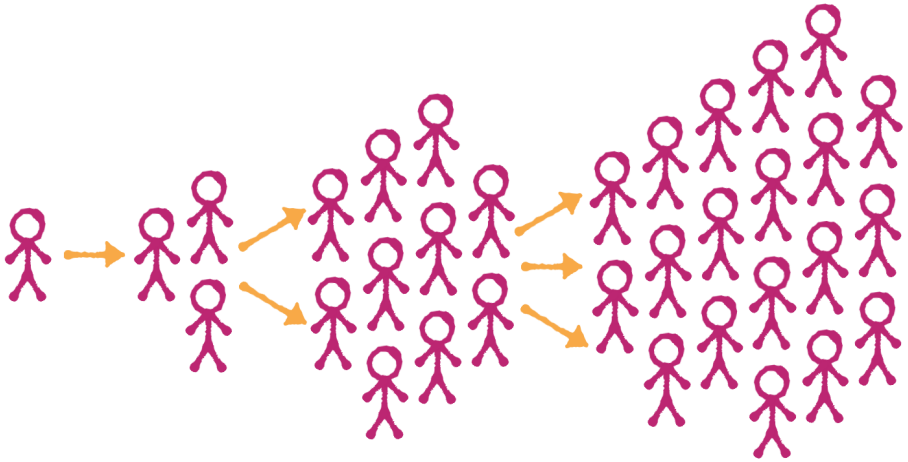
Finding Temporal Influential Users over Evolving Social Networks

Shixun Huang, Zhifeng Bao, J.Shane Culpepper and Bang Zhang

Introduction

Viral Marketing



Information Diffusion

http://multimediamarketing.com/mkc/viralmarketing/

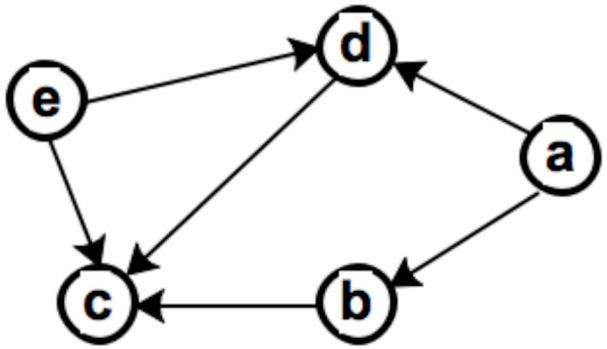
https://medium.com/the-megacool-blog/how-to-generate-word-of-mouth-buzz-for-your-mobile-game-50408e209df0

Introduction

Given (1) an integer k, (2) a diffusion model, the Influence Maximization (IM) problem aims to find a seed set of k target nodes that have the greatest influence spread in the network.

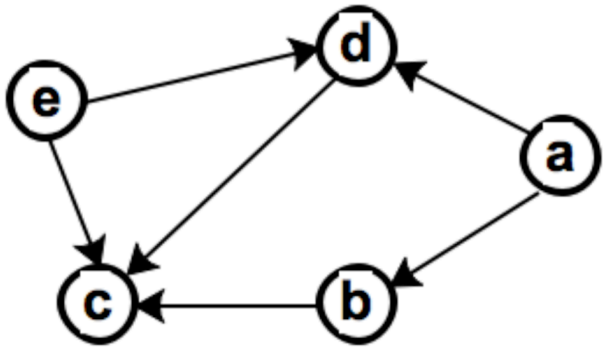
Introduction

Given (1) an integer k, (2) a diffusion model, the Influence Maximization (IM) problem aims to find a seed set of k target nodes that have the greatest influence spread in the network.



Introduction

Given (1) an integer k, (2) a diffusion model, the Influence Maximization (IM) problem aims to find a seed set of k target nodes that have the greatest influence spread in the network.



The IM problem is NP-hard and has two cases:

- The static case and dynamic case.

App: find influential users at a specific timestamp.

Introduction

Some limitations have not been considered in evolving networks:

1. Limited coverage of distinct users.

2. Difficulty of deploying personalized advertising messages.

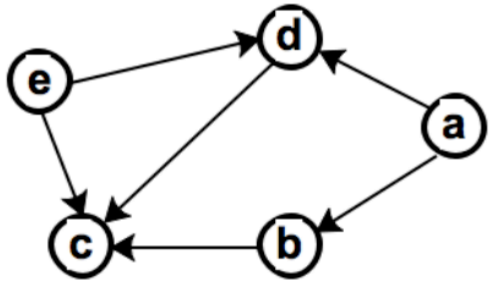
3. Difficulty of achieving effective user exposures to advertisements.

Introduction

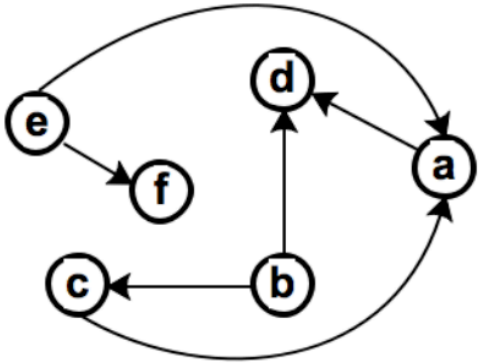
We study the **Distinct Influence Maximization** (DIM) problem to find a *fixed* seed set of k target users to maximize the expected number of distinct users influenced by the target users in an evolving social network.

Introduction

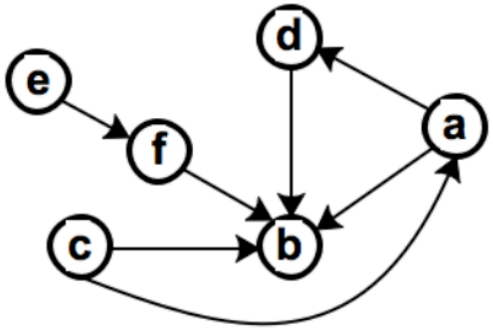
We study the **Distinct Influence Maximization** (DIM) problem to find a *fixed* seed set of k target users to maximize the expected number of distinct users influenced by the target users in an evolving social network.



Snapshot1



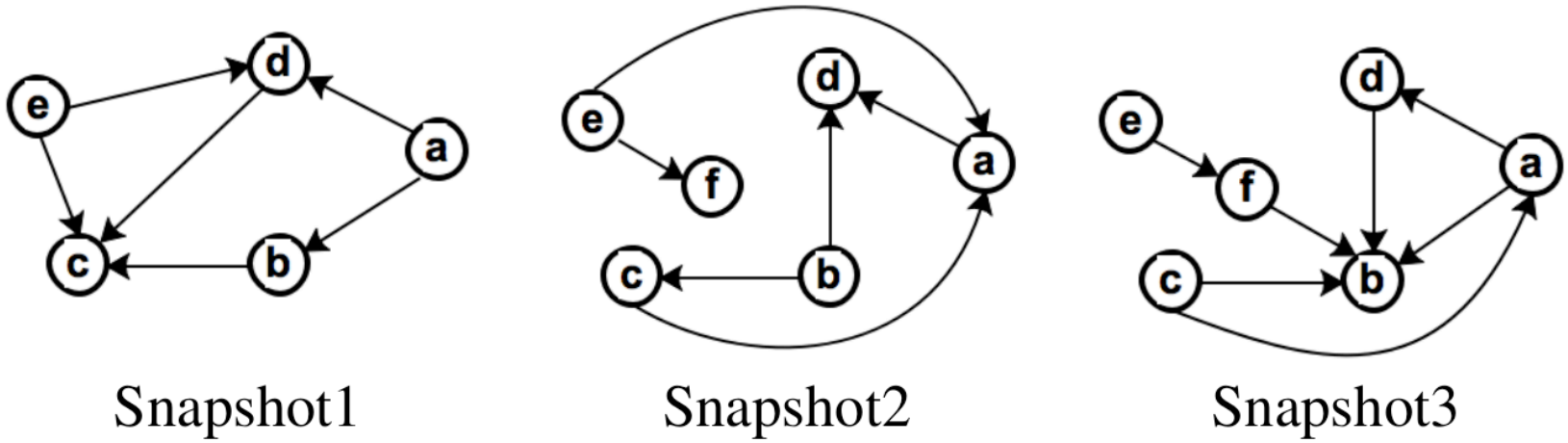
Snapshot2



Snapshot3

Introduction

We study the **Distinct Influence Maximization** (DIM) problem to find a *fixed* seed set of k target users to maximize the expected number of distinct users influenced by the target users in an evolving social network.



For finding the top-1 target users:

1.Previous studies: select users a, b or c in different snapshots.

2.Our solution: selects user e among all snapshots. (App: find influential users over a period.)

Overview of Our Solutions

We approximate distinct influence spread by averaging distinct reachability (via BFS) on the subgraphs via Monte-Carlo (MC) simulations.

Overview of Our Solutions

We approximate distinct influence spread by averaging distinct reachability (via BFS) on the subgraphs via Monte-Carlo (MC) simulations.

Our contributions are:

1. The quality of solutions is theoretically bounded.

2. We propose two compression techniques VCS and HCS.

3. Extensive experiments show that:

(1) for the DIM problem, our solutions significantly outperform baselines w.r.t. memory costs.

(2) for the IM problem, our solutions provide good trade-offs between running time and memory costs.

Preliminaries

1. The influence diffusion model – Independent Cascade (IC) model [1].

2. The greedy strategy with theoretical guarantees [2].
   Iteratively selects node with maximum marginal gain.

3. The subgraph strategy with theoretical guarantees [3].
   Keeps each edge (u,v) with prob as the normalized edge weight $p_{(u,v)}$.

[1] D. Kempe, et al. "Maximizing the spread of influence through a social network," in *SIGKDD*, 2003.

[2] G. L. Nemhauser, et al. "An analysis of approximations for maximizing submodular set functions," in Mathematical programming, 1978.

[3] N. Ohsaka, et al. "Fast and accurate influence maximization on large networks with pruned monte-carlo simulations," in *AAAI*, 2014.

Problem Formulation

Suppose we have:

1. A sequence of snapshots $D = \{G^1, G^2, \ldots, G^w\}$ ($G^i = (V^i, E^i)$ and $1 \leq i \leq w$)
2. A common node set $V_c = \bigcap_{i=1}^{w} V^i$.
3. A positive integer (budget) k.
4. $\zeta_D(S)$ denotes the distinct influence spread of S in D.

The **Distinct Influence Maximization** (DIM) problem aims to find a seed set of size k such that

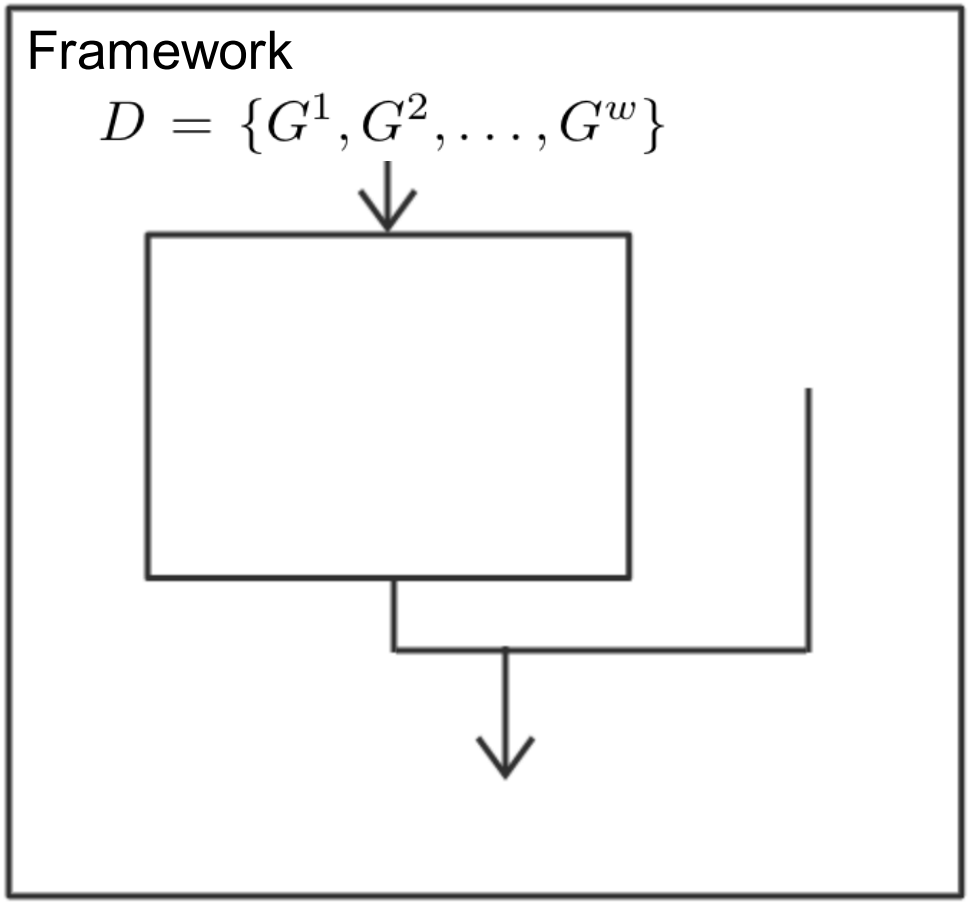
$$\zeta_D(S^*) = \arg\max_{S \subseteq V_c \wedge |S| \leq k} \zeta_D(S)$$

Our Solutions

We propose two methods HCS and VCS to efficiently compute $\zeta_D(S)$. (Averaging the distinct reachability on subgraphs generated from snapshots.)

Our Solutions

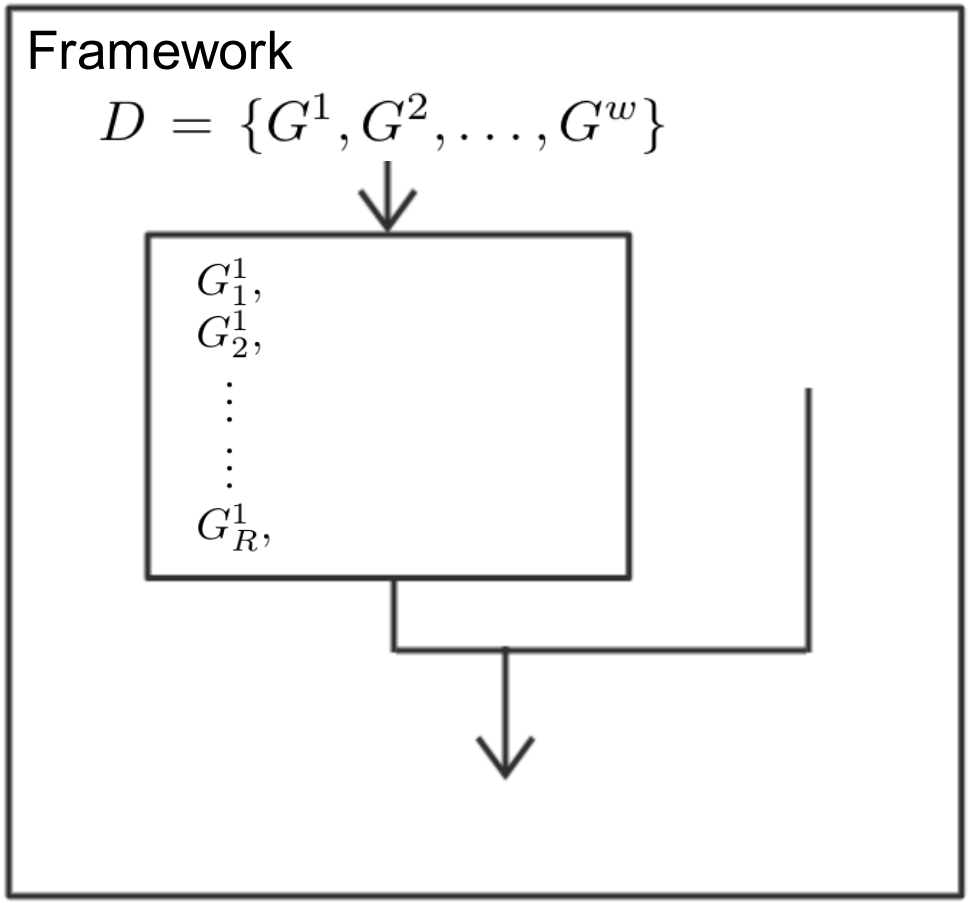
We propose two methods HCS and VCS to efficiently compute $\zeta_D(S)$. (Averaging the distinct reachability on subgraphs generated from snapshots.)



Our Solutions

We propose two methods HCS and VCS to efficiently compute $\zeta_D(S)$. (Averaging the distinct reachability on subgraphs generated from snapshots.)

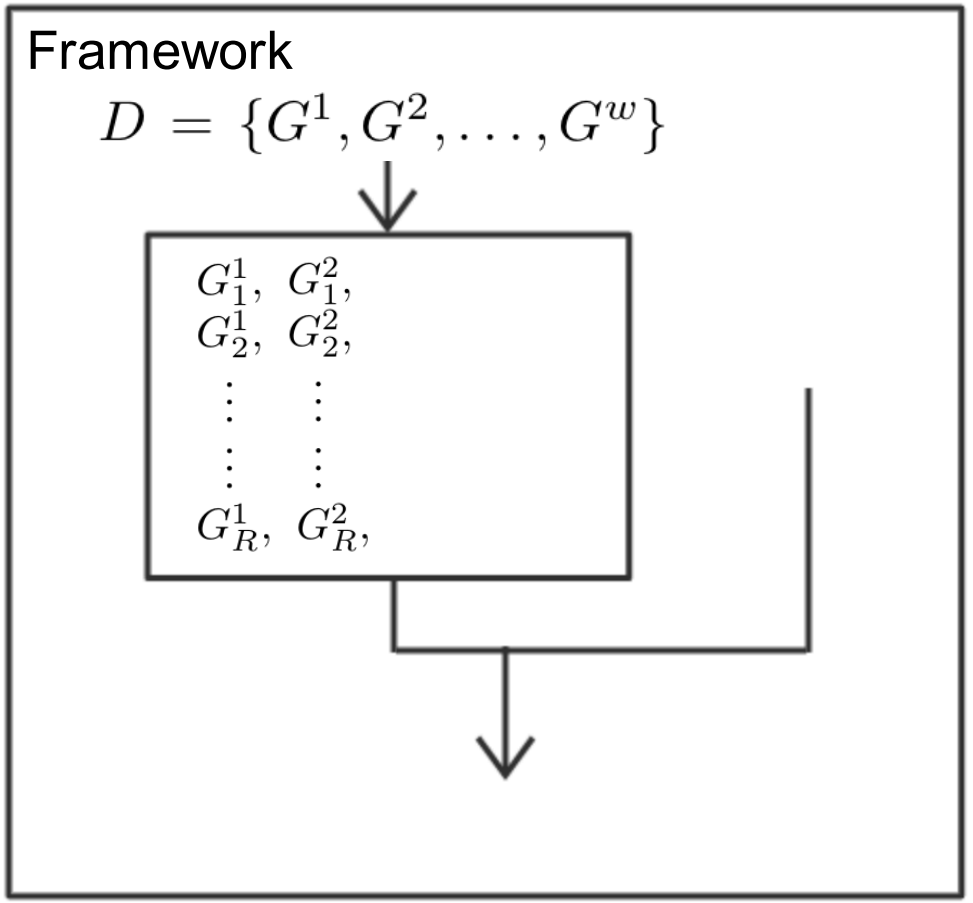
G^i_j denotes the *j-th* subgraph generated from G^i.



Our Solutions

We propose two methods HCS and VCS to efficiently compute $\zeta_D(S)$. (Averaging the distinct reachability on subgraphs generated from snapshots.)

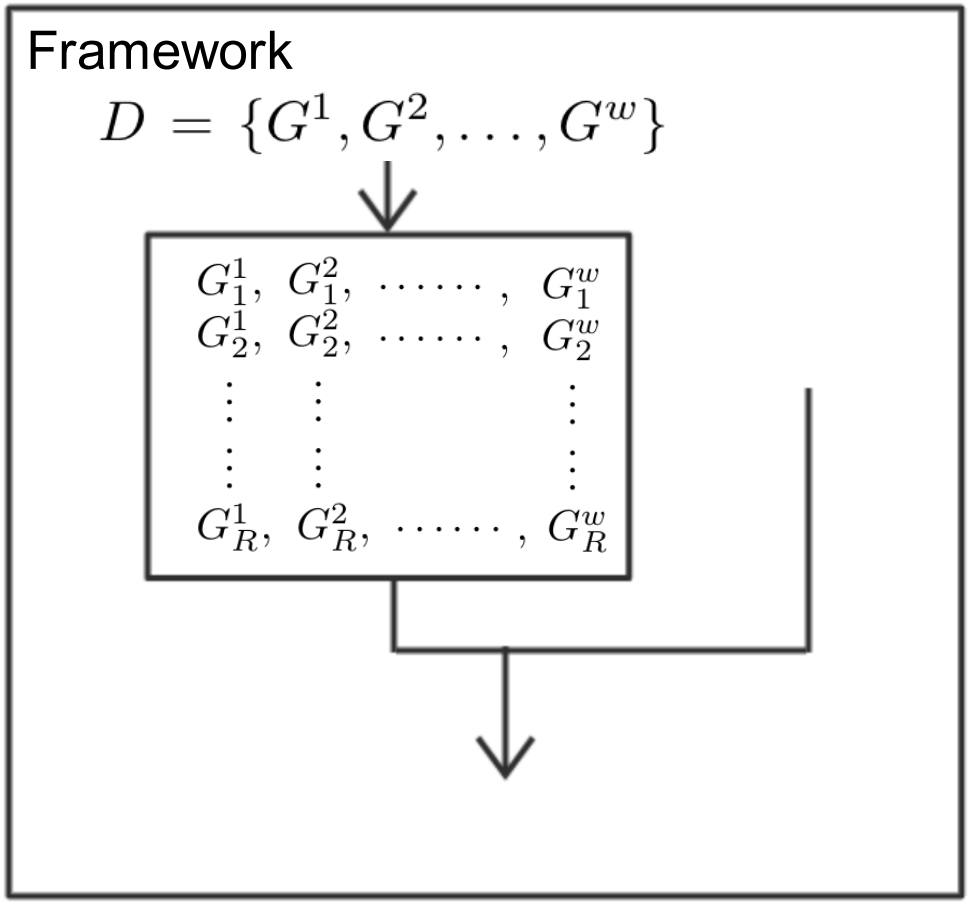
G_j^i denotes the *j-th* subgraph generated from G^i.



Our Solutions

We propose two methods HCS and VCS to efficiently compute $\zeta_D(S)$. (Averaging the distinct reachability on subgraphs generated from snapshots.)

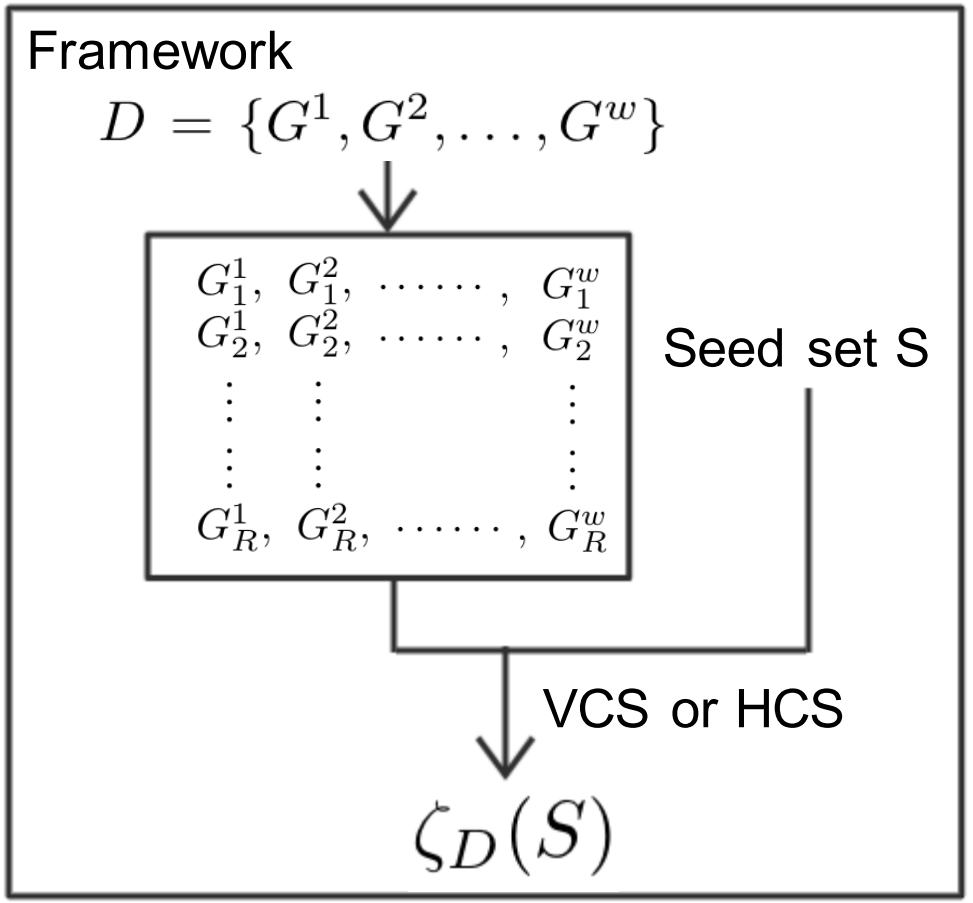
G_j^i denotes the *j-th* subgraph generated from G^i.



Our Solutions

We propose two methods HCS and VCS to efficiently compute $\zeta_D(S)$. (Averaging the distinct reachability on subgraphs generated from snapshots.)

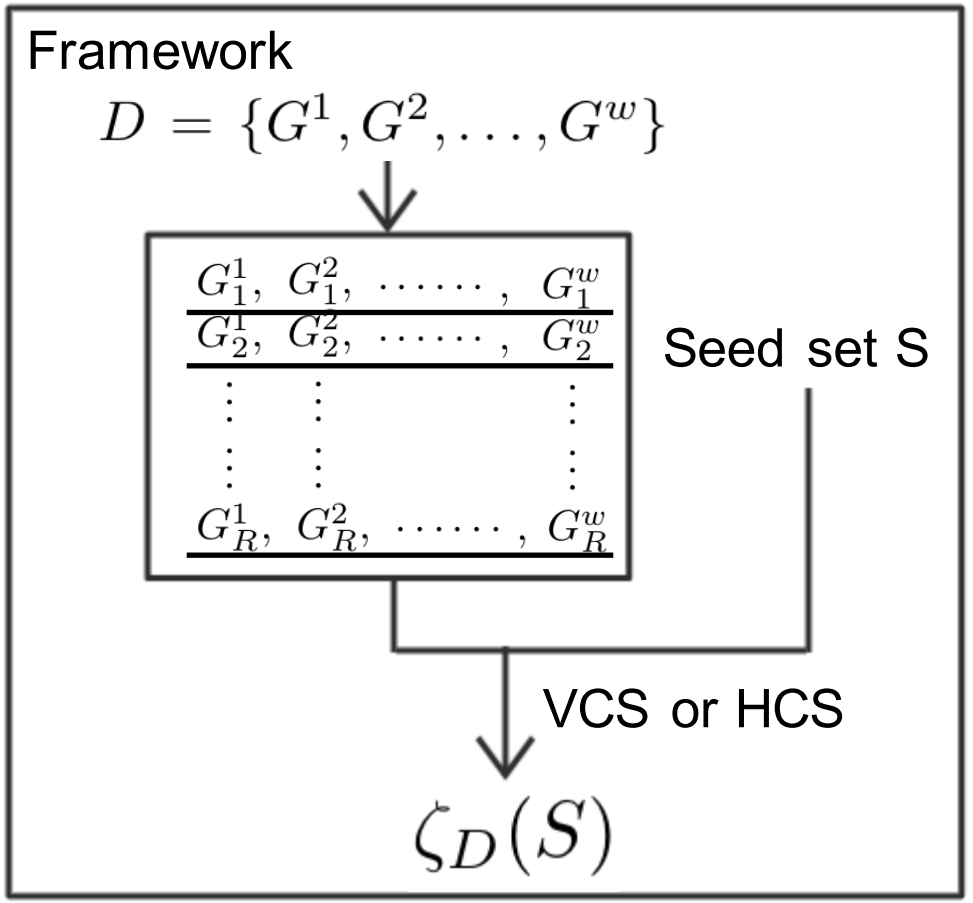
G_j^i denotes the *j-th* subgraph generated from G^i.



Our Solutions

Suppose G_j^i denotes the *j-th* subgraph generated from G^i, and $\mathcal{I}_{G_j^i}(S)$ denotes the set of nodes reached by S in G_j^i.

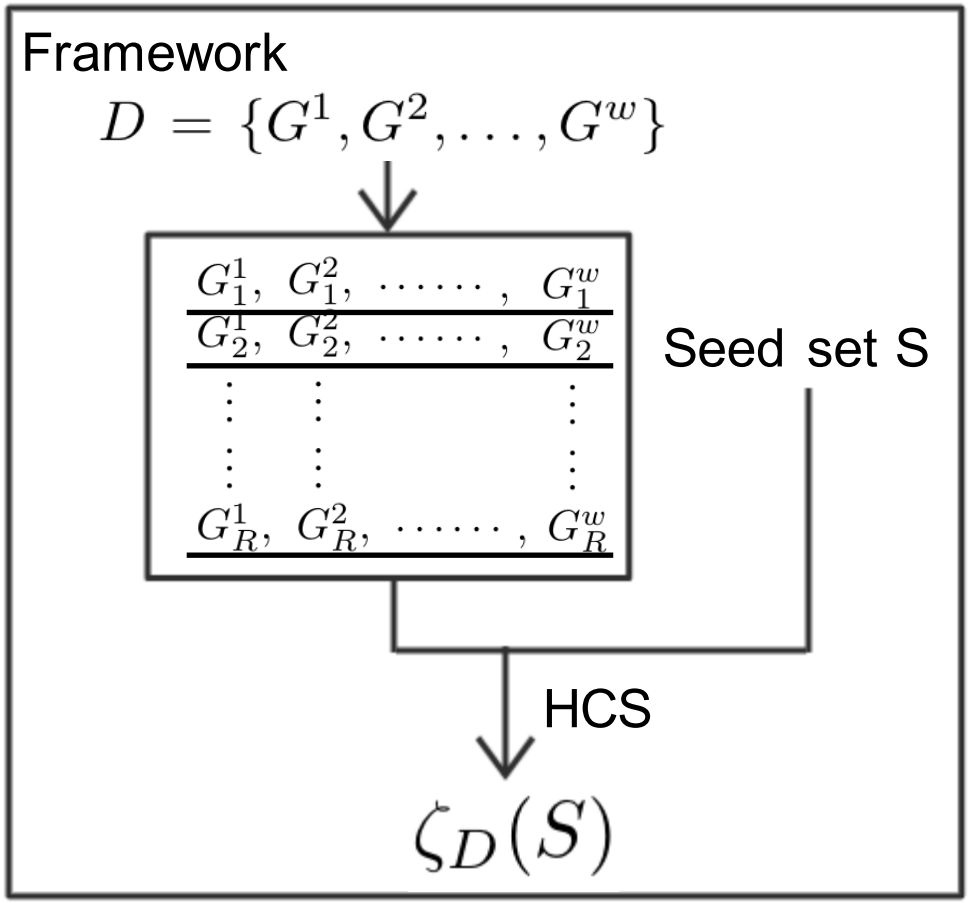
$$\zeta_D(S) = \frac{1}{R}\sum_{j=1}^{R}|\bigcup_{i=1}^{w}\mathcal{I}_{G_j^i}(S)| \quad (1)$$



The Horizontal-Compression-Based Strategy (HCS)

$$\zeta_D(S) = \frac{1}{R}\sum_{j=1}^{R} |\bigcup_{i=1}^{w} \mathcal{I}_{G_j^i}(S)| \quad (1)$$

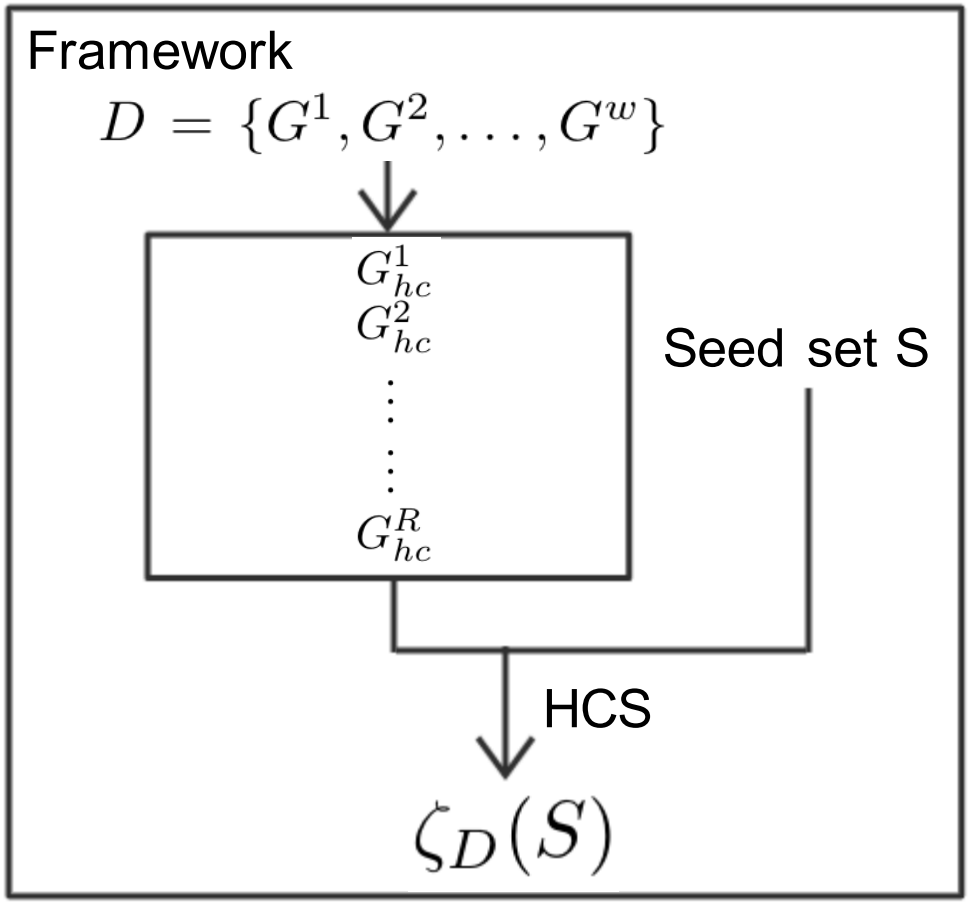
- The naïve has high memory costs and is inefficient.
- HCS
  Compress each horizontal instance $\{G_j^1, G_j^2, \ldots, G_j^w\}$ into a single graph.



The Horizontal-Compression-Based Strategy (HCS)

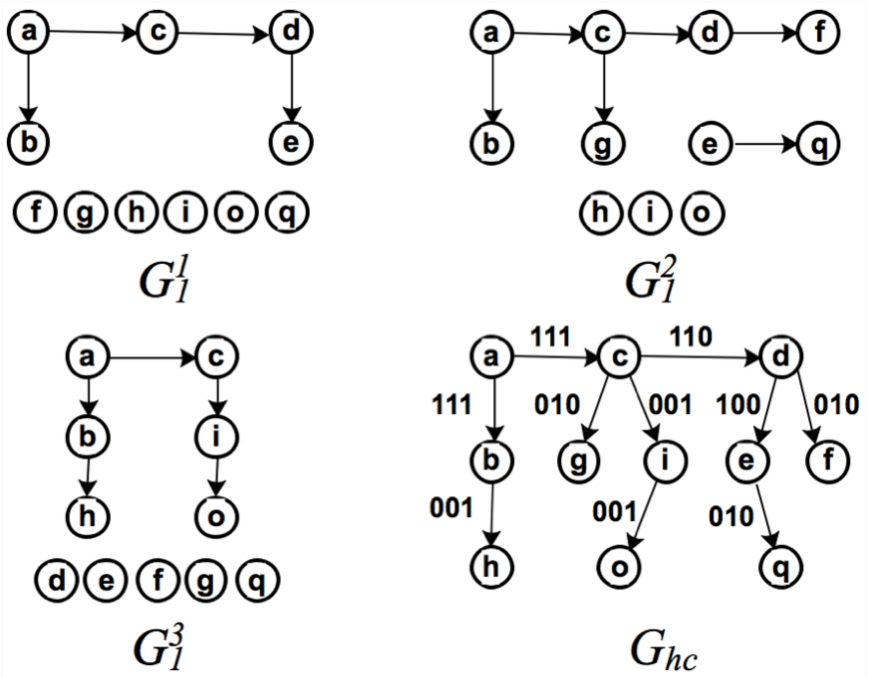
$$\zeta_D(S) = \frac{1}{R}\sum_{j=1}^{R} |\bigcup_{i=1}^{w} \mathcal{I}_{G_j^i}(S)| \quad (1)$$

- The naïve has high memory costs and is inefficient.
- HCS
  Compress each horizontal instance $\{G_j^1, G_j^2, \ldots, G_j^w\}$ into a single graph.



The Horizontal-Compression-Based Strategy (HCS)

- Horizontal Compression



The Horizontal-Compression-Based Strategy (HCS)

- Three Data Structures:

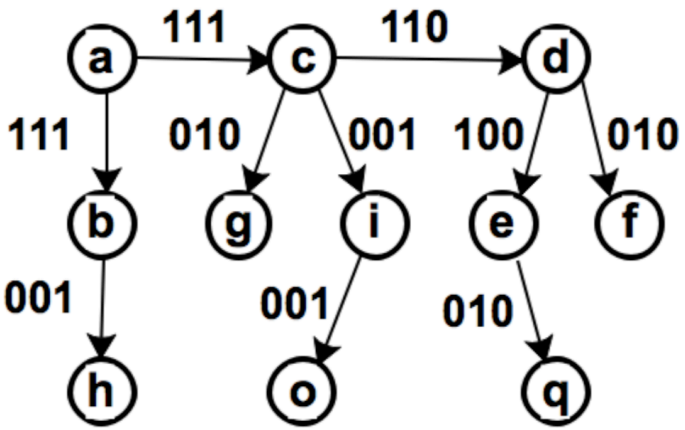
  1. **Containment bitset** $\mathfrak{B}^c$ (for every edge/node).
     Which subgraphs contain this node/edge.
  2. **Traversal bitset** $\mathfrak{B}^t$ (for node u which travels reside at).
     Which subgraphs can continue traversals from the current node.
  3. **Local containment bitset** $\mathfrak{B}^l$ (for every node).
     Initialized as the $\mathfrak{B}^c$ and stores info about which subgraphs contain this node but have not visited this node yet.

- Traversal Rules:

  Node *u* can traverse to neighbor *w* iff the result of AND among $(u, w).\mathfrak{B}^c$ in G^j_{hc} and $w.\mathfrak{B}^l$ and $u.\mathfrak{B}^t$ is not 0.

  1. $u.\mathfrak{B}^t[i] = 1 : G^i_j$ can proceed the traversal.
  2. $(u, w).\mathfrak{B}^c[i] = 1 : G^i_j$ contains edge (u, w).
  3. $w.\mathfrak{B}^l[i] = 1 : G^i_j$ contains *w* and has not visited *w* yet.

The Horizontal-Compression-Based Strategy (HCS)

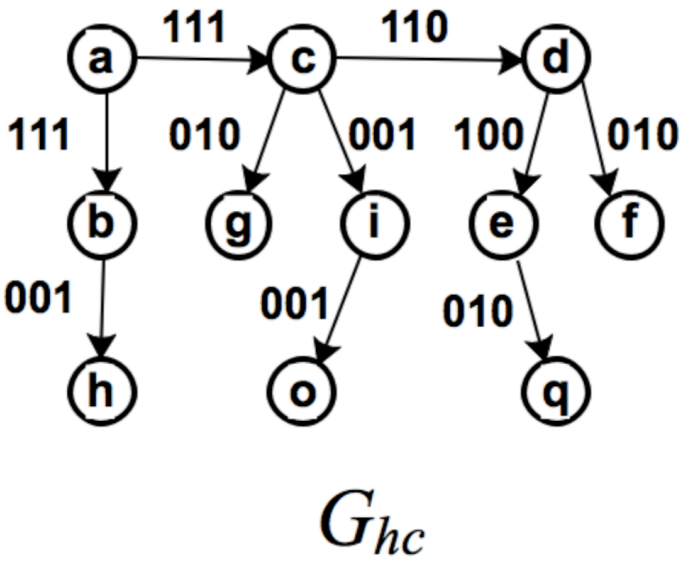


G_{hc}

Example of edge traversals. Bitsets with underscore refers to the updated traversal bitset.

Traversal	B^t	B^l
a to c	B^t & (a,c).B^c & c.B^l $\underline{111}$&111&111=$\underline{111}$	$B^t \oplus$ c.B^l c.B^l : $\underline{111}\oplus$111=000

The Horizontal-Compression-Based Strategy (HCS)

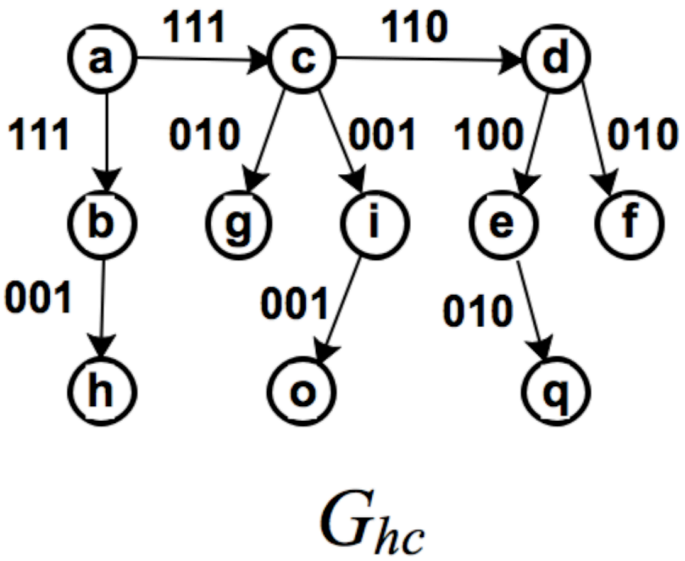


G_{hc}

Example of edge traversals. Bitsets with underscore refers to the updated traversal bitset.

Traversal	B^t	B^l
a to c	B^t & (a,c).B^c & c.B^l 111&111&111=111	$B^t \oplus$ c.B^l c.B^l : 111⊕111=000
c to d	B^t & (c,d).B^c & d.B^l 111&110&111=110	$B^t \oplus$ d.B^l d.B^l : 110⊕111=001

The Horizontal-Compression-Based Strategy (HCS)

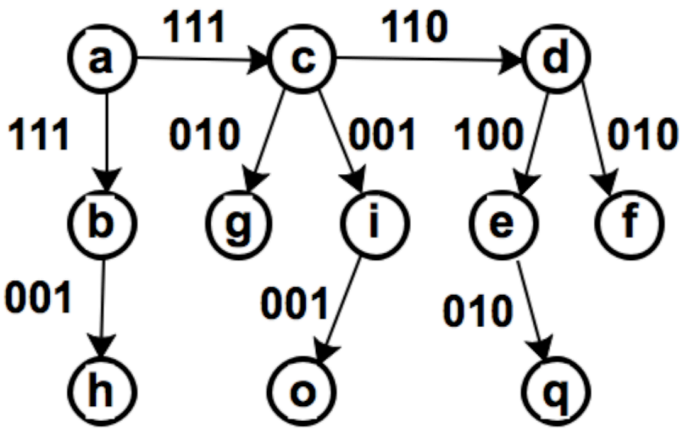


G_{hc}

Example of edge traversals. Bitsets with underscore refers to the updated traversal bitset.

Traversal	B^t	B^l
a to c	B^t & (a,c).B^c & c.B^l 111&111&111=111	$B^t \oplus$ c.B^l c.B^l : 111⊕111=000
c to d	B^t & (c,d).B^c & d.B^l 111&110&111=110	$B^t \oplus$ d.B^l d.B^l : 110⊕111=001
d to e	110&100&111=100	e.B^l : 100⊕111=011

The Horizontal-Compression-Based Strategy (HCS)



G_{hc}

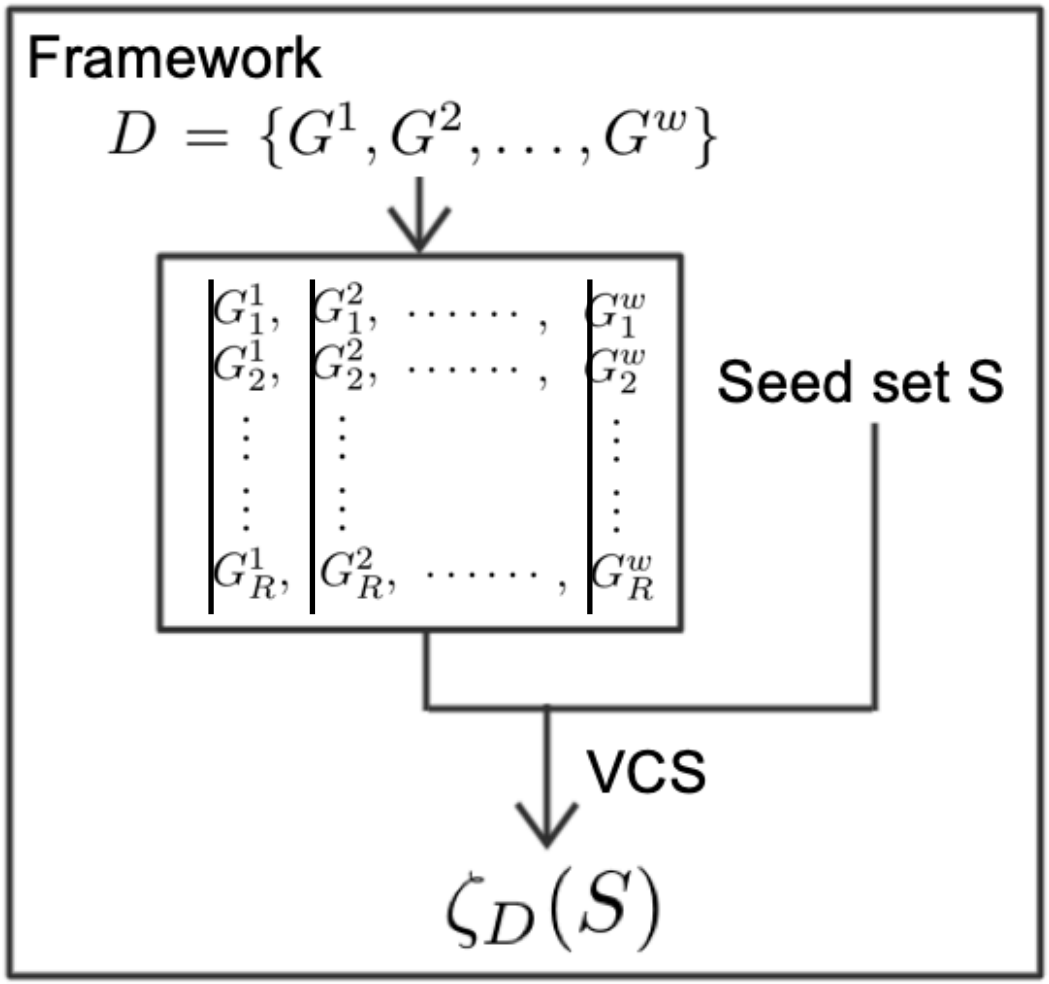
Example of edge traversals. Bitsets with underscore refers to the updated traversal bitset.

Traversal	B^t	B^l
a to c	B^t & (a,c).B^c & c.B^l 111&111&111=111	$B^t \oplus$ c.B^l c.B^l : 111⊕111=000
c to d	B^t & (c,d).B^c & d.B^l 111&110&111=110	$B^t \oplus$ d.B^l d.B^l : 110⊕111=001
d to e	110&100&111=100	e.B^l : 100⊕111=011
e to q	100&010&100=000	No update to q.B^l

The Vertical-Compression-Based Strategy (VCS)

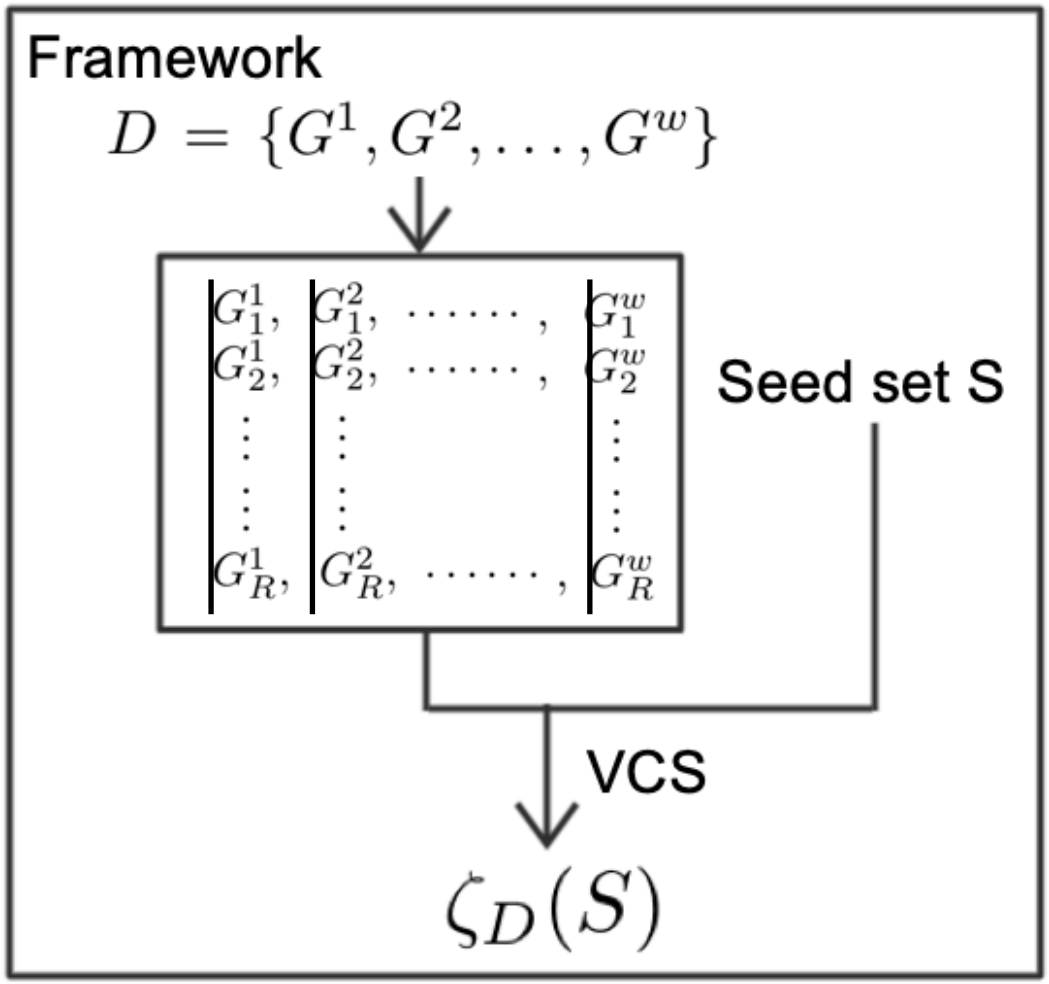
Observation:
More node/edge overlaps exist among subgraphs generated from the same snapshot.

Vertically processing:
Process graphs by columns.



The Vertical-Compression-Based Strategy (VCS)

- The naïve has high memory costs and is inefficient.
- VCS
  Compress each vertical instance $\{G_1^i, G_2^i, \ldots, G_R^i\}$ into a single graph.

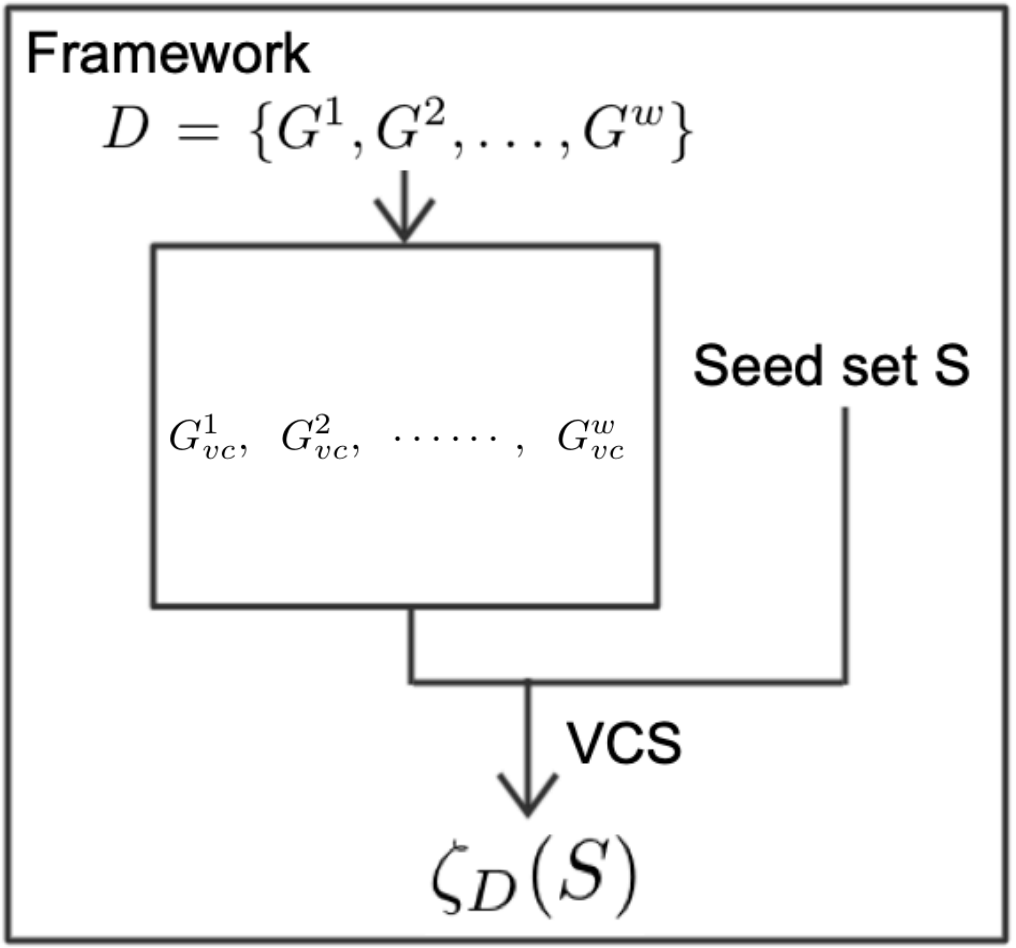


The Vertical-Compression-Based Strategy (VCS)

VCS

Compresses each vertical instance $\{G_1^i, G_2^i, \ldots, G_R^i\}$ into a single graph.

Requires additional bitsets and new traversal rules.



Experiment

1. Datasets

Dataset	$\|V\|$	$\|E\|$
NetHEPT	15K	59K
DBLP	654K	2M
Hyves	1.4M	3M
Flixster	2.5M	8M
LiveJournal	5.2M	49M

2. Baselines

(1) SGDU [1] (subgraph-based)
(2) PMC [2] (subgraph-based)
(3) IMM [3] (Sketch-based)
(4) EasyIM [4] (Heuristic)
(5) IMRank [5] (Heuristic)
(6) CELF [6] (Simulation-based)

[1] S. Cheng, et al, "Staticgreedy: solving the scalability-accuracy dilemma in influence maximization," in CIKM, 2013.

[2] N. Ohsaka, et al, "Fast and accurate influence maximization on large networks with pruned monte-carlo simulations," in AAAI, 2014.

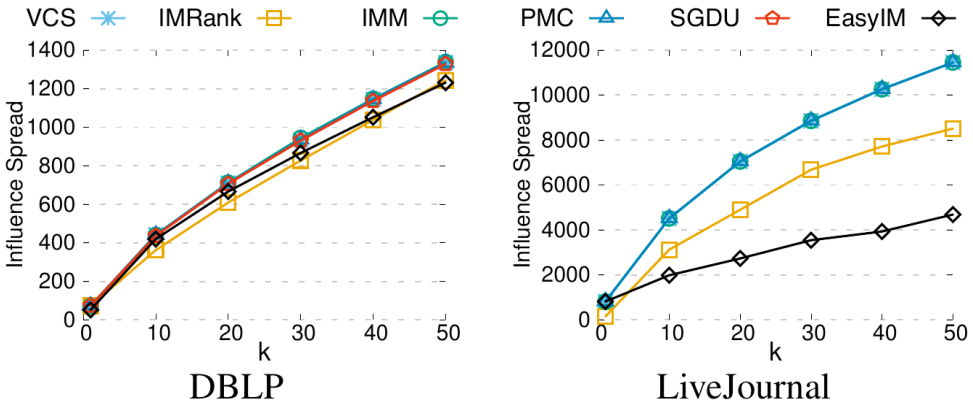
[3] Y. Tang, et al, "Influence maximization in near-linear time: A martingale approach," in SIGMOD, 2015.

[4] S. Galhotra, et al, "Holistic influence maximization: Combining scalability and efficiency with opinion-aware models," in SIGMOD, 2016.

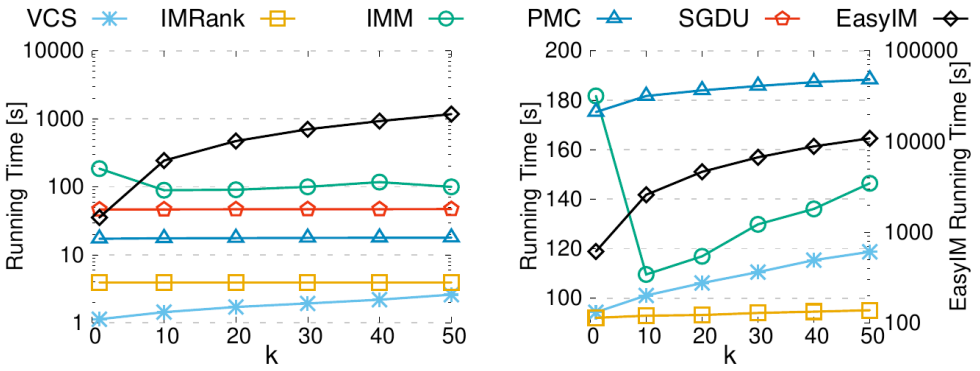
[5] S.Cheng, et al,"IMrank: influence maximization via finding self-consistent ranking," in SIGIR, 2014.

[6] J. Leskovec, et al, "Cost-effective outbreak detection in networks," in SIGKDD, 2007.

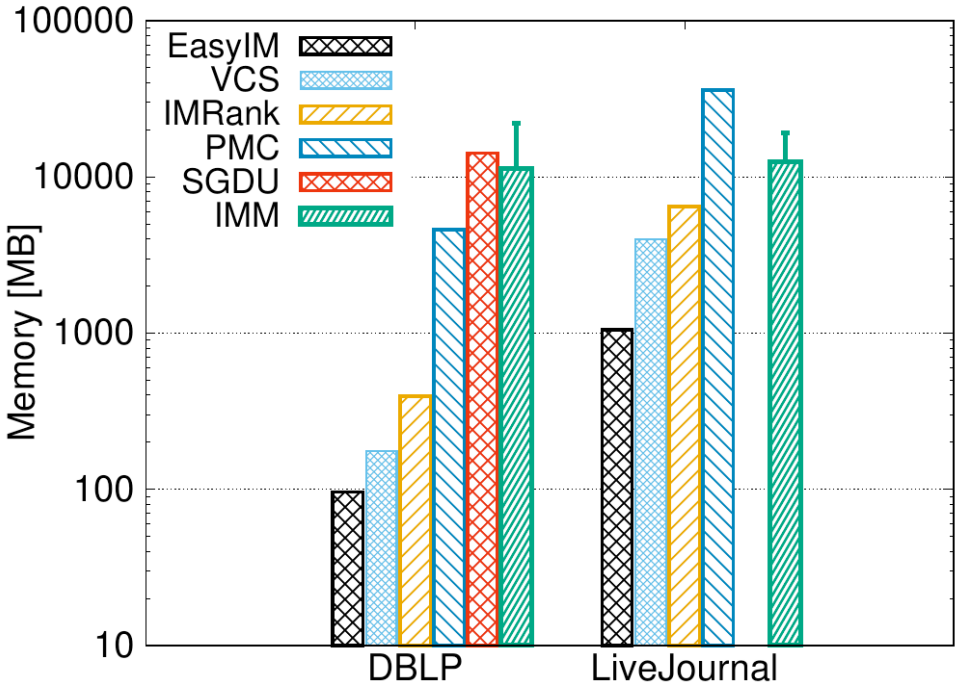
Experiment Results for the IM Problem



Comparison of influence spreads.



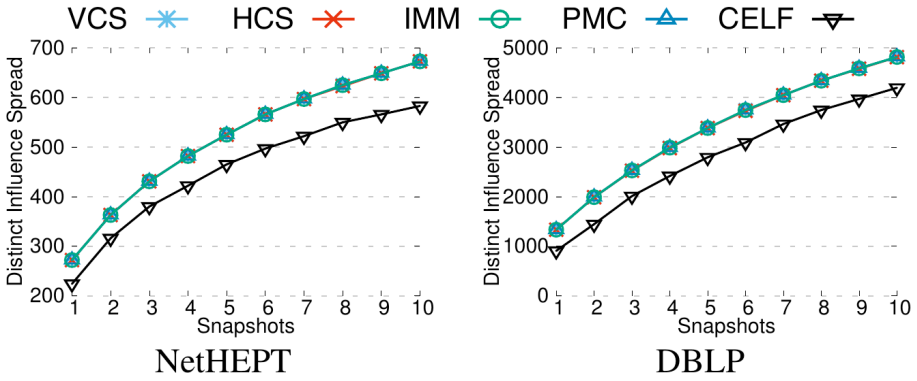
Comparison of running time.*



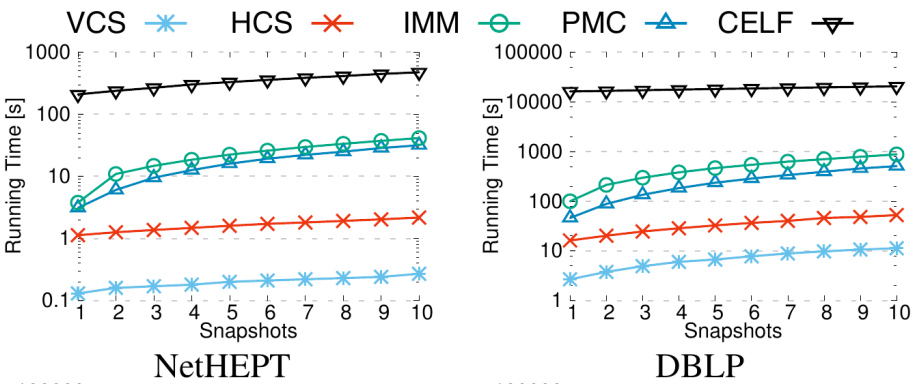
Comparison of memory consumption.*

* The y-axes in these graphs are in log scale.

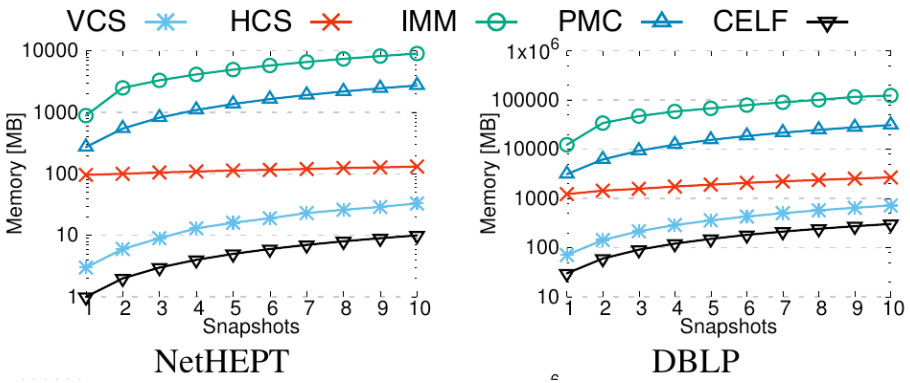
Experiment Results for the DIM Problem



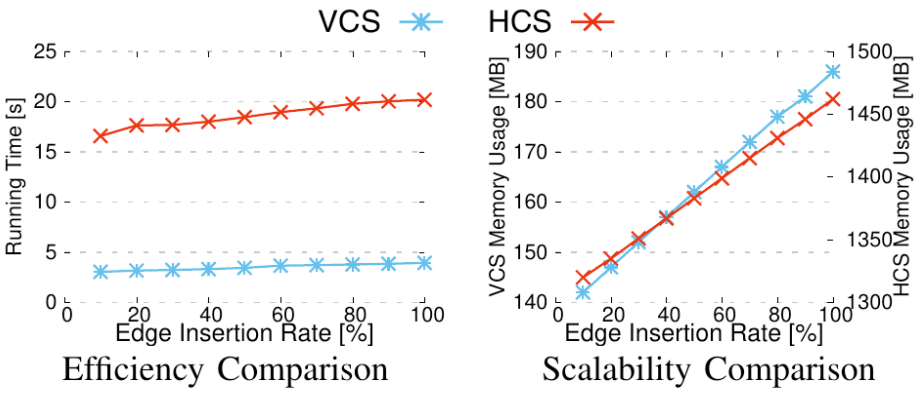
Comparison of distinct influence spreads ($k = 50$).



Comparison of running time ($k = 50$).*



Comparison of memory consumption ($k = 50$).*



Performance comparison between HCS and VCS.

* The y-axes in these graphs are in log scale.

Thanks!