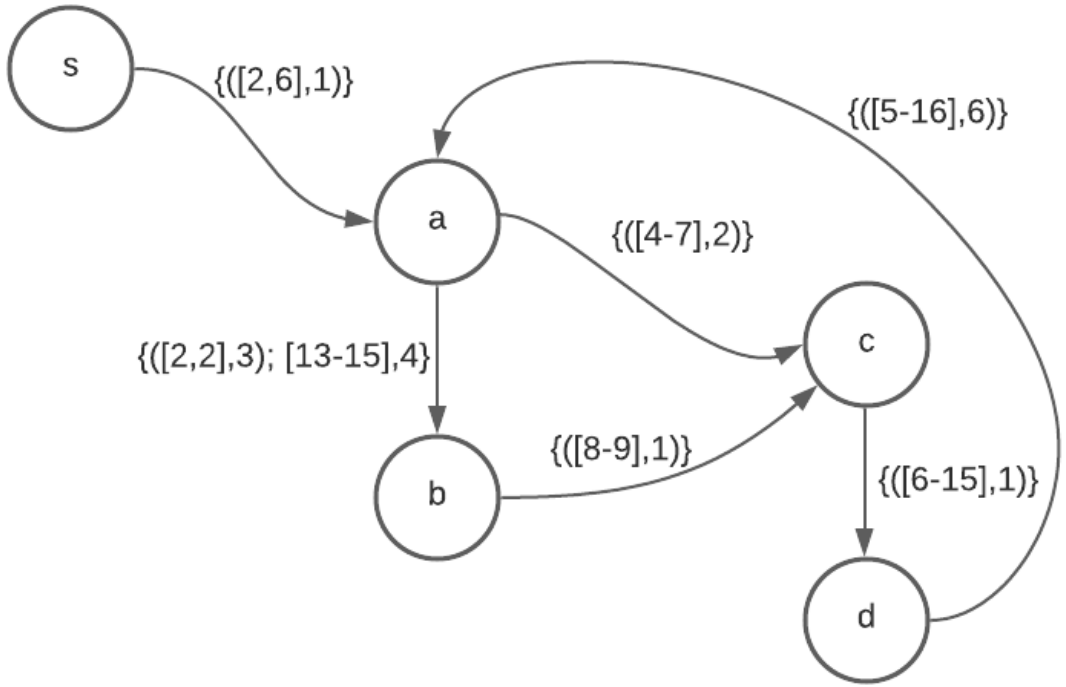


Mwf walks

- We prove *mwf* problem is NP-hard in Interval Temporal Graphs

- W(s,b)=<s,2,a,13,b> avl=17, wt=10

- W_{mwf}(s,b)= <s,3,a,4,c,6,d,7,a,13,b> arvl= 17, wait = 0

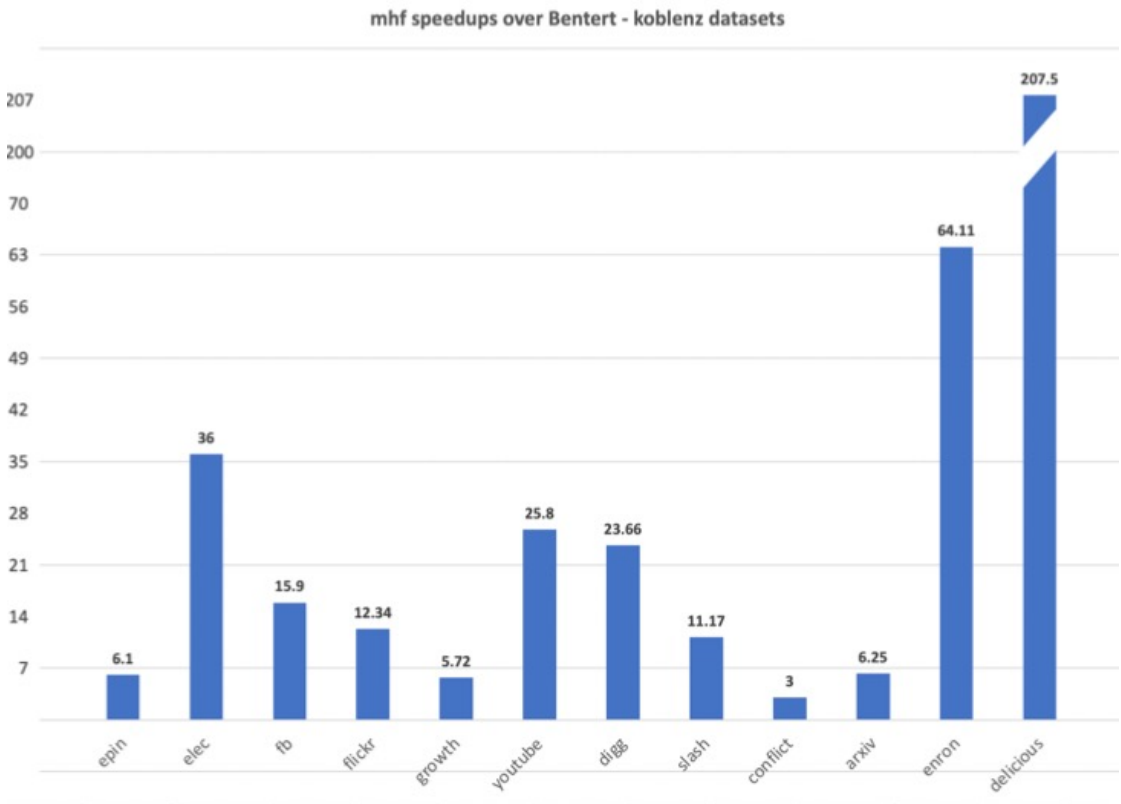


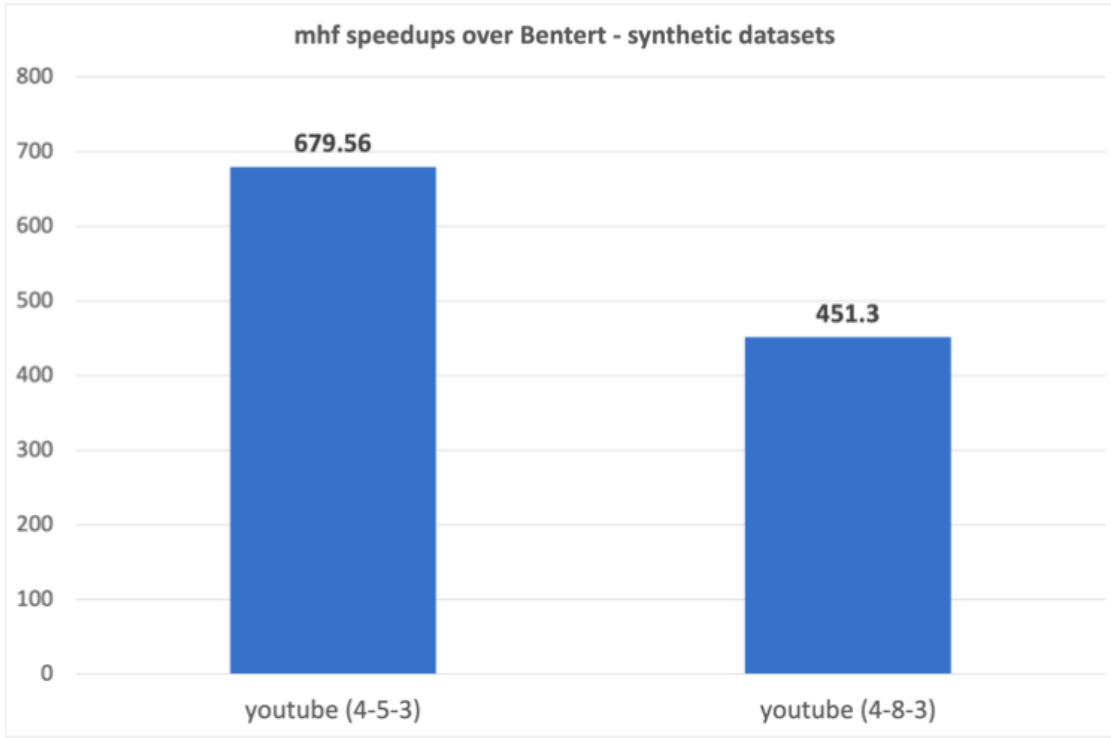
Linear Combination Algorithm in Contact Sequence Graphs by Bentert et al. (2020)

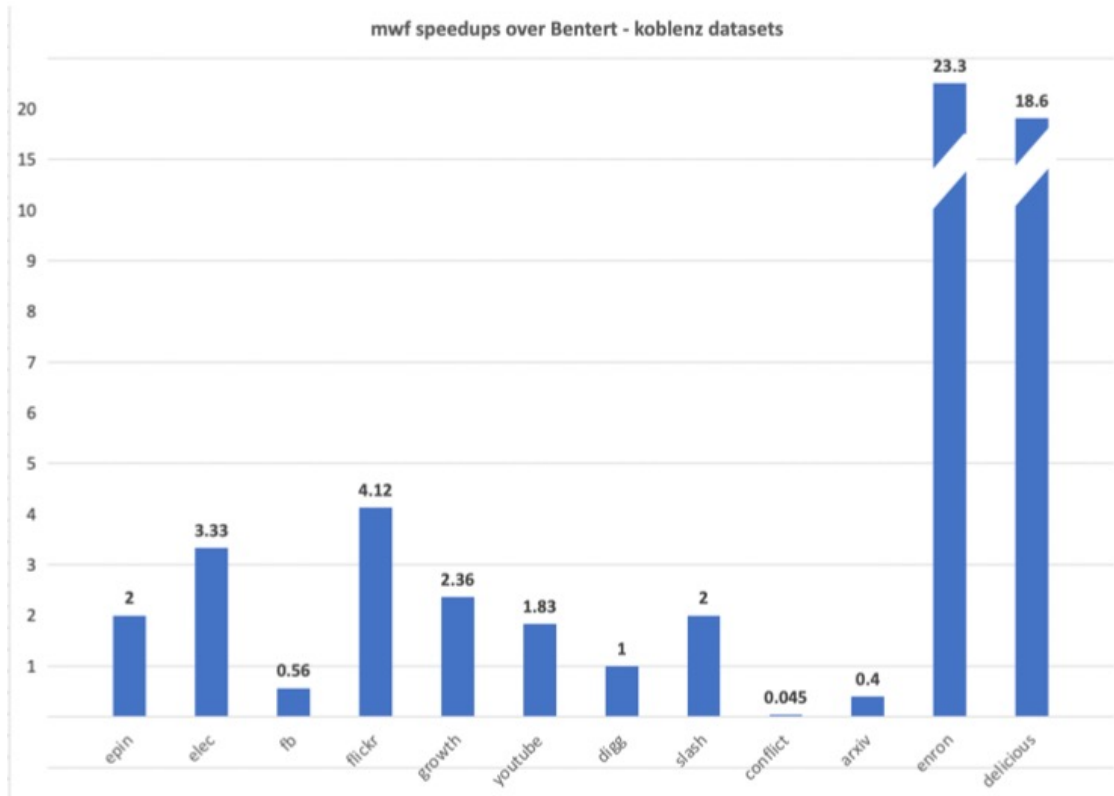
- Algorithm to find walks that minimize a linear combination of multiple criteria in contact sequence graphs
  - For eg minimize $\mathbf{c}_f t_a(v) + \mathbf{c}_w wait(s,v) + \mathbf{c}_h hops(v)$

- Choose coefficients such that linear combination is formulated to produce *mhf* paths and *mwf* walks.

- Benchmarked our *mhf* and *mwf* Algorithms on ITG against Bentert algorithm on CSG using this formulation.
  - For some of the datasets, the Bentert programs failed to run because of memory limitations
  - ITG models CSG graphs with zero duration intervals.


mhf speedups over Bentert - koblenz datasets
207
200
70
63
56
49
42
35
28
21
14
7
6.1
36
15.9
12.34
5.72
25.8
23.66
11.17
3
6.25
64.11
207.5
epin
elec
fb
flickr
growth
youtube
digg
slash
conflict
arxiv
enron
delicious


mhf speedups over Bentert - synthetic datasets
800
700
600
500
400
300
200
100
0
679.56
451.3
youtube (4-5-3)
youtube (4-8-3)


mwf speedups over Bentert - koblenz datasets
20
15
10
9
8
7
6
5
4
3
2
1
2
3.33
0.56
4.12
2.36
1.83
1
2
0.045
0.4
23.3
18.6
epin
elec
fb
flickr
growth
youtube
digg
slash
conflict
arxiv
enron
delicious

Optimization Criteria for Linear Combination

Consider a walk $w = \{u_1, t_1, u_2, t_2, \ldots, u_k\}; k \geq 2.$
Edges along w are $e_i = (u_i, u_{i\ +\ 1}, t_i, \lambda_i); 1 \leq i \leq k-1$

1. Arrival time at u_k: $(t_{k\ \_\ 1} + \lambda_{k\ \_\ 1})$. Min is foremost
2. Hop Count $k-1$: Min is min-hop
3. Start time t_1 : Latest departure time is reverse-foremost
4. Travel Duration $((t_{k\ \_\ 1} + \lambda_{k\ \_\ 1}) - t_1)$: Minimum duration is fastest
5. Travel time $\sum_{i=1}^{k-1} \lambda_i$: Minimum is shortest
6. Waiting time $\sum_{i=2}^{k-1}(t_i - (t_{i\ \_\ 1} + \lambda_{i\ \_\ 1}))$: Minimum is min-wait walk
7. Cost $\sum_{i=1}^{k-1} c_i$: Minimum is min-cost walk
8. Walk probability $\prod_{i=1}^{k-1} p_i$: Max Probability same as min $(\sum_{i=1}^{k-1}(-\log(p_i))$. So same as Cost criteria with $c_i = (-\log(pi))$

Optimal walks with waiting time constraints

$$w = \{u_1, t_1, u_2, t_2, \dots, u_k\}; k \geq 2$$
$$e_i = (u_i, u_{i\ +\ 1}, t_i, \lambda_i); 1 \leq i \leq k-1$$

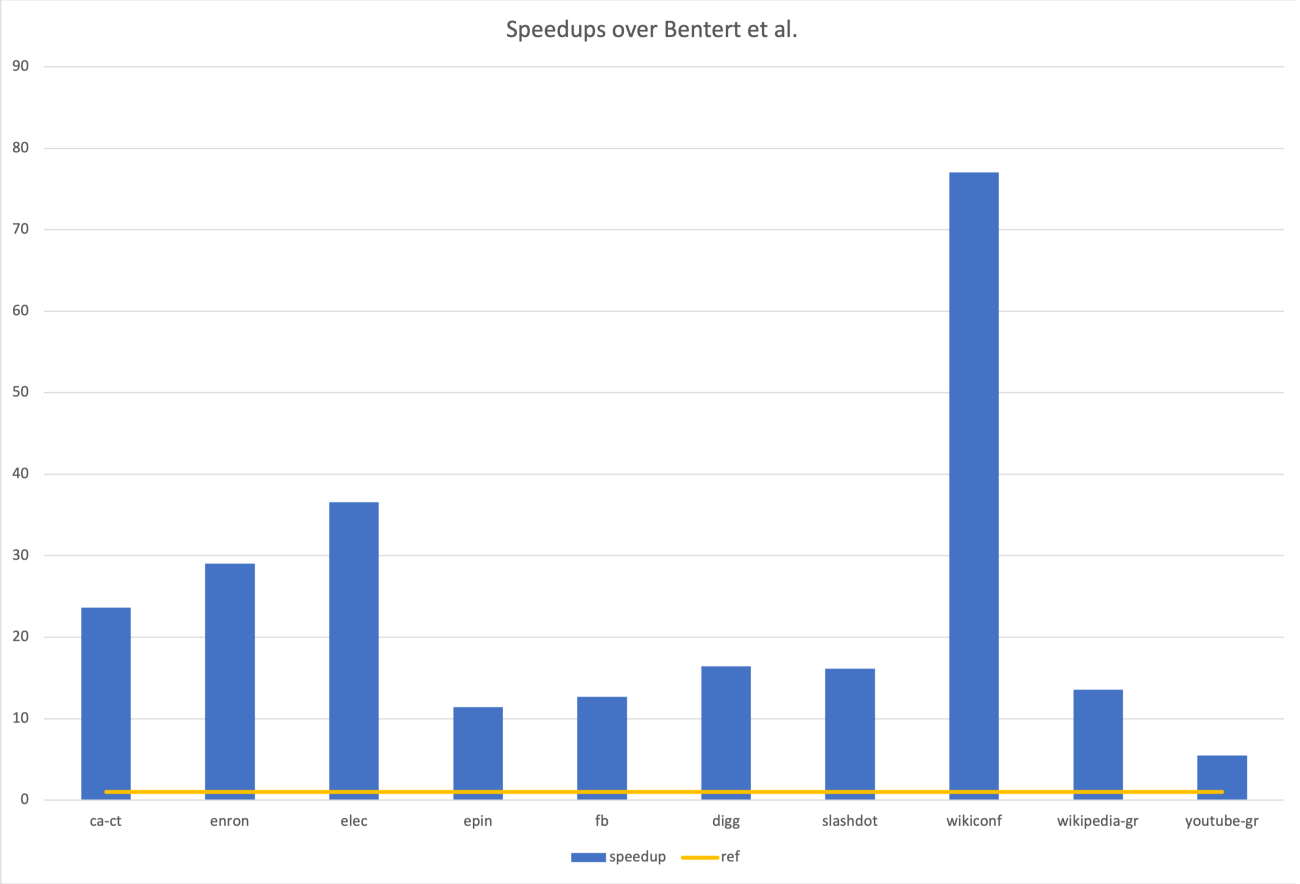
- Waiting time at each vertex: $t_i - (t_{i\ \_\ 1} + \lambda_{i\ \_\ 1})$
- Assume min-wait and max-wait constraints at each vertex, v are $\alpha(v)$ and $\beta(v)$, respectively
- Therefore, min and max wait time constraints:

$$t_i - (t_{i-1} + \lambda_{i-1}) \geq \alpha(v)$$
$$t_i - (t_{i-1} + \lambda_{i-1}) \leq \beta(v)$$

$$\begin{aligned} lin(w) = {} & c_{a.t} * (t_{k-1} + \lambda_{k-1}) \\ & + c_{dep} * (-t_1) \\ & + c_{t.d} * ((t_{k-1} + \lambda_{k-1}) - t_1) \\ & + c_{t.t} * \sum_{i=1}^{k-1} (\lambda_i) \\ & + c_{h.c} * (k-1) \\ & + c_{w.t} * \sum_{i=2}^{k-1} (t_i - (t_{i-1} + \lambda_{i-1})) \\ & + c_{cost} * \sum_{i=1}^{k-1} c_i \end{aligned}$$

New algorithm to minimize linear combination of optimization criteria with waiting time constraints for CSGs with no zero-duration cycle

- Our new Algorithm faster than Bentert et. al on all Koblenz datasets
- Our Algorithm is based on *TRG data structure to represent CSGs



*Extension of TRG by Gheibi, Banerjee, Ranka, Sahni, “An effective data structure for contact sequence temporal graphs”

Summary results for Path and Walk problems

- Demonstrated several problems are NP-Hard for ITGs but polynomial in CSGs
- Foremost path algorithm - benchmarked against Wu et al. (best known for CSG). Experimentally shown to be up to 1800 times faster
- Min-hop paths algorithm. Experimentally shown to be up to 6700 times faster.
- *mhf* algorithm. Experimentally shown to be up to 679 times faster than Bentert et al.
- Pseudopolynomial *mwf* algorithm. Experimentally shown to be up to 23.3 times faster than Bentert et al.
- Linear Combination Algorithm with no zero-duration cycle. Experimentally shown to be up to 77 times faster than Bentert et al.

Publications for this talk

1. Anuj Jain and Sartaj Sahni, “Min Hop and Foremost Paths in Interval Temporal Graphs” *2021 IEEE Symposium on Computers and Communications*

2. Anuj Jain and Sartaj Sahni, “Algorithms for Optimal Paths in Interval Temporal Graphs” *Applied Network Science Journal*, *Springer Open Access Journal*

3. Anuj Jain and Sartaj Sahni, “Min Hop Foremost Paths in Interval Temporal Graphs” *International Conference on Contemporary Computing, ACM ICPS 2022*

4. Anuj Jain and Sartaj Sahni, “Foremost Walks and Paths in Interval Temporal Graphs” *Algorithms Journal 10/2022*

5. Anuj Jain and Sartaj Sahni, “Optimal Walks in Contact Sequence Temporal Graphs with No Zero Duration Cycle” *2023 IEEE Symposium on Computers and Communications (to appear)*