

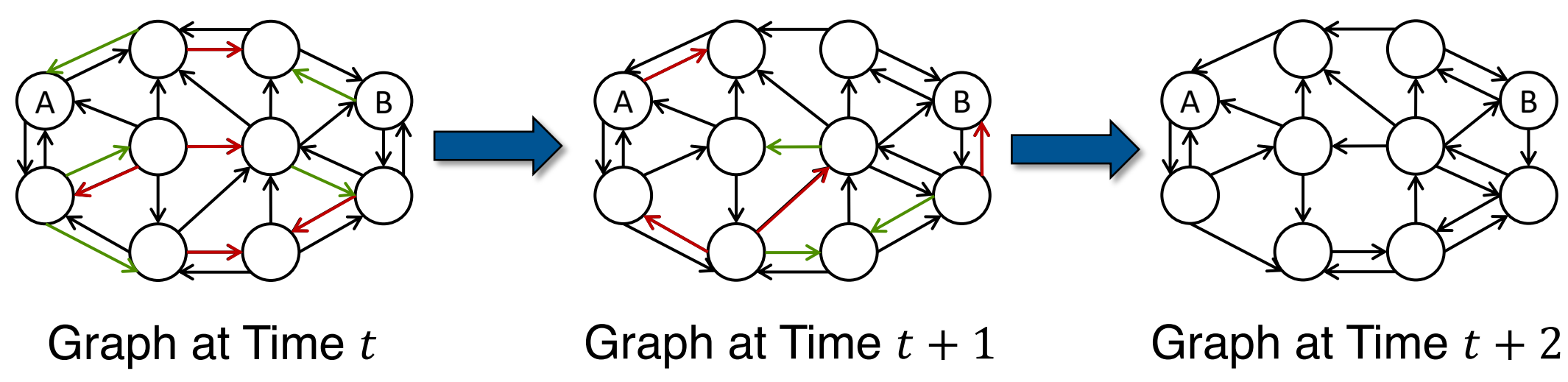
# CommonGraph: Graph Analytics on Evolving Data

Mahbod Afarin, Chao Gao, Shafiu Rahman, Nael Abu-Ghazaleh, Rajiv Gupta  
University of California, Riverside

Session 8C: Graphs B  
1:00-2:05 PM Wed

## 1. Background

- Dynamic Graph Systems
  - Streaming Graph Processing
  - Evolving Graph Processing

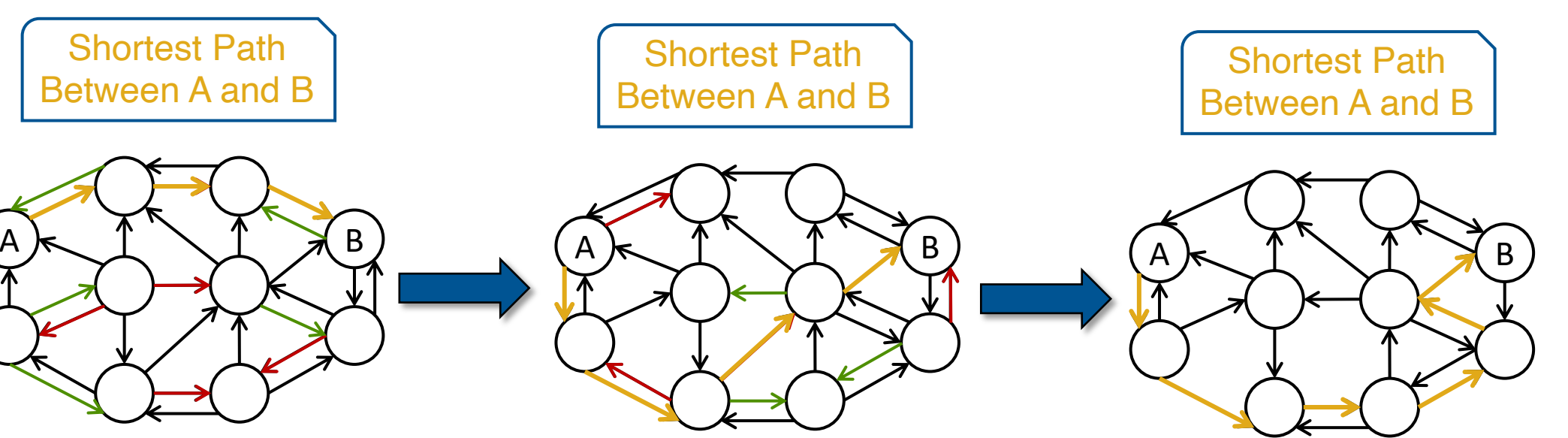


Graph at Time  $t$       Graph at Time  $t + 1$       Graph at Time  $t + 2$

Add 5 Edges  
Delete 5 Edges

Add 3 Edges  
Delete 4 Edges

Fig.1 Evolving graph processing for three different snapshots of a graph



Graph at Time  $t$       Graph at Time  $t + 1$       Graph at Time  $t + 2$

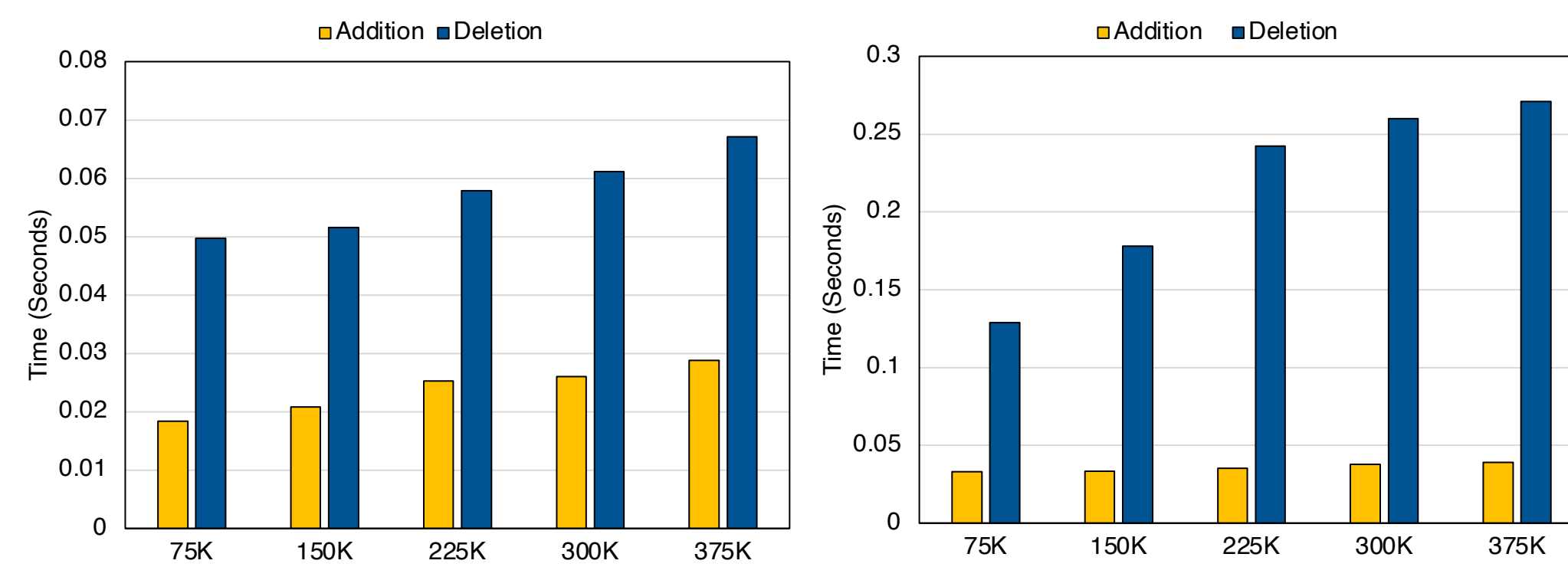
Add 5 Edges  
Delete 5 Edges

Add 3 Edges  
Delete 4 Edges

Fig.2 Shortest path will change when we add and delete some edges

## 2. Problem

- Approaches for query evaluation on Evolving Graphs
  - Naïve Approach
  - Incremental Approach
  - CommonGraph Approach
- Naïve approach is not efficient because  $\rightarrow$ 
  - Solve the query on each snapshots independently



Computation Cost of the Additions vs. Deletions in KickStarter for SSSP      Graph Mutation Cost of the Additions vs. Deletions in KickStarter

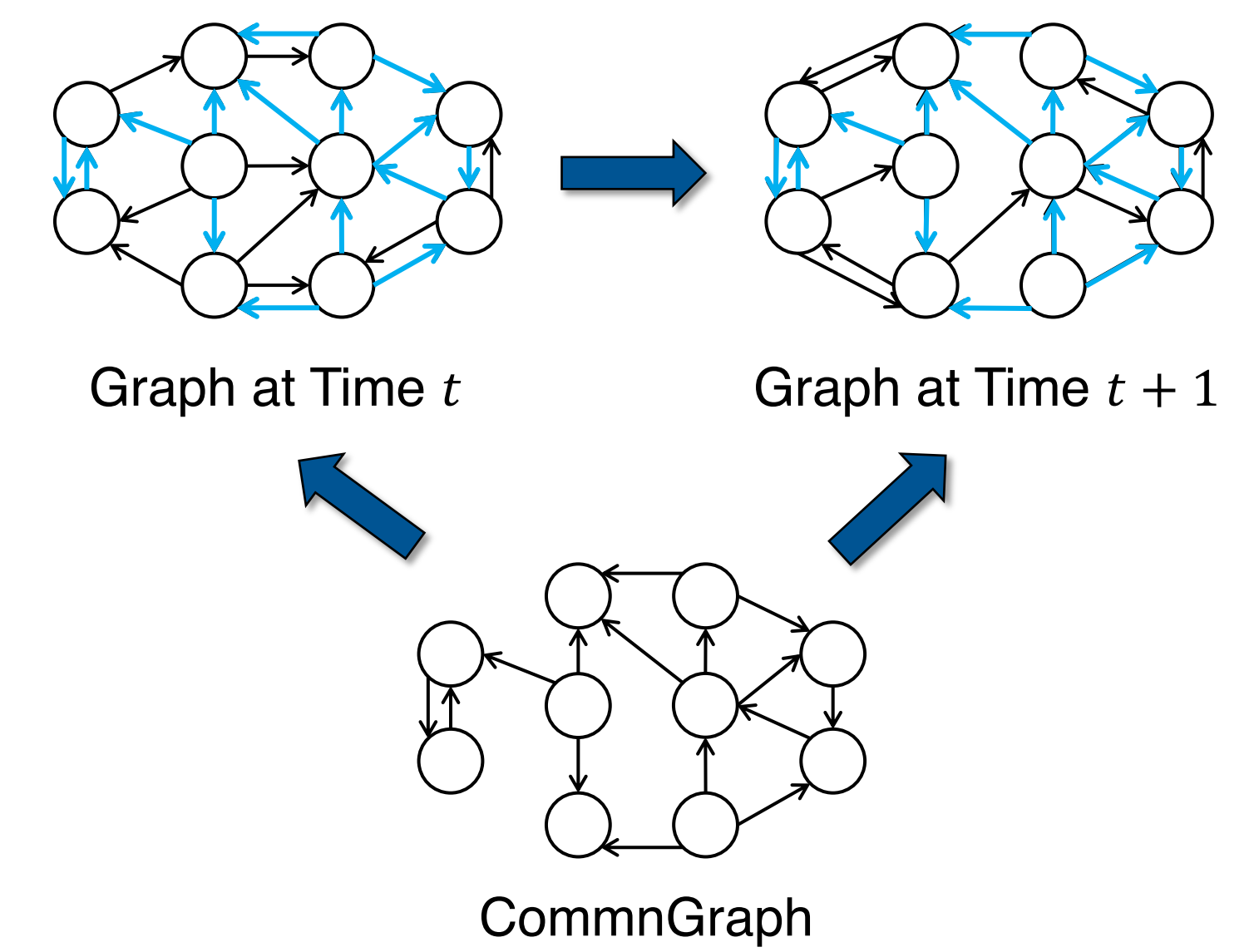
Fig.3 The mutation and computation cost of the addition vs. deletion

- Naïve approach is not efficient because  $\rightarrow$ 
  - Deletion operation is significantly more expensive than addition operations

## 3. Solution

Our Idea:

- Transform deletions to additions using CommonGraph
- Finding the common edges between snapshots
- CommonGraph Approach  $\rightarrow$ 
  - Solve the query on the CommonGraph
  - Add the missing edges and incrementally update the results



Graph at Time  $t$       Graph at Time  $t + 1$

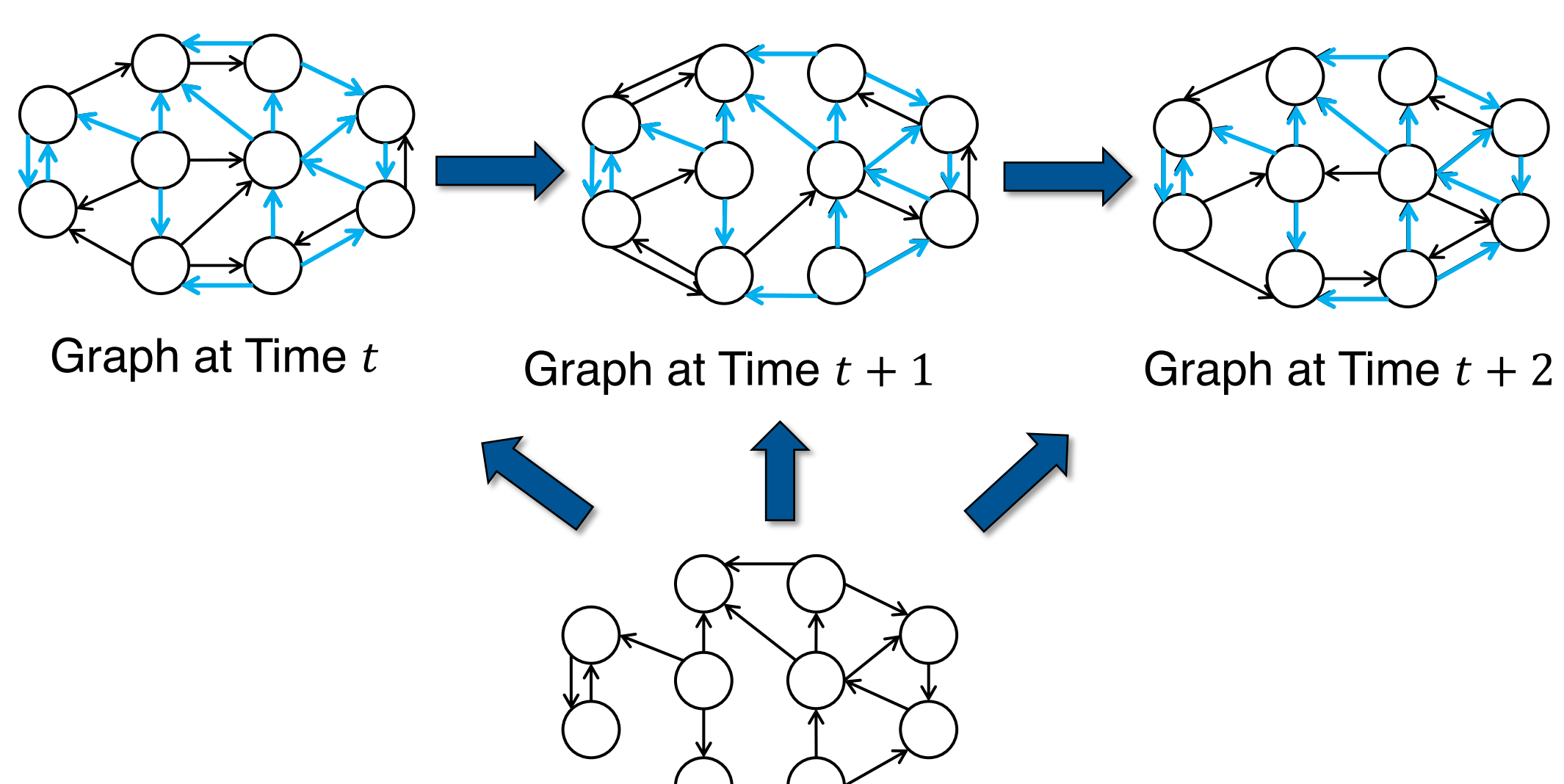
CommonGraph

Fig.4 Finding the CommonGraph between two snapshots of a graph

## 4. Direct Hop Algorithm

CommonGraph Query Evaluation:

- Direct Hop Query Evaluation
  - Find each snapshot directly from the CommonGraph
  - Higher number of Addition
  - Better performance comparing to conventional method with only additions



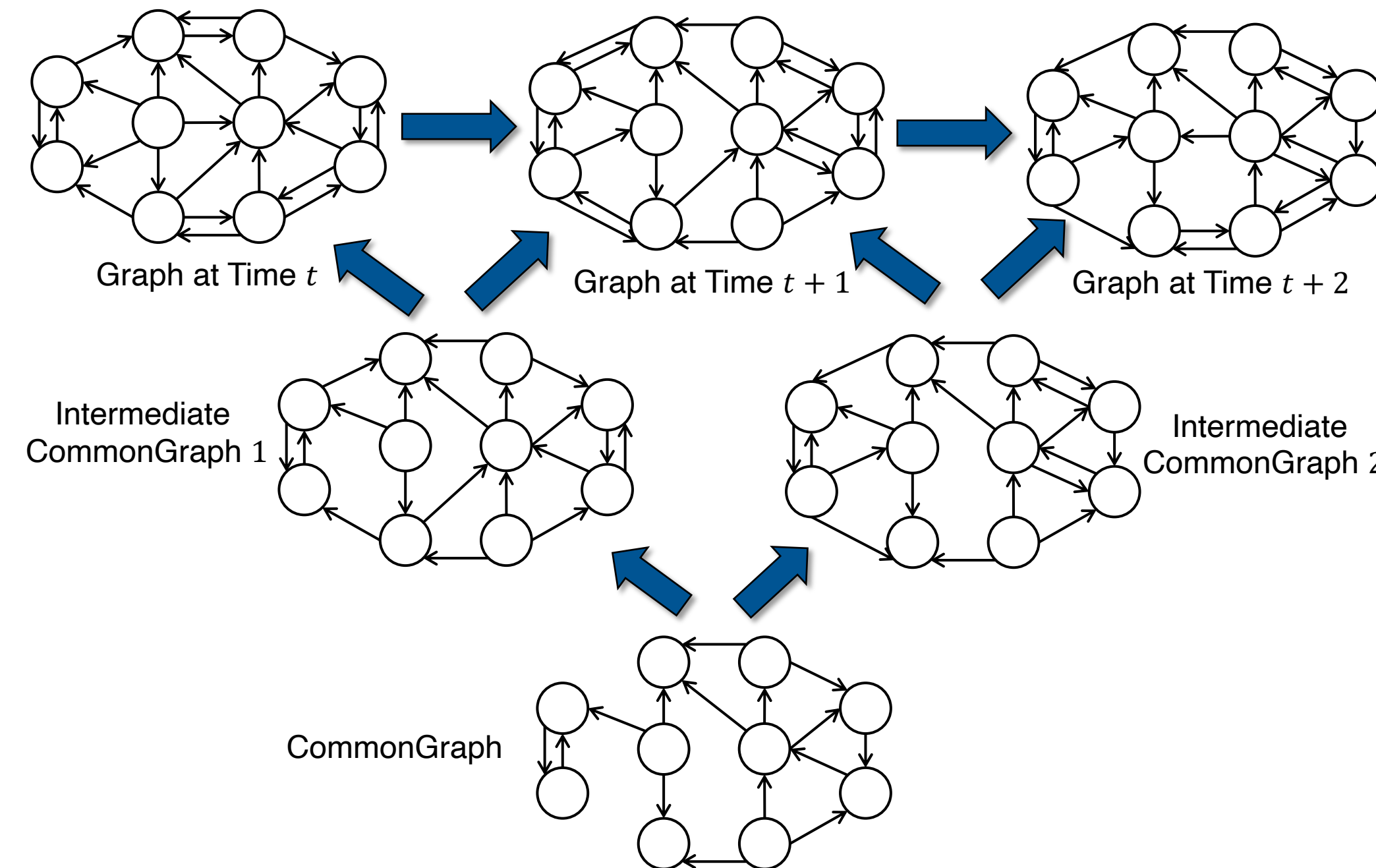
Graph at Time  $t$       Graph at Time  $t + 1$       Graph at Time  $t + 2$

Fig.5 CommonGraph Direct Hop approach for three snapshots

## 5. Work Sharing Algorithm

CommonGraph Query Evaluation:

- Work Sharing Query Evaluation
  - Find each snapshots from CommonGraph
  - Significantly lower number of additions
  - Better performance comparing to the Direct-Hop approach with less additions



Graph at Time  $t$       Graph at Time  $t + 1$       Graph at Time  $t + 2$

Intermediate CommonGraph 1      Intermediate CommonGraph 2

CommonGraph

Fig.6 CommonGraph Work Sharing approach for three snapshots

## 6. Scheduling

- Finding the best scheduling in Work-Sharing algorithm
  - Creating the Triangular Grid
  - Steiner Tree Algorithm

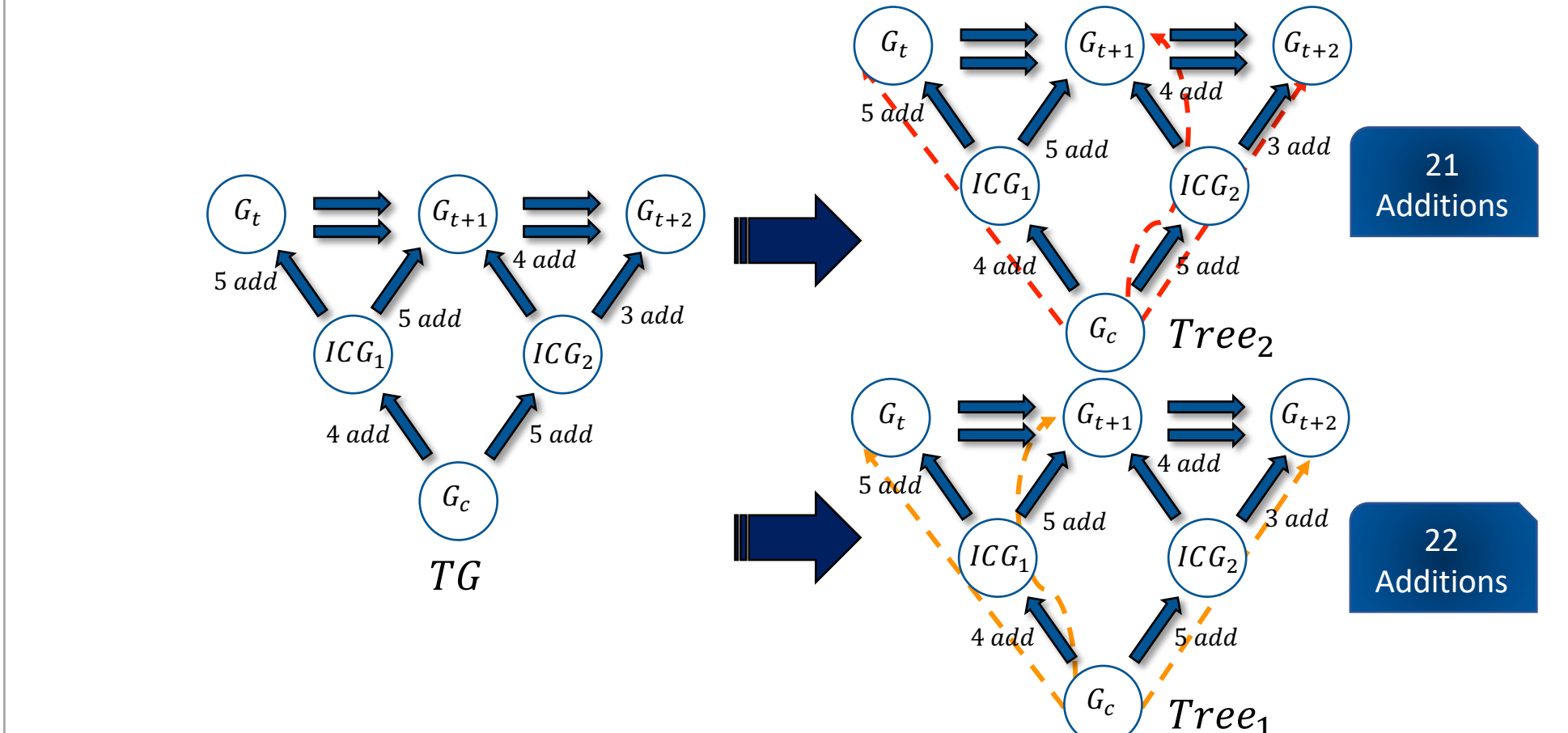


Fig.7 Two Trees Corresponding to Query Evaluation Schedules with Different Costs

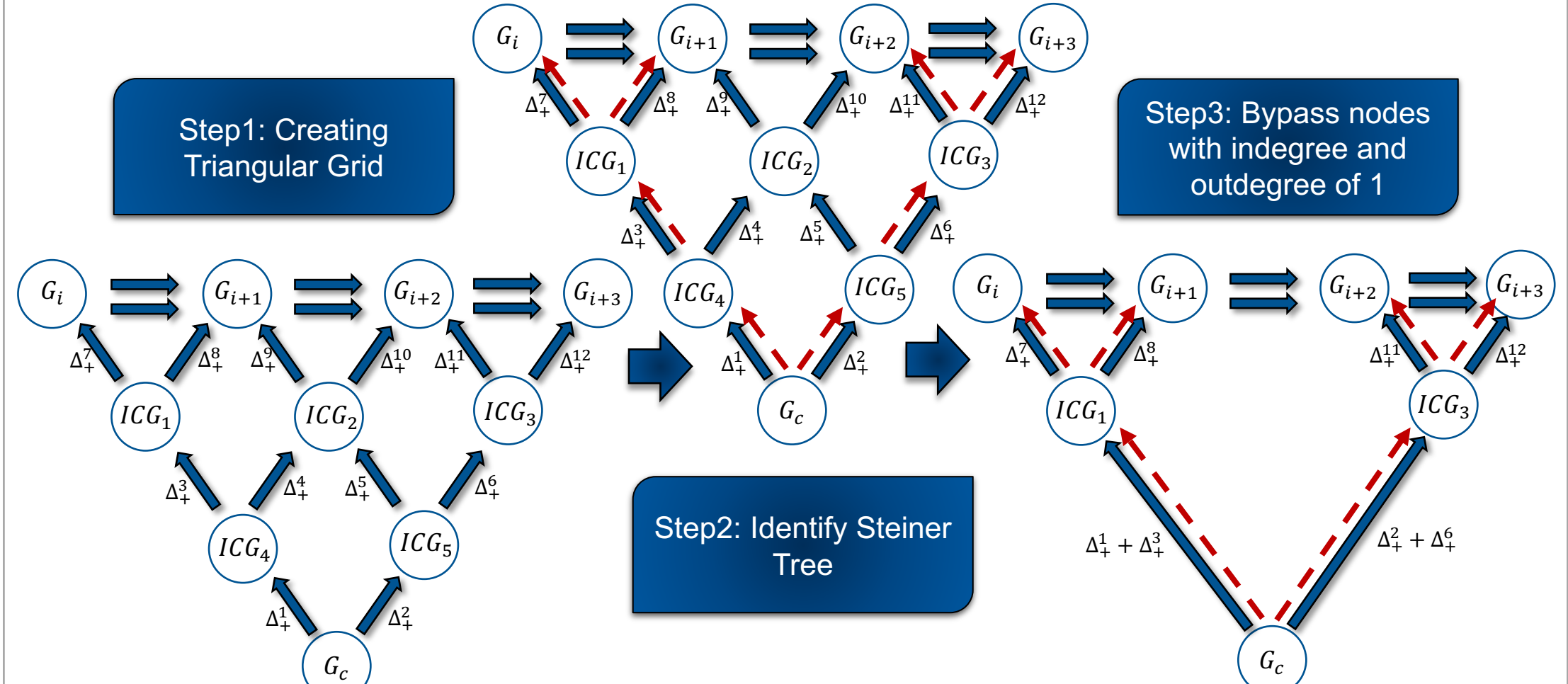


Fig.8 Algorithm for Identifying Minimum Cost Query Evaluation Schedule

## 7. CommonGraph System

- Evolving Graph Engine
  - Shared CommonGraph
  - Addition delta batches

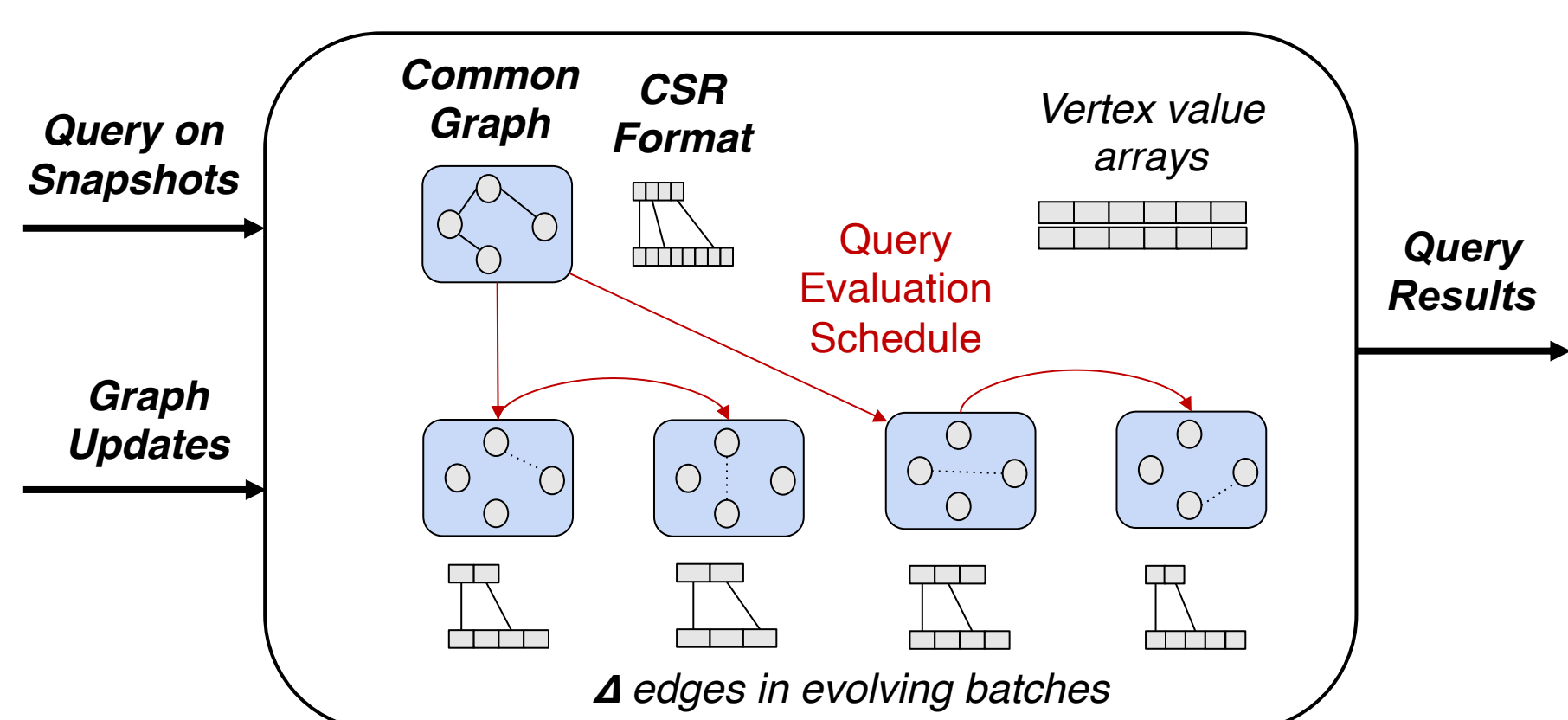


Fig.9 Evolving Graph Engine

- CommonGraph Primitives

Version control API	Description
get_version (number)	Retrieve a snapshot
diff (snapshot, snapshot)	Identifies difference between two snapshots
new_version ( $\Delta_+$ , $\Delta_-$ )	Create a new snapshot and update common graph

Query API	API function
edge_function (edge)	Algorithm specific edge function
schedule (vertex_id, mode)	Schedule active vertex
update (vertex_id)	Atomic update function

## 8. Evaluation

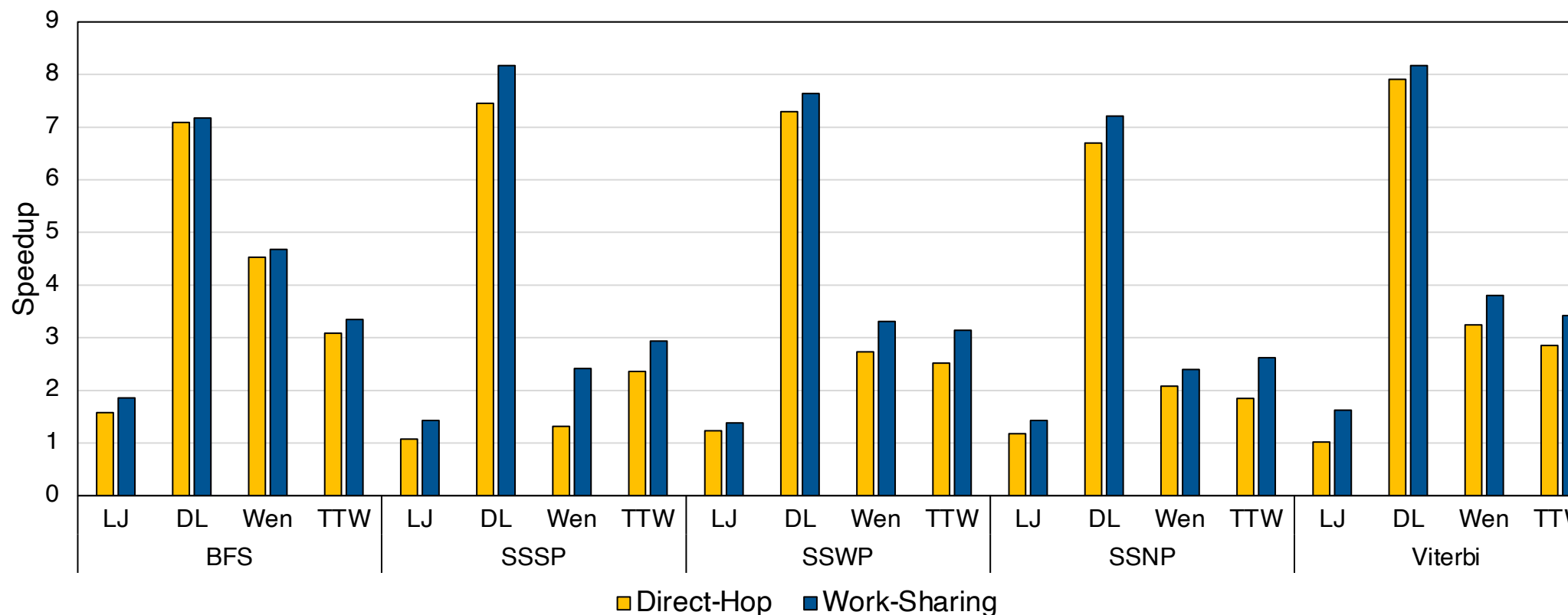
- Benchmarks

Algorithms	EdgeFunction ( $e(u, v)$ )
BFS	$CASMIN(Val(v), \min(Val(u) + 1, val(v)))$
SSWP	$CASMIN(Val(v), \min(Val(u), wt(u, v)))$
SSNP	$CASMIN(Val(v), \max(Val(u), wt(u, v)))$
SSSP	$CASMIN(Val(v), Val(u) + wt(u, v))$
Viterbi	$CASMIN(Val(v), Val(u)/wt(u, v))$

- Input Graphs

Input Graphs	Nodes	Edges	Avg degree
LiveJournal (LJ)	4M	70M	28.26
DBpediaLinks (DL)	18M	170M	18.85
WikipediaLinks (Wen)	13M	400M	64.32
Twitter (TTW)	41M	1.5B	70.51

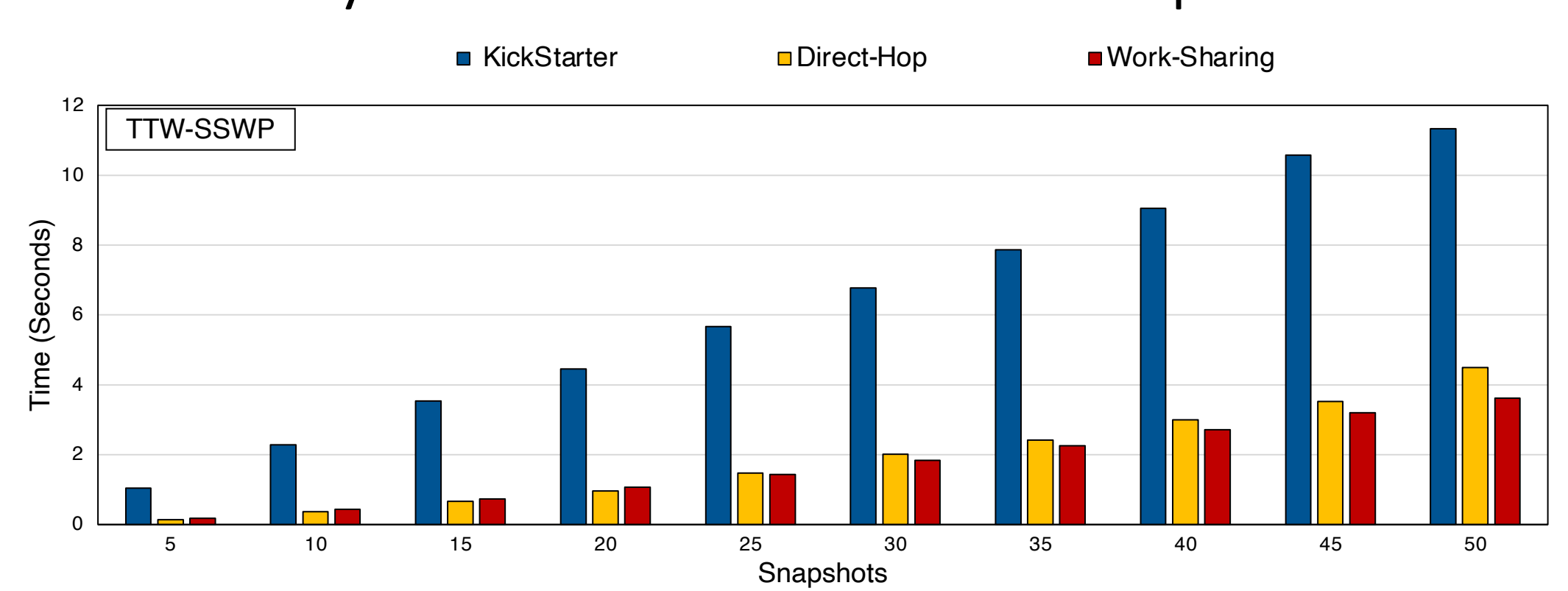
- Performance



CommonGraph achieves  $1.38 \times - 8.17 \times$  improvement in performance over Kickstarter across multiple benchmarks.

## 9. Sensitivity Analysis

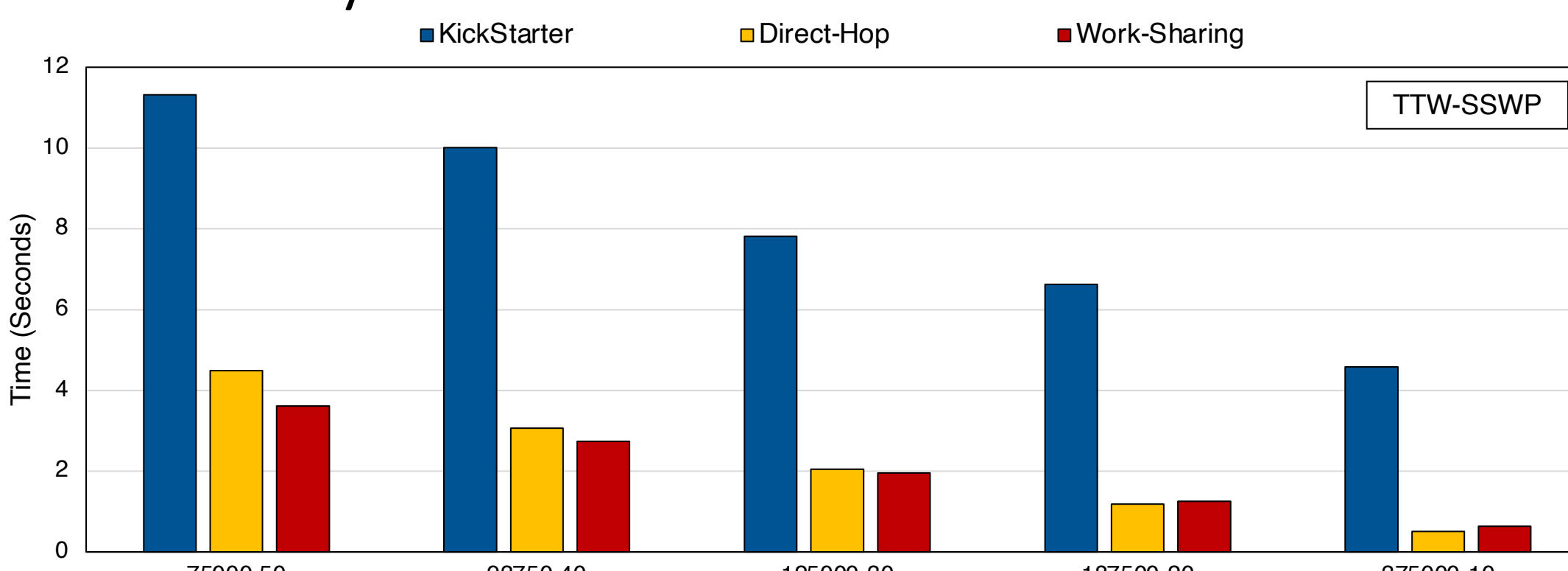
- Sensitivity to the Different Number of Snapshots



For the fewer number of snapshots, the Direct-Hop algorithm works better than Work-Sharing.

Work-Sharing outperforms Direct-Hop when we increase the number of snapshots beyond 23 to 35 for different benchmarks.

- Sensitivity to Batch Size



For the bigger batch size, the direct-hop algorithm works better compared to the work-sharing, and for the smaller number of the batch size, the work-sharing works better.