# A Privacy-Preserving Algorithm for Graph Spanners

# Outline of the Project

- Graph Spanners
- K - Graph Spanner
- Previous Related Work on Privacy
- Problem Formulation
- Implementing Linear Programming
- Performance Measures and Evaluations
- Results
- Conclusions

# What is A Graph Spanner

Graph Spanners are important in graph applications where we want to reduce the number of edges in the graph without affecting the navigability of the graph.

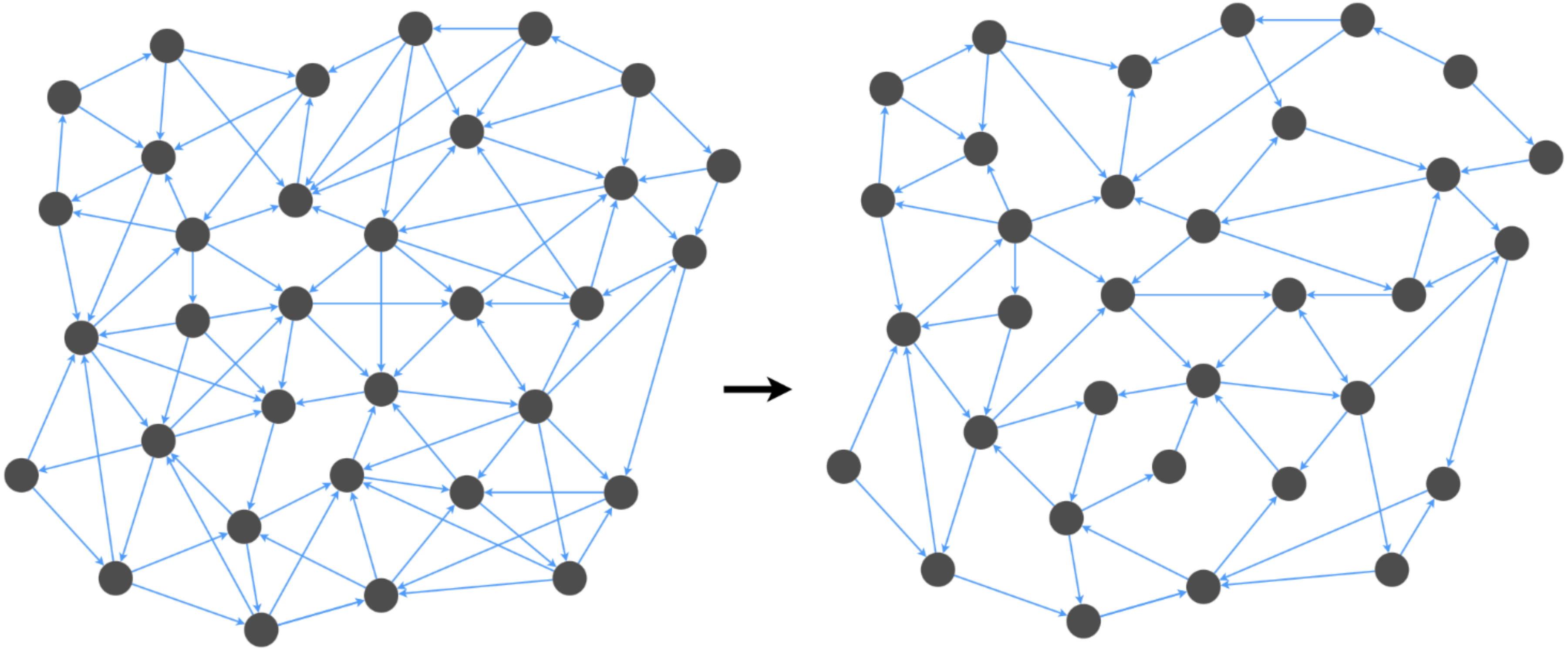
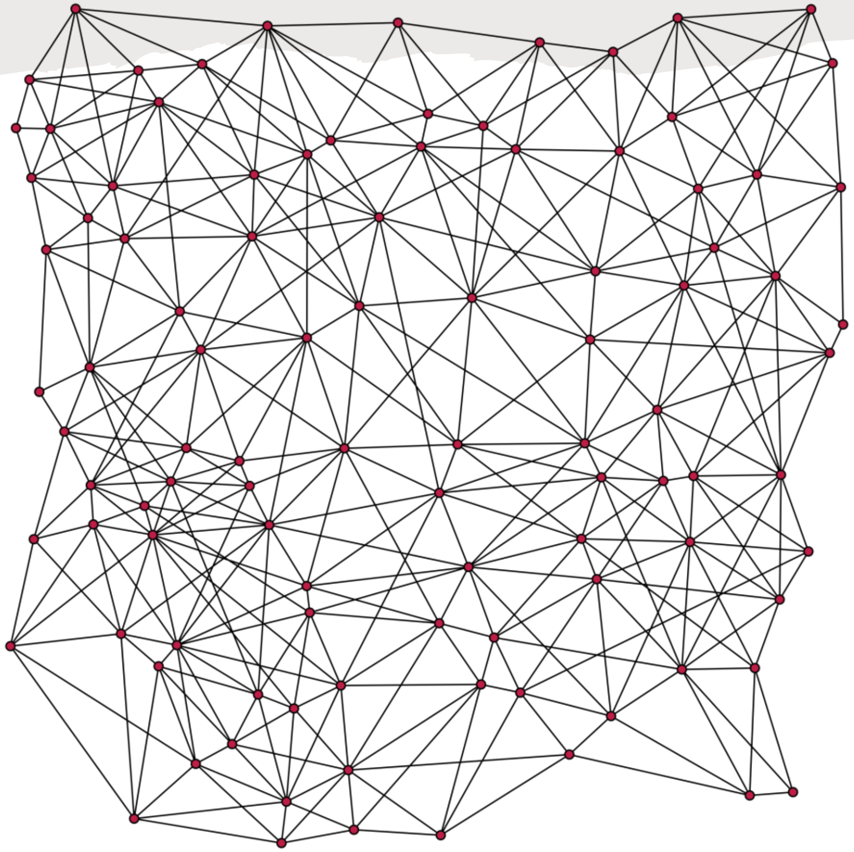


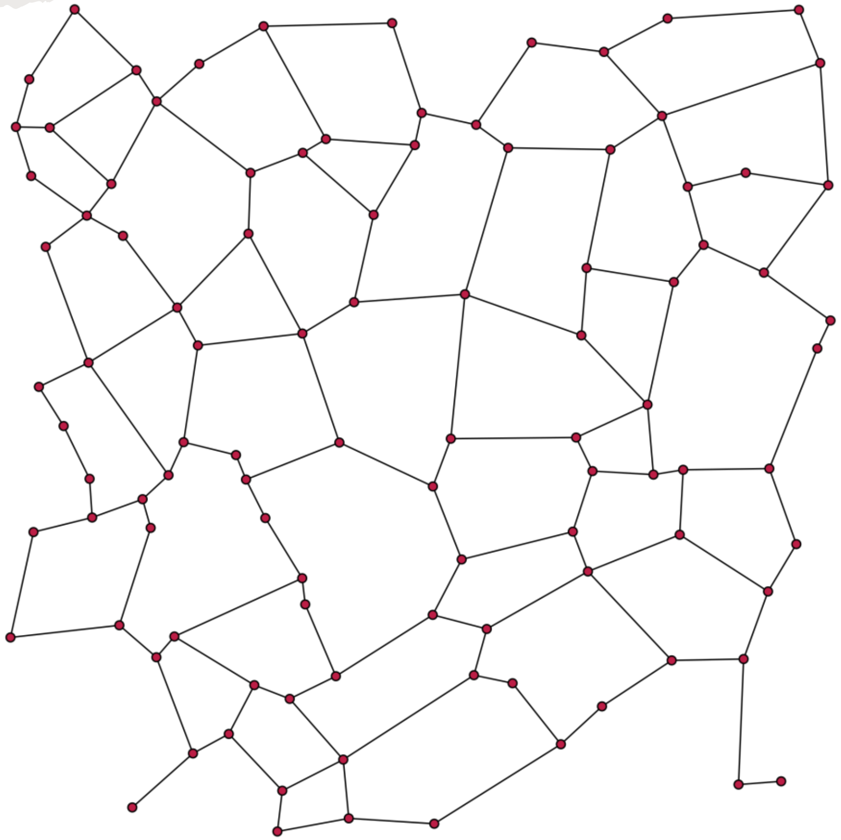
Image source : https://randorithms.com/2021/04/12/directed-spanners.html

# K - Spanner

- Given an input Graph G (V, E) then a Graph Spanner (H) is a sub-graph of the Graph G such that $H \subseteq G$.
- k - Spanner can be defined as for all edges $(u, v) \in G$, the Spanner graph 'H' contains a path between *u* and *v* of length no greater than $k.d(u, v)$
- A Graph Spanner preserves lengths of the shortest paths in G.

# Example of k - spanner



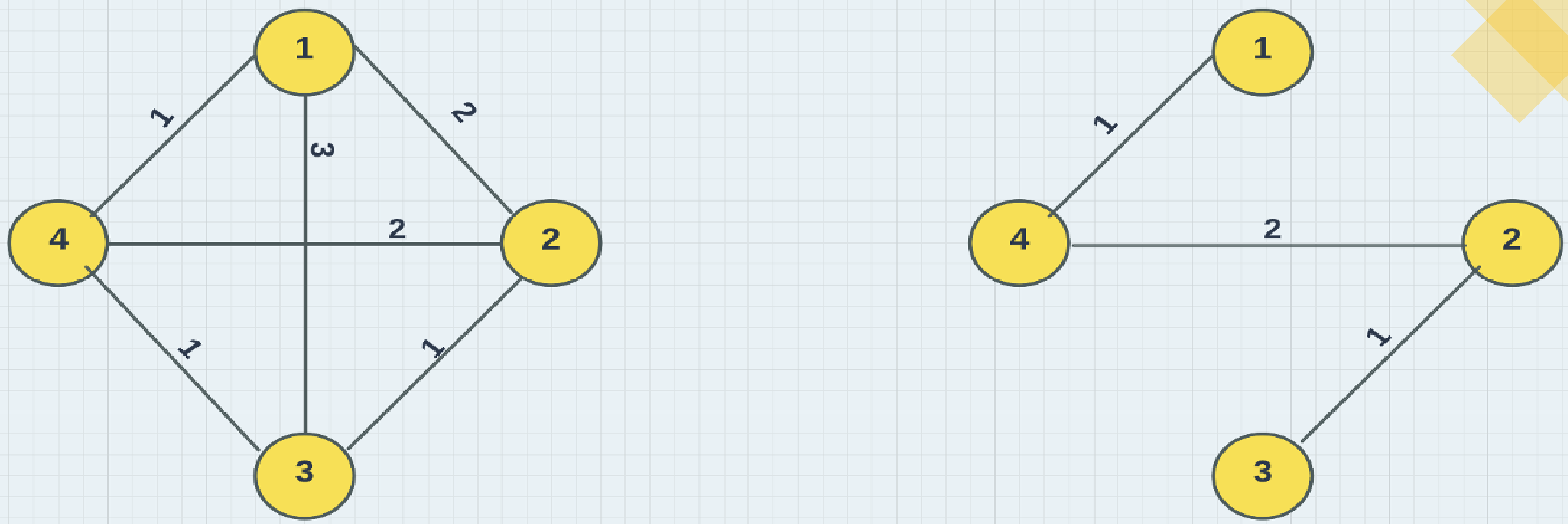


Stretch factor when k = 1 and k = 2

Image source : https://en.wikipedia.org/wiki/Greedy_geometric_spanner

# K - Spanner

Suppose if k = 3, then the subgraph H of G has at most 3 times the distance of that in G.




$$\begin{matrix} 0 & 2 & 3 & 1 \\ 2 & 0 & 1 & 2 \\ 3 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 \end{matrix} \qquad \begin{matrix} 0 & 3 & 4 & 1 \\ 3 & 0 & 1 & 2 \\ 4 & 1 & 0 & 3 \\ 1 & 2 & 3 & 0 \end{matrix}$$

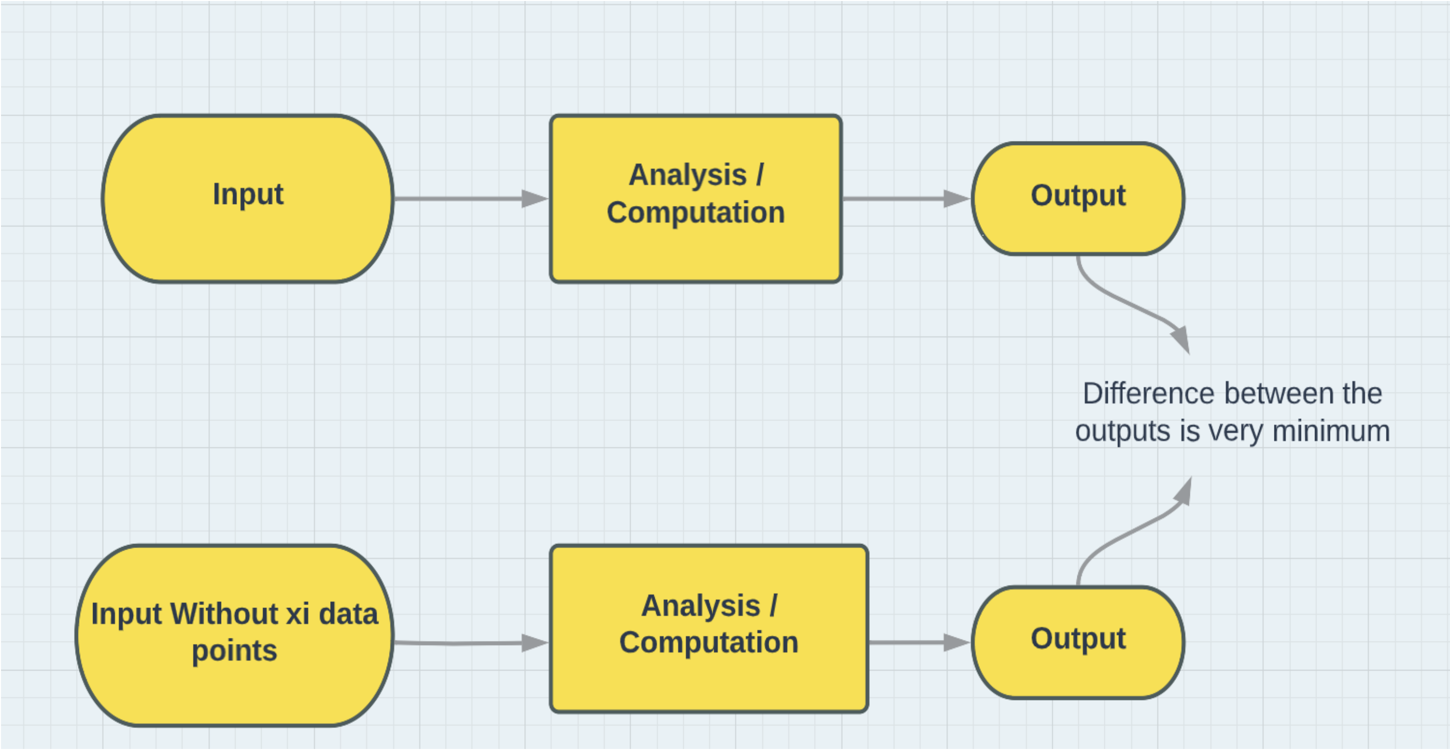
# Is there a Need for Privacy?

- Privacy enables us to **Create and Manage barriers/boundaries** to protect the network from unwanted interference.
- Privacy is a key for **Protecting the Data** to ensure the Individual's safety (Examples include Banking Data, Medical Data and so on…).
- With increase in massive data analysis, notion of privacy is under focus over highly sensitive user information.

# Previous related work

## Differential Privacy:

- Initial work on the privacy preservation is by Differential Privacy.
- In this algorithm the output does not change significantly with altering the some data points in the data.



- Implemented on large datasets
- This technique was employed by various companies like Apple, Facebook, Google and many more.
- Implemented on Medical Data, Employee Data etc.

# Our Initial Problem

Given a set of edges P (Private edges) such that $P \subset E$ in a Graph G (V, E) and an integer $k > 1$, we need to construct a $(2k-1)$-spanner H $(V, F \subseteq E)$ of the Graph G such that $F \cap P = \phi$

# Can we Avoid Private Data completely while building the model for our problem?

- Is it possible to construct a spanner while excluding all the private edges?
- May be difficult in some scenarios
- Try to relax the above objective and construct a spanner with fewest possible Private Edges

# Problem formulation

So, we can now formulate the problem from the Initial Problem

- Given a set of edges $P \subset E$ in a Graph $G(V, E)$ and an integer $k > 1$, we need to construct a Spanner $H (V, F \subseteq E)$ of the Graph $G$ while minimizing the objective function of $|F \cap P|$.

# Linear Programming

- To solve the above said problem, Our strategy is to formulate them as Linear Programming (LP) problems.
- Implementing Pulp Package in Python for Linear Programming.
- LP method in used in construction of optimizing the objective function.

# Linear Programming

- Linear Programming is a method to achieve the best outcome like Maximum profit or Lower cost to a model which has it’s representations by linear relationships.
- Linear programming is widely implemented in the field of optimization.
- Special cases such as Network Flow Problems can be solved by linear programming.

- $y_p = 1\, if\ path\ is\ selected\ to\ included\ in\ the\ spanner$ and 0 otherwise.
- $x_e$ = 1 if edge is included in the spanner and 0 otherwise
- $p_e$ = 1 if edge is a private edge and 0 otherwise. (known in advance)
- $a_e^p$ = 1 if edge e is on path p and 0 otherwise
- $P_{u,v}$ be the set of all paths of distance at most k · d(u, v) between u and v.

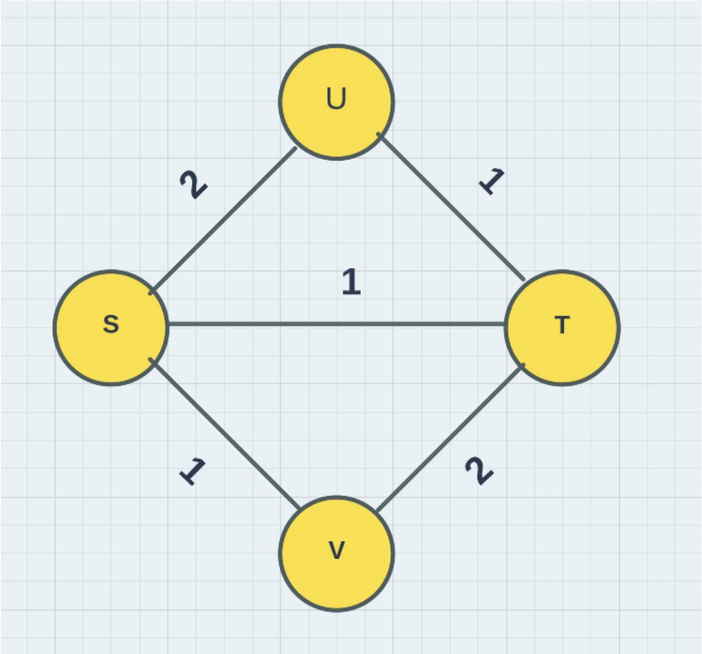
## $a_p^e$ table for the below graph:

| | <S, U> | <U, T> | <S, T> | <S, V> | <V, T> |
|---|---|---|---|---|---|
| P1 <S, U, T> | 1 | 1 | 0 | 0 | 0 |
| P2 <S, T> | 0 | 0 | 1 | 0 | 0 |
| P3 <S, V, T> | 0 | 0 | 0 | 1 | 1 |

Considering k = 3 in this example

$a_p^e = 1$ if edge e is on the path p else 0.

Distance P1 <S, U, T> = 3
Distance P2 <S, T> = 1
Distance P3 <S, V, T> = 3



# Main Objective Of the Linear Program

$$\min \sum_{e \in E} x_e p_e$$

- A model is constructed for minimizing the number of private edges selected to be included in the spanner (thus reducing information leak) for a given network.

- $x_e p_e$ are the number of private edges included in this spanner

# Constraints in the linear program

$$s.t. \sum_{p \in P_{u,v}} a_p^e y_p \leq x_e, \forall e \in E, \forall u, v \in V \times V$$

$$\sum_{p \in P_{u,v}} y_p \geq 1, \forall u, v \in V \times V$$

- 1st constraint: if a path is included in the spanner, all its edges are also included.
- 2nd constraint: For every node pair (constructing a spanner from source to target) there is at least a path P with distance at most k times the distance included in the spanner.
- Relax every pair to some pair in the experiments for computational saving.

# Datasets Implemented for analysis

- Data Sets from Stanford SNAP project (https://snap.stanford.edu)
- Facebook Dataset (88k edges and 4k node)
- EPINIONS Dataset (508k edges and 75k nodes)

# Performance Measures
# (Model Construction and Solving time)

- Objective Function
  - The number of private edges included in the constructed spanner
- LP Model Construction Time
  - The time to construct a linear program by adding constraints
- LP Model Solving Time
  - The time of solving the linear program

# Default Parameter Setting (Experiment A)

- Considering 5 (s, t) pairs : Source – Target nodes are selected in random.
- Edge Weights are not Considered (That is equal edge weights for all edges in the graph).
- 0.1 percent of edges of the graph are initially selected as private edges.
- K = 2 (Spanner constructed should have distance max ‘k’ times of d(u, v)).
- Number pairs = 1 (Number of (s, t) pairs selected to include in the spanner).
- Early stop= True (Parameter set to only included minimum number of paths of all the available paths for the node pairs selected to obtain sub-optimal solution and so we early stop at a certain level).

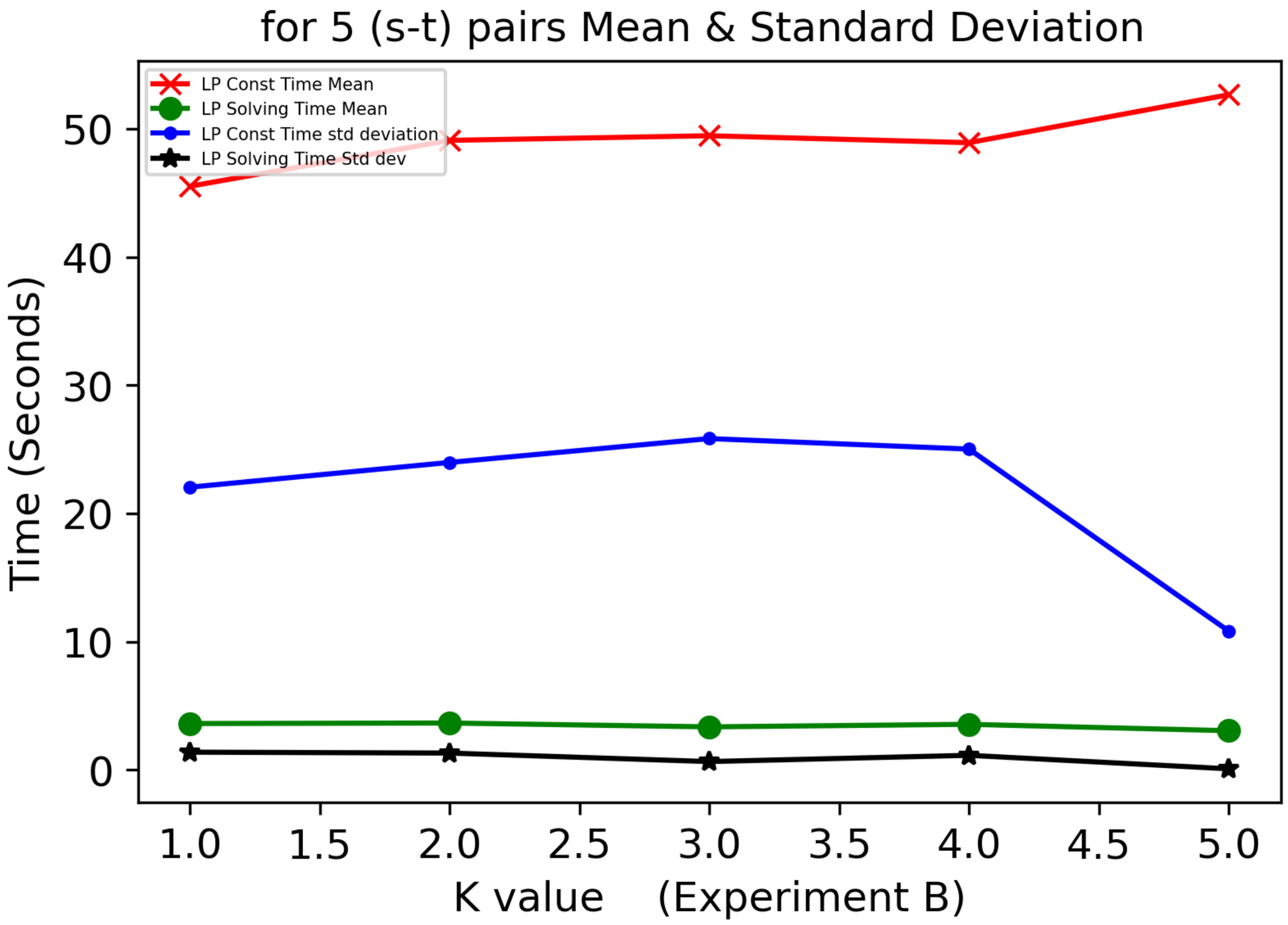
# Experiment A results (Facebook Dataset)

Table: Facebook Dataset for 5 (s-t) pairs (for Default Setting)

| **s – t pairs** | **Construction time** | **Solving Time** | **Number of Private edges included** |
|---|---|---|---|
| s=2139, t=1359, distance=4 | 53.3 Seconds | 3.8 Seconds | 0 |
| s=3405, t=2882, distance=3 | 56.5 Seconds | 3.9 Seconds | 0 |
| s=2304, t=1961, distance=2 | 59.0 Seconds | 3.9 Seconds | 0 |
| s=3240, t=383, distance=7 | 53.3 Seconds | 3.8 Seconds | 0 |
| s=3297, t=415, distance=6 | 59.5 Seconds | 3.9 Seconds | 0 |
| Range (in Seconds) | 56.32 $\pm$ 2.98 Seconds | 3.86 $\pm$ 0.054 Seconds | |

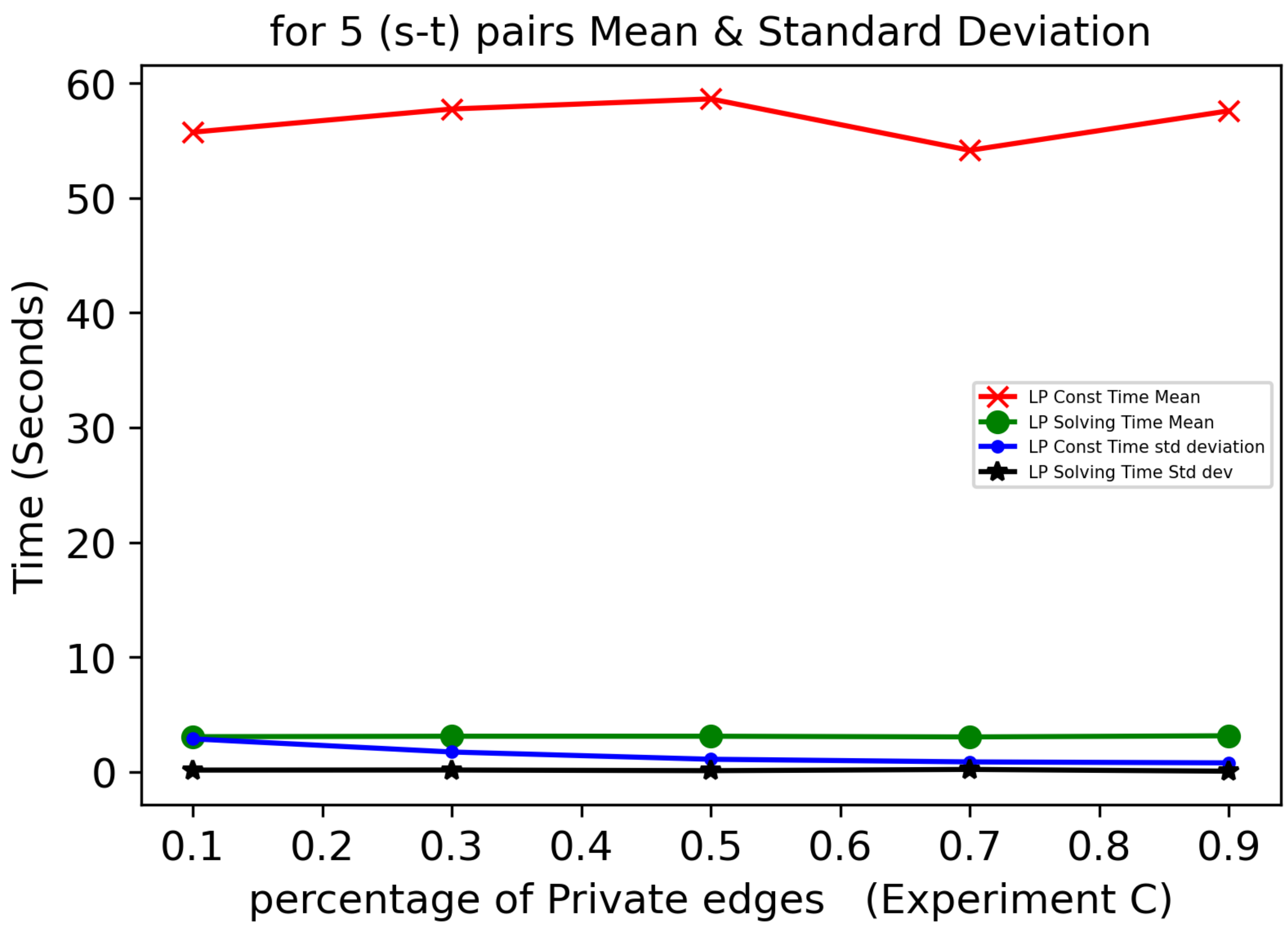
# Experiment B results (Facebook Dataset)

- Experiment B: Changing k = [1, 2, 3, 4, 5] while keeping other parameters constant from the default experiment.
- Mean and Standard Deviations of the Model Construction time and the Model Solving time for the Considered 5 (s-t) pairs.



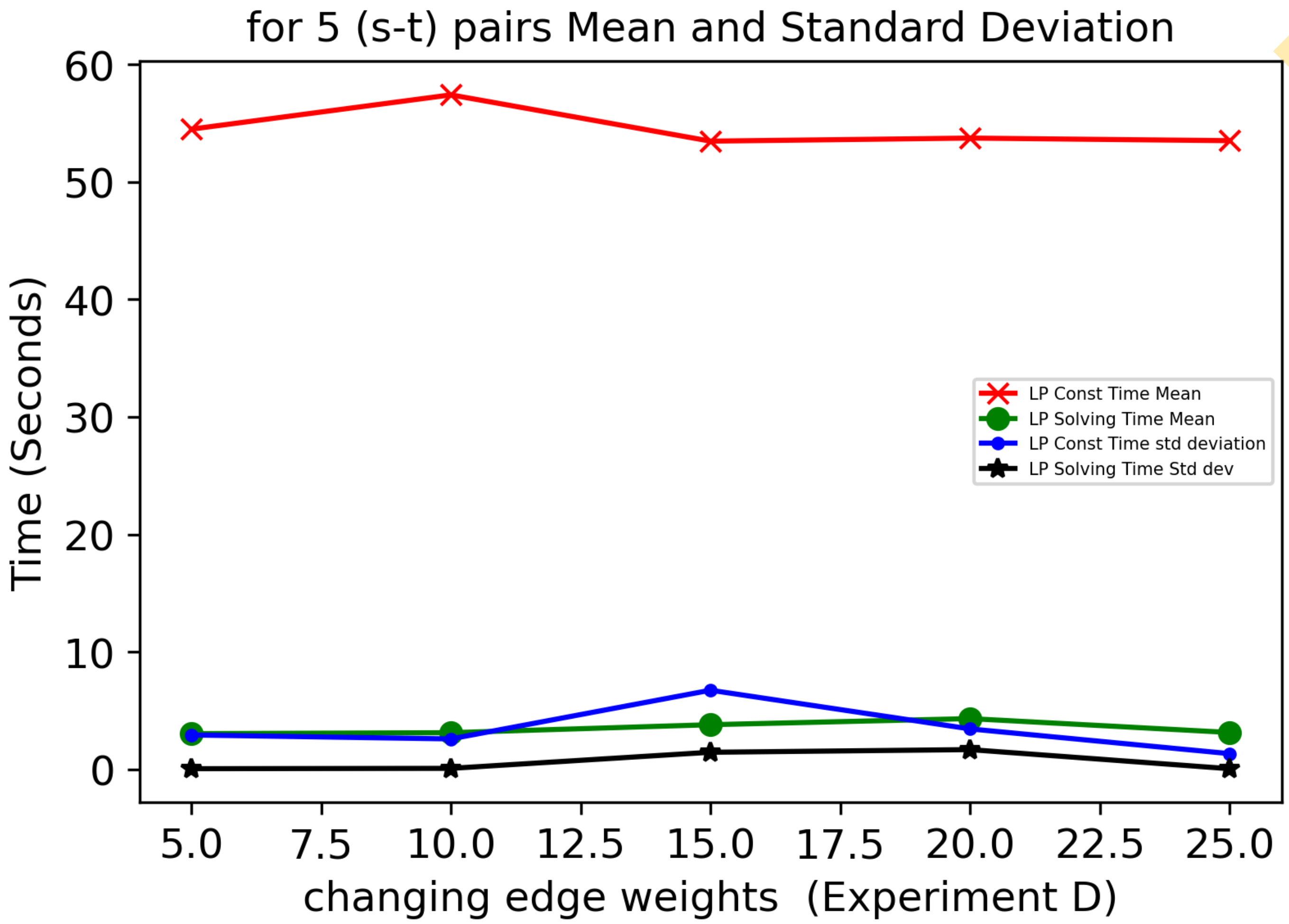
# Experiment C results (Facebook Datatset)

- Experiment C : Changing Percentage of private edges PercP = [0.1, 0.3, 0.5, 0.7, 0.9]



# Experiment D results (Facebook Dataset)

- Experiment D : Changing Edge Weights from False to True



# Experiment E results

- Our method: We implement early stop
- Baseline method : Without early stopping

| s = 2139, t = 1359, distance = 4 | **Construction time** | | **Solving time** | | **Private Edges Included** | |
|---|---|---|---|---|---|---|
| | **Ours** | **Baseline** | **Ours** | **Baseline** | **Ours** | **Baseline** |
| When k = 1.2 | 15.0 | 15.1 | 2.9 | 2.9 | 0 | 0 |
| When k = 1.4 | 58.0 | 303.2 | 2.9 | 3.1 | 0 | 0 |

| s = 3405, t = 2882, distance = 3 | **Construction time** | | **Solving time** | | **Private Edges Included** | |
|---|---|---|---|---|---|---|
| | **Ours** | **Baseline** | **Ours** | **Baseline** | **Ours** | **Baseline** |
| K = 1.2 | 14.7 | 15.0 | 2.9 | 2.9 | 0 | 0 |
| K = 1.4 | 37.1 | 34.8 | 2.9 | 2.8 | 0 | 0 |

| s = 2304, t = 1961, distance = 2 | **Construction time** | | **Solving time** | | **Private Edges Included** | |
|---|---|---|---|---|---|---|
| | **Ours** | **Baseline** | **Ours** | **Baseline** | **Ours** | **Baseline** |
| K = 1.2 | 7.7 | 8.1 | 6.1 | 5.9 | 0 | 0 |
| K = 1.4 | 8.5 | 7.8 | 5.9 | 6.3 | 0 | 0 |

| s = 3240, t = 383, distance = 7 | **Construction time** | | **Solving time** | | **Private Edges Included** | |
|---|---|---|---|---|---|---|
| | **Ours** | **Baseline** | **Ours** | **Baseline** | **Ours** | **Baseline** |
| K = 1.2 | 59.0 | 4159.1 | 2.9 | 3.0 | 1 | 0 |
| K = 1.4 | 59.4 | - | 3.3 | - | 0 | - |

| s = 3297, t = 415, distance = 6 | **Construction time** | | **Solving time** | | **Private Edges Included** | |
|---|---|---|---|---|---|---|
| | **Ours** | **Baseline** | **Ours** | **Baseline** | **Ours** | **Baseline** |
| K = 1.2 | 59.0 | 1866.7 | 2.9 | 3.1 | 0 | 0 |
| K = 1.4 | 57.9 | - | 3.0 | - | 0 | - |

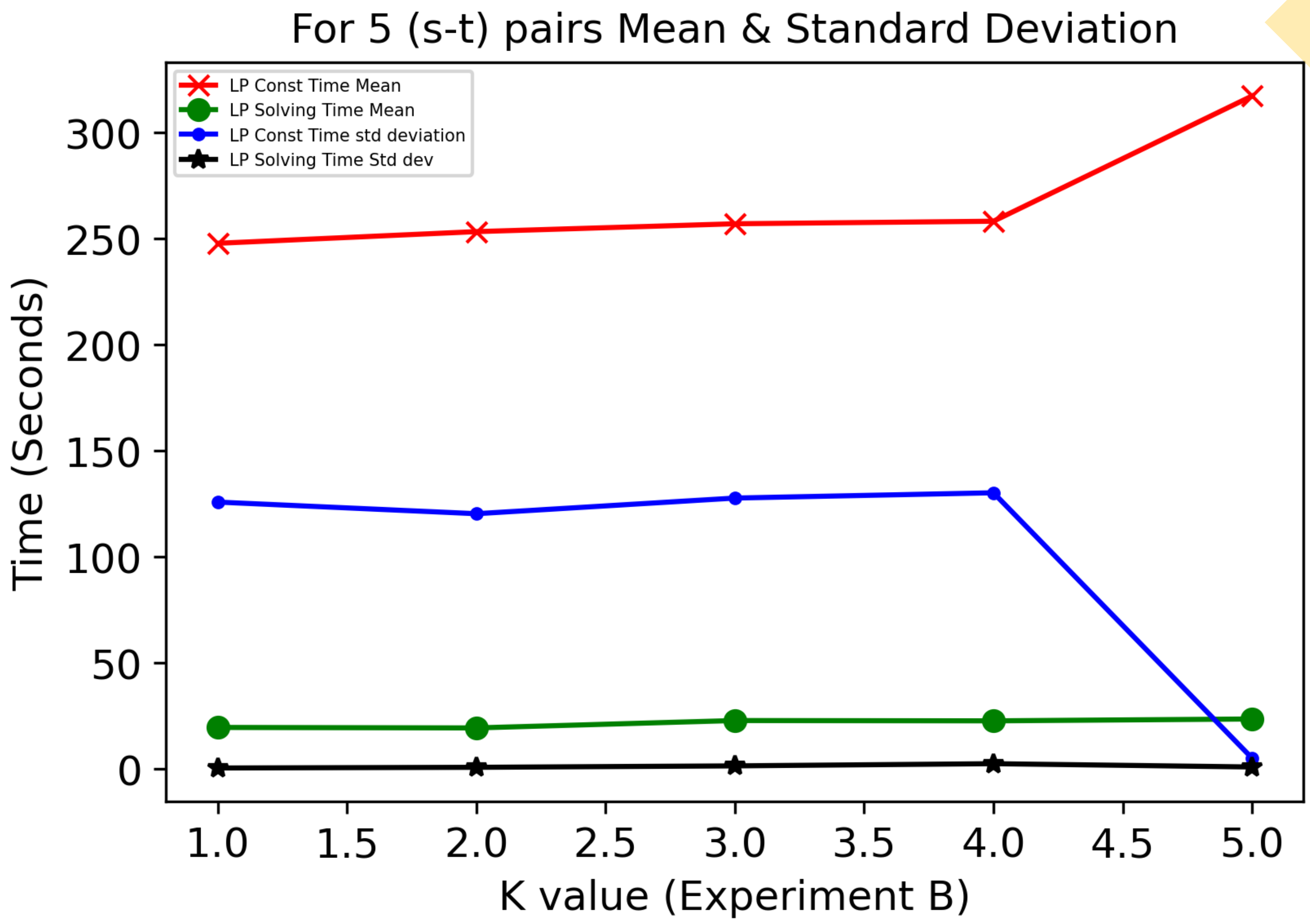
# Experiment A  Results (EPINIONS Dataset)

- Table: EPINIONS Dataset for 5 (s-t) pairs.

| s – t pairs | Construction time | Solving Time | Number of Private edges included |
|---|---|---|---|
| s=48722, t=723, distance=3 | 311.1 Seconds | 19.3 Seconds | 0 |
| s=15605, t=11914, distance=4 | 303.5 Seconds | 19.4 Seconds | 0 |
| s=42615, t=9901, distance=5 | 316.2 Seconds | 20.2 Seconds | 0 |
| s=12365, t=14160, distance=3 | 316.0 Seconds | 19.9 Seconds | 0 |
| s=54096, t=69623, distance=8 | 314.2 Seconds | 24.1 Seconds | 0 |
| Range (in Seconds) | 312.2 $\pm$ 5.2 Seconds | 20.58 $\pm$ 2 Seconds | |

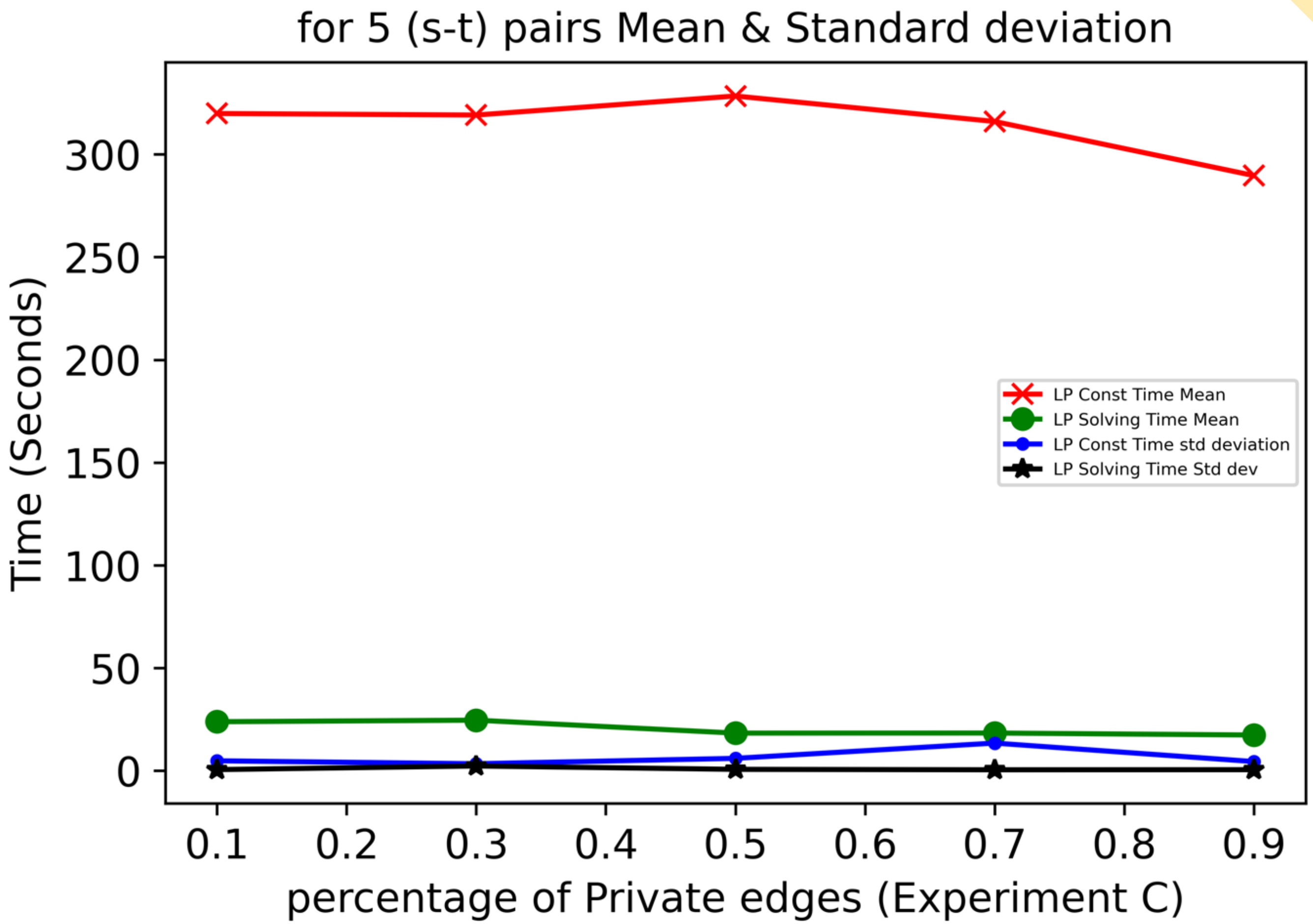
# Experiment B (EPINIONS Dataset)

- Changing k = [1, 2, 3, 4, 5] while keeping other parameters constant



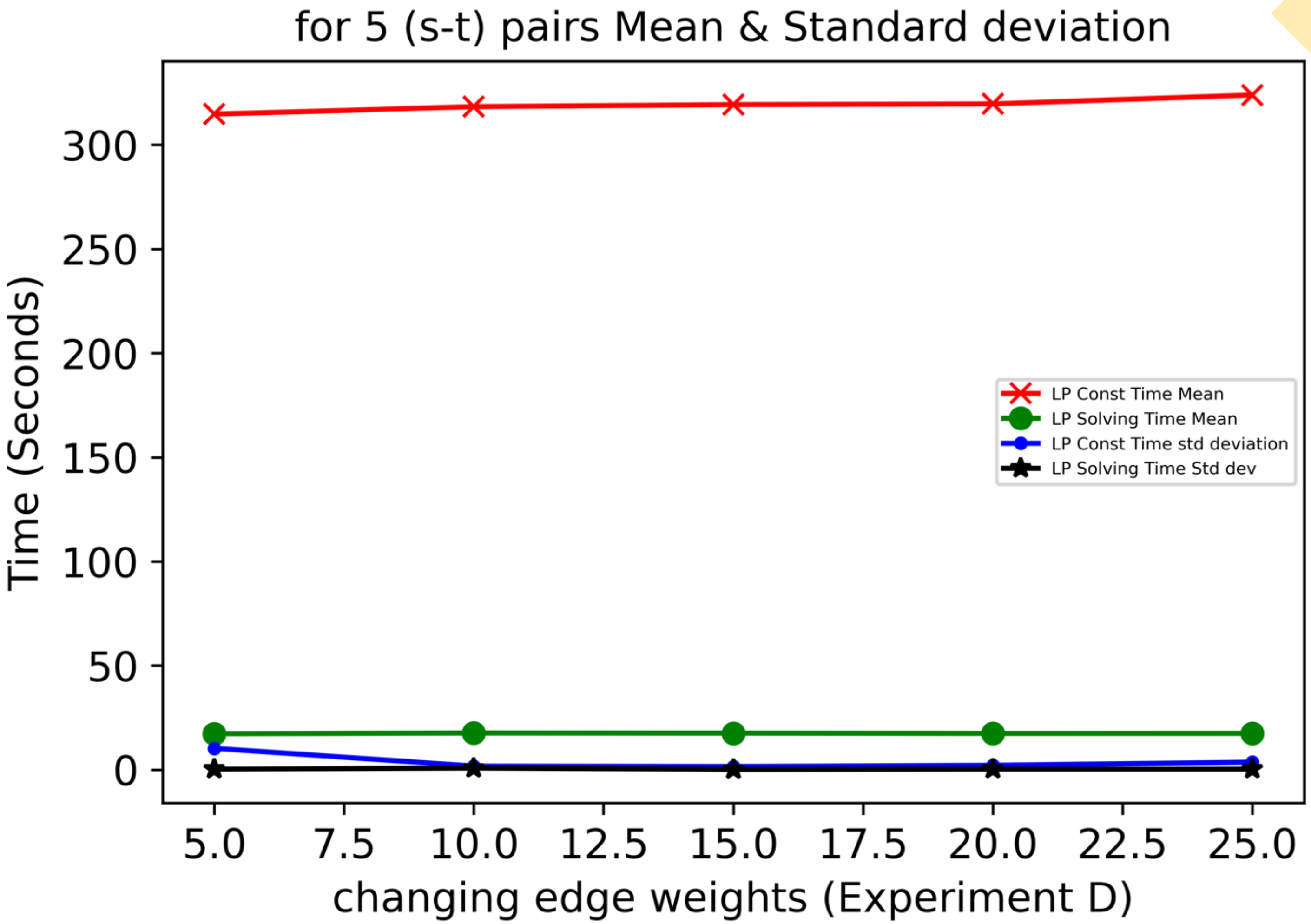
# Experiment C (EPINIONS Dataset)

- Experiment C : Changing Percentage of private edges PercP = [0.1, 0.3, 0.5, 0.7, 0.9]



# Experiment D (EPINIONS Dataset)

- Experiment D : Changing Edge Weights from False to True



# Experiment E results

- Our method: We implement early stop
- Baseline method : Without early stopping

| s = 12365, t = 14160, distance = 3 | Construction time | | Solving time | | Private Edges Included | |
|---|---|---|---|---|---|---|
| | **Ours** | **Baseline** | **Ours** | **Baseline** | **Ours** | **Baseline** |
| When k = 1.2 | 21.6 | 22.0 | 16.6 | 17.6 | 0 | 0 |
| When k = 1.4 | 125.9 | 127.0 | 16.8 | 17.9 | 0 | 0 |

| s = 15605, t = 11914, distance = 4 | Construction time | | Solving time | | Private Edges Included | |
|---|---|---|---|---|---|---|
| | **Ours** | **Baseline** | **Ours** | **Baseline** | **Ours** | **Baseline** |
| K = 1.2 | 33.9 | 33.6 | 18.0 | 14.9 | 0 | 0 |
| K = 1.4 | 329.4 | 1418.0 | 17.4 | 15.5 | 0 | 0 |

| s = 42615, t = 9901, distance = 5 | Construction time | | Solving time | | Private Edges Included | |
|---|---|---|---|---|---|---|
| | **Ours** | **Baseline** | **Ours** | **Baseline** | **Ours** | **Baseline** |
| K = 1.2 | 328.4 | 690.2 | 17.0 | 17.2 | 0 | 0 |
| K = 1.4 | 329.8 | - | 16.9 | - | 0 | 0 |

| s = 48722, t = 723, distance = 3 | Construction time | | Solving time | | Private Edges Included | |
|---|---|---|---|---|---|---|
| | **Ours** | **Baseline** | **Ours** | **Baseline** | **Ours** | **Baseline** |
| K = 1.2 | 22.2 | 22.6 | 17.0 | 16.8 | 0 | 1 |
| K = 1.4 | 326 | 762.7 | 17.0 | 17.3 | 1 | 0 |

| s = 54096, t = 69623, distance = 8 | Construction time | | Solving time | | Private Edges Included | |
|---|---|---|---|---|---|---|
| | **Ours** | **Baseline** | **Ours** | **Baseline** | **Ours** | **Baseline** |
| K = 1.2 | 326.1 | - | 16.8 | - | 0 | - |
| K = 1.4 | 324.7 | - | 16.9 | - | 0 | - |

# Conclusion

- In this project we implement the Objective function and get sub optimal solution in minimizing the private edges in a spanner.
- Early stopping is set as we have less computational power and time.
- Model performance is analyzed by several experiments.
- As the future work, we can try to implement this algorithm for all the s-t pairs in the graph.