Layph: Making Change Propagation Constraint in Incremental Graph Processing by Layering Graph

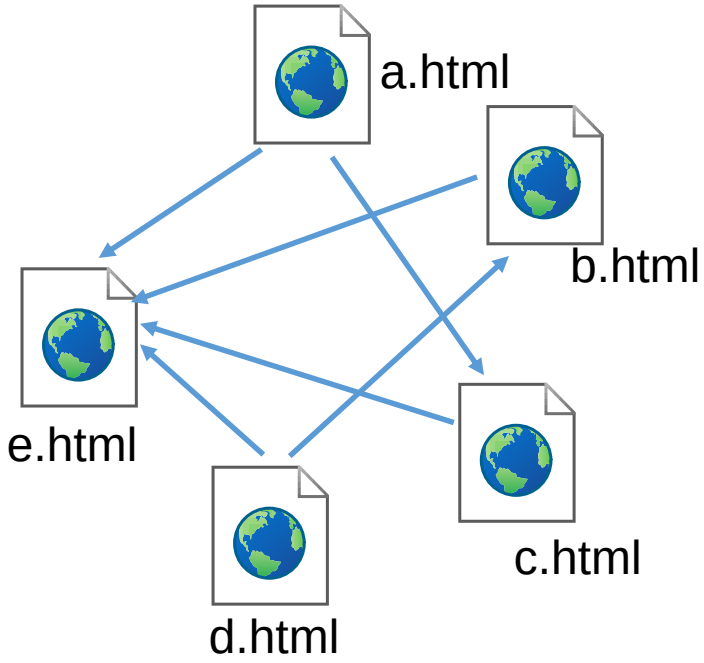
Song Yu[1] , Shufeng Gong[1,5] , Yanfeng Zhang[1], Wenyuan Yu[2] , Qiang Yin[3] , Chao Tian[4] Qian Tao[2],

Yongze Yan[1], Ge Yu[1], Jingren Zhou[2]

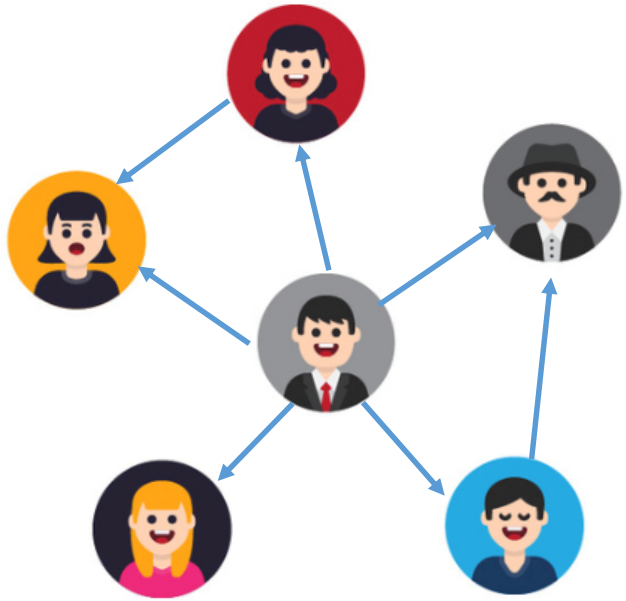
Northeastern University[1] Alibaba Group[2] Shanghai Jiao Tong University[3] Chinese Academy of Sciences[4]

Key Laboratory of Intelligent Computing in Medical Image of Ministry of Education, Northeastern University[5]

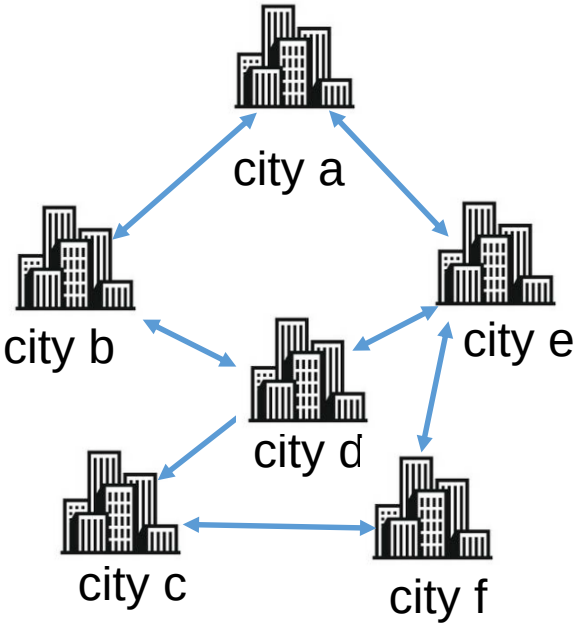
Graph



Web network



Social network



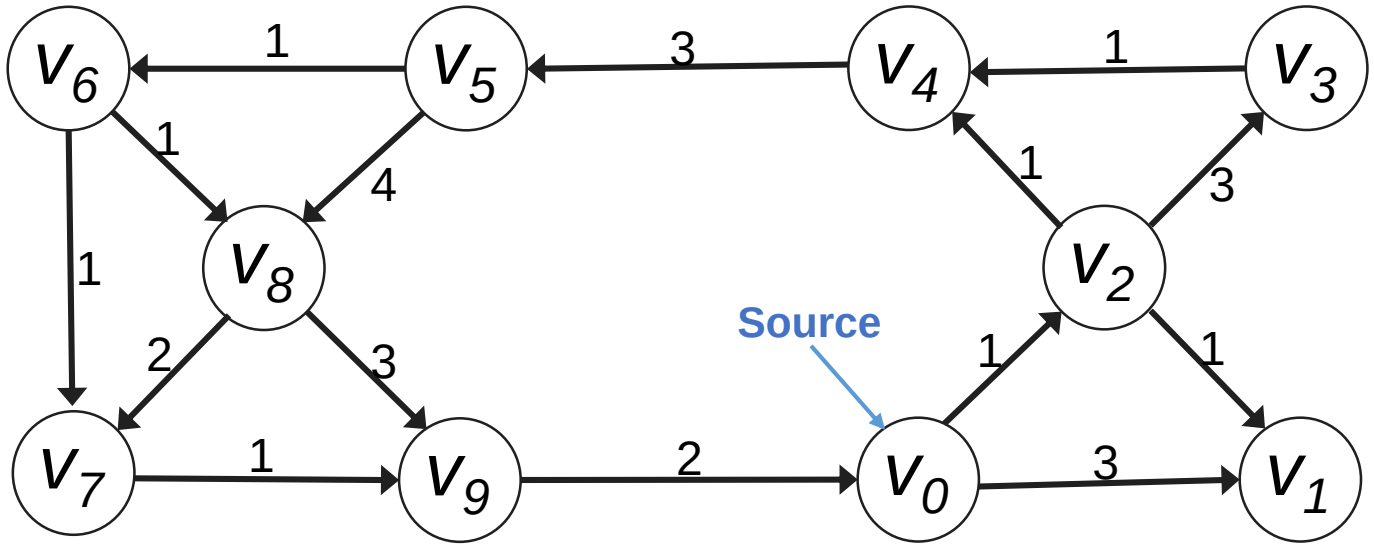
Road network

Iterative Computation

Iterative graph algorithms, e.g., single source shortest path(SSSP) and PageRank, have been widely applied in many fields.

Iterative Computation

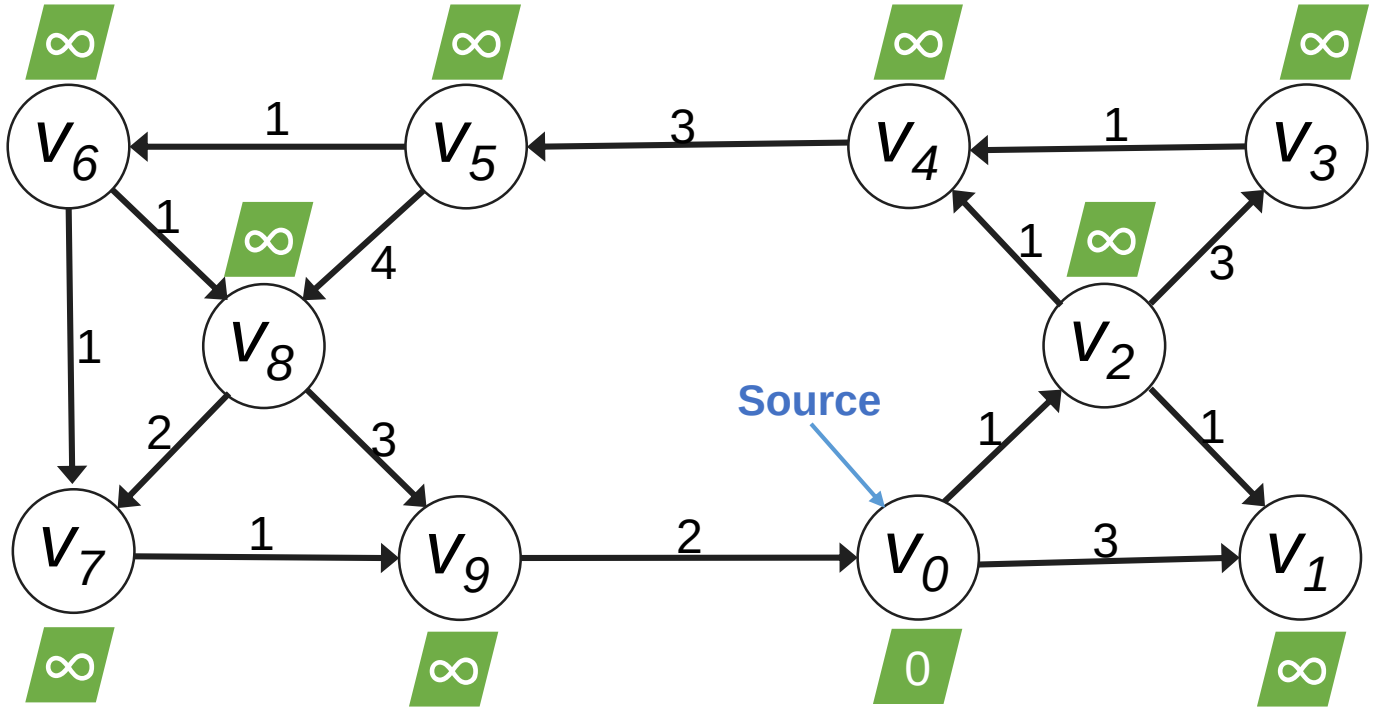
Iterative graph algorithms, e.g., single source shortest path(SSSP) and PageRank, have been widely applied in many fields.



An example iterative computation for SSSP.

Iterative Computation

Iterative graph algorithms, e.g., single source shortest path(SSSP) and PageRank, have been widely applied in many fields.



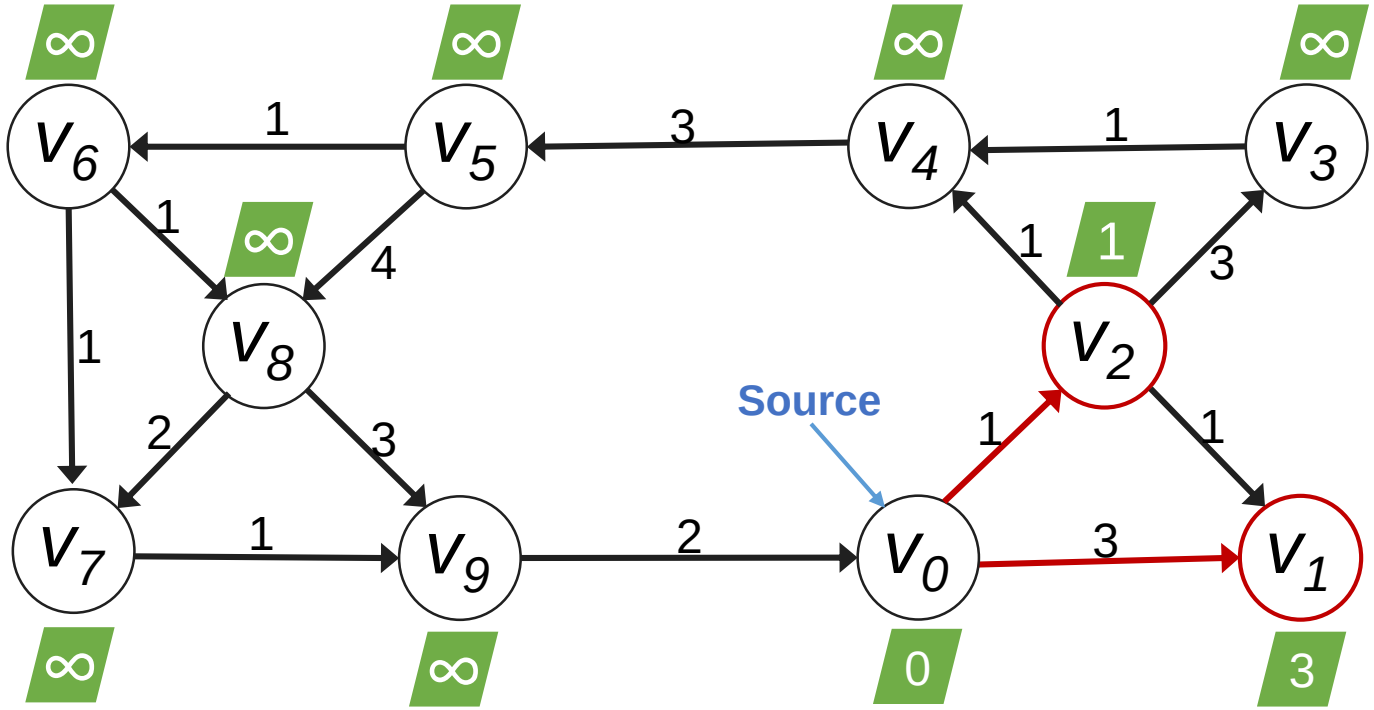
An example iterative computation for SSSP.

Initialization

The source vertex's state is 0, and the others are infinity.

Iterative Computation

Iterative graph algorithms, e.g., single source shortest path(SSSP) and PageRank, have been widely applied in many fields.



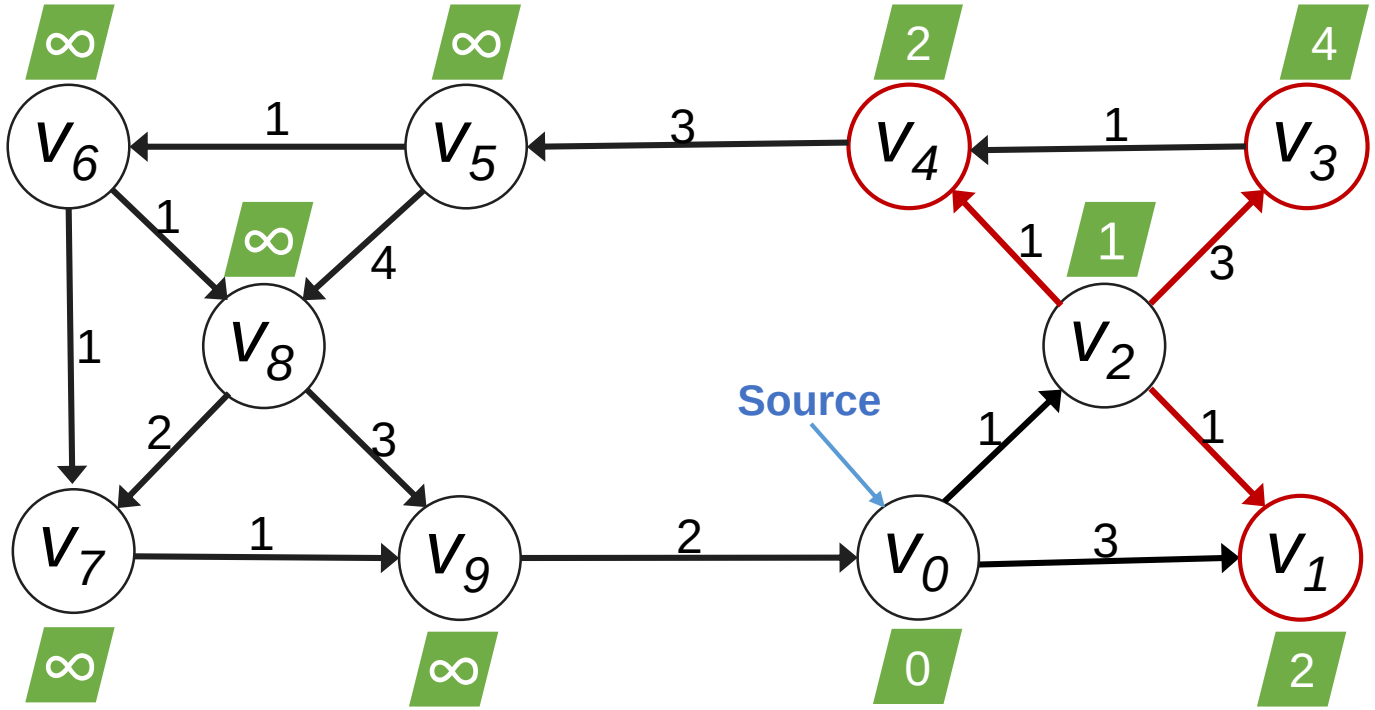
An example iterative computation for SSSP.

First iteration

Update the states of v_1 and v_2.

Iterative Computation

Iterative graph algorithms, e.g., single source shortest path(SSSP) and PageRank, have been widely applied in many fields.



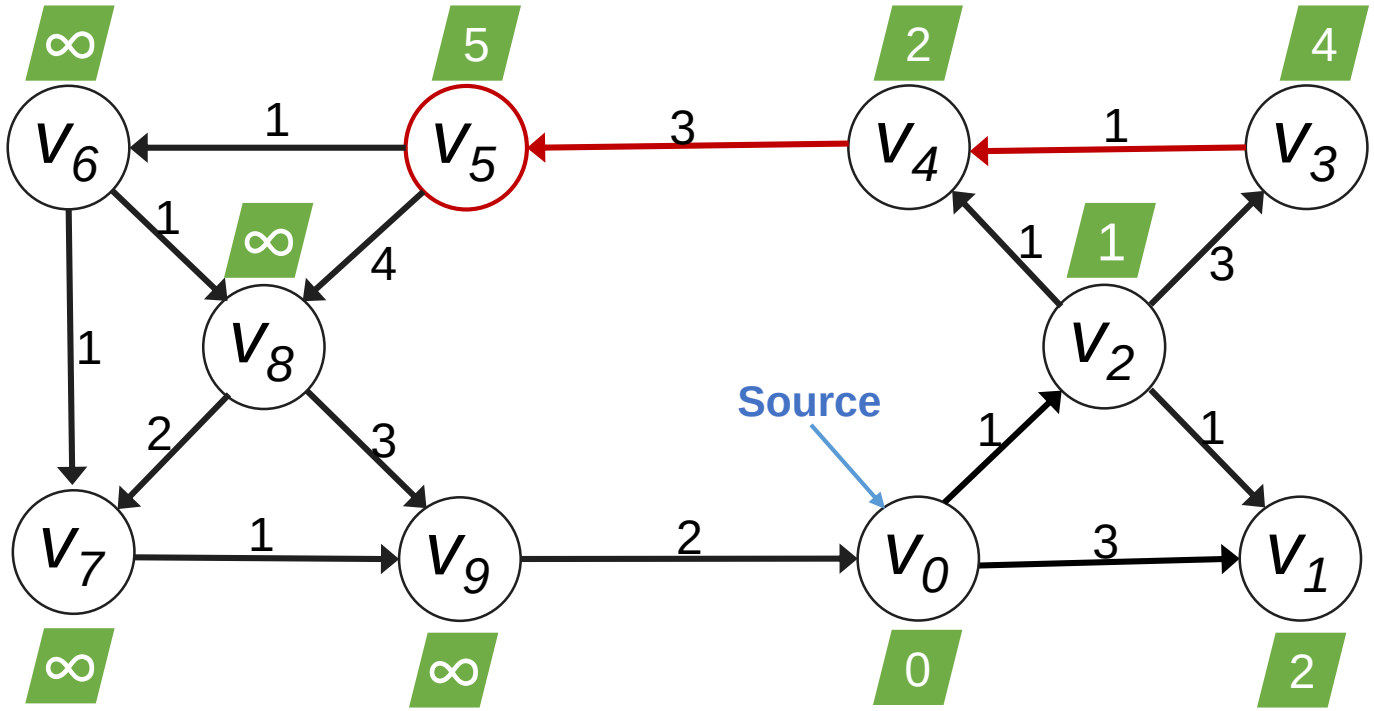
An example iterative computation for SSSP.

Second iteration

Update the states of v_1, v_3, and v_4.

Iterative Computation

Iterative graph algorithms, e.g., single source shortest path(SSSP) and PageRank, have been widely applied in many fields.



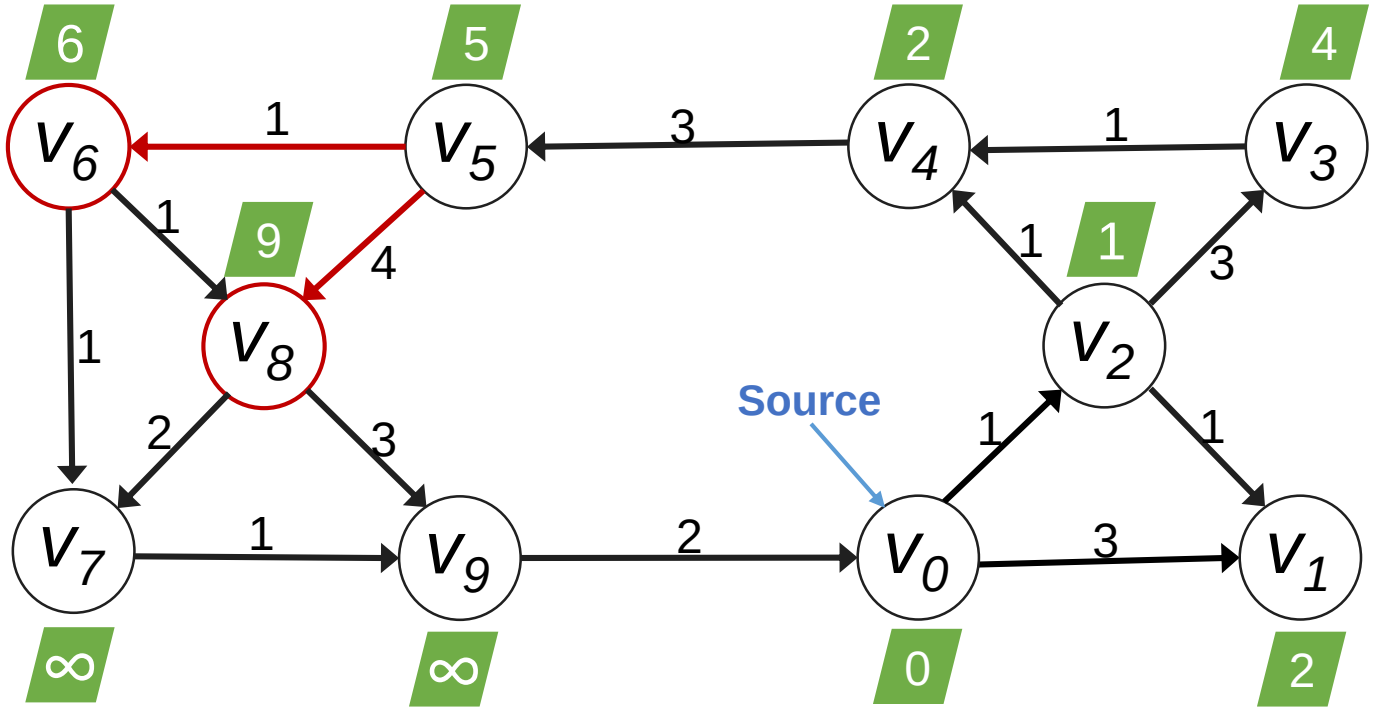
An example iterative computation for SSSP.

Third iteration

Update the state of v_5.

Iterative Computation

Iterative graph algorithms, e.g., single source shortest path(SSSP) and PageRank, have been widely applied in many fields.



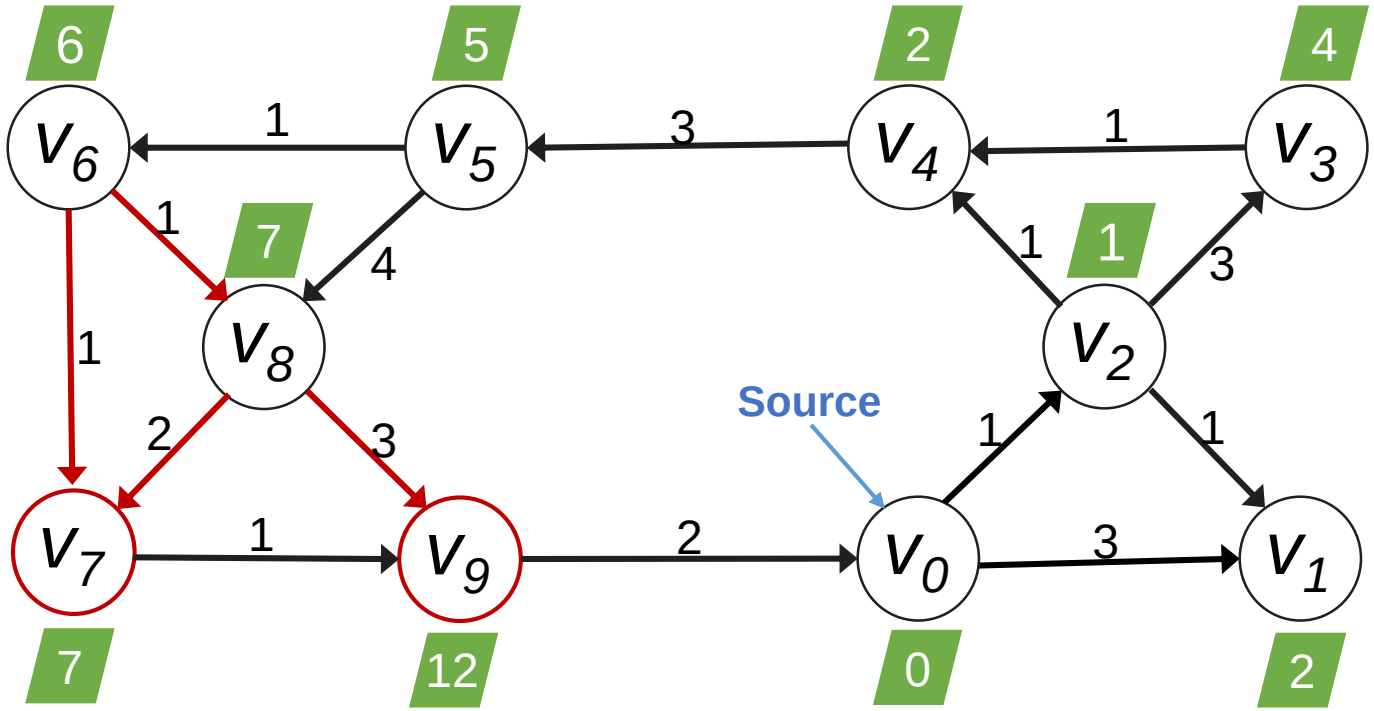
An example iterative computation for SSSP.

Fourth iteration

Update the states of v_6 and v_8.

Iterative Computation

Iterative graph algorithms, e.g., single source shortest path(SSSP) and PageRank, have been widely applied in many fields.



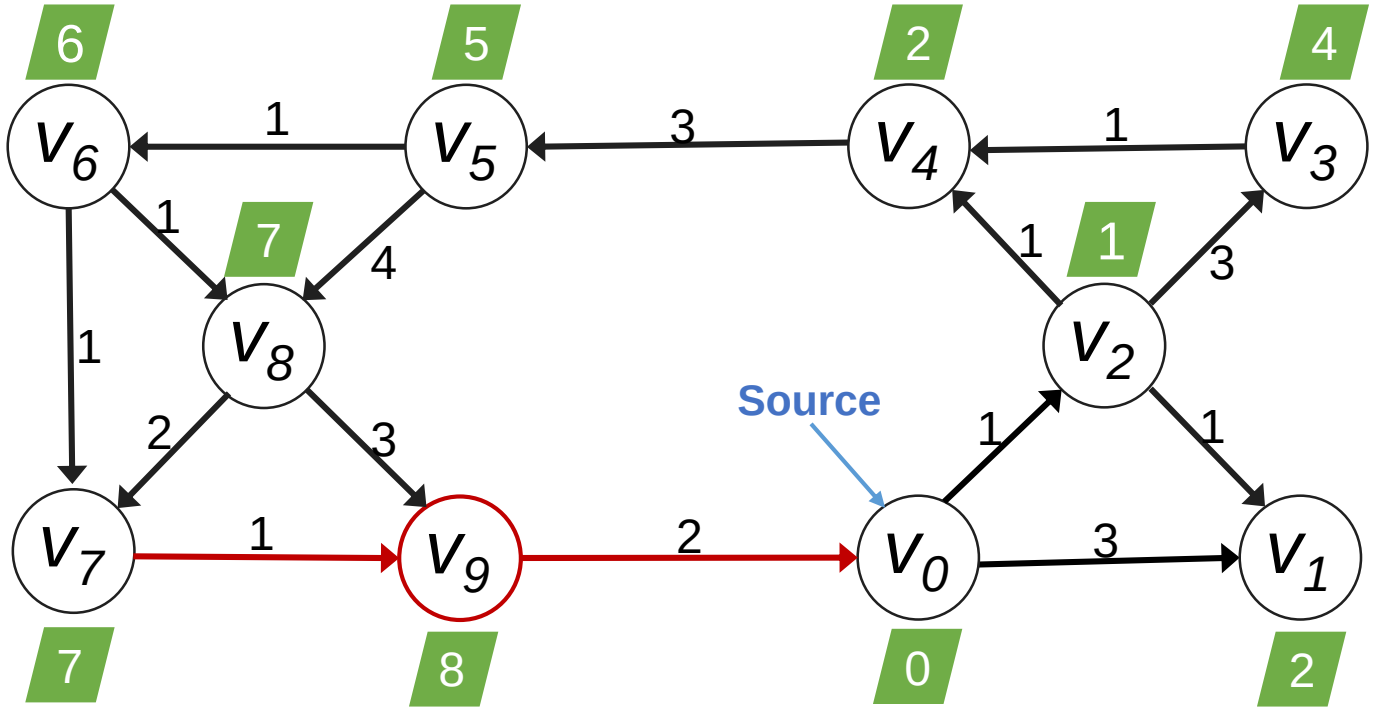
An example iterative computation for SSSP.

Fifth iteration

Update the states of v_7, v_8 and v_9.

Iterative Computation

Iterative graph algorithms, e.g., single source shortest path(SSSP) and PageRank, have been widely applied in many fields.



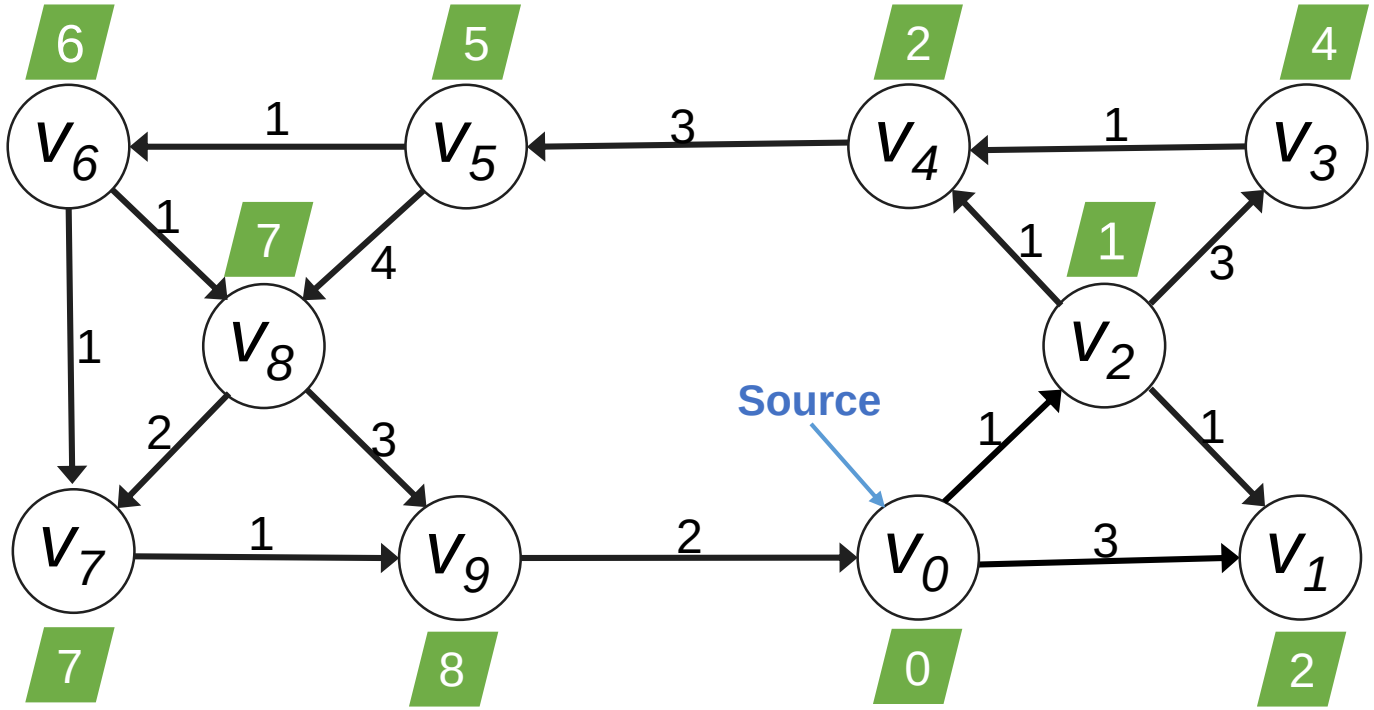
An example iterative computation for SSSP.

Sixth iteration

Update the state of v_9.

Iterative Computation

Iterative graph algorithms, e.g., single source shortest path(SSSP) and PageRank, have been widely applied in many fields.

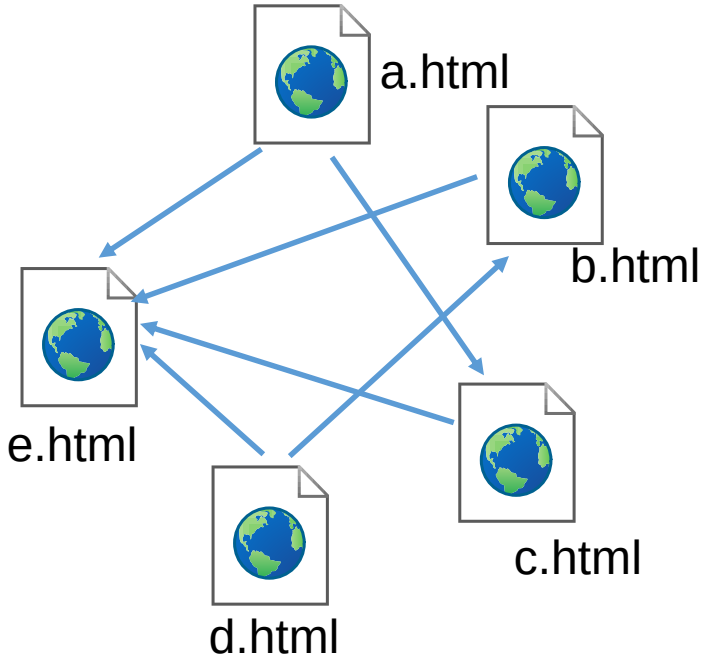


An example iterative computation for SSSP.

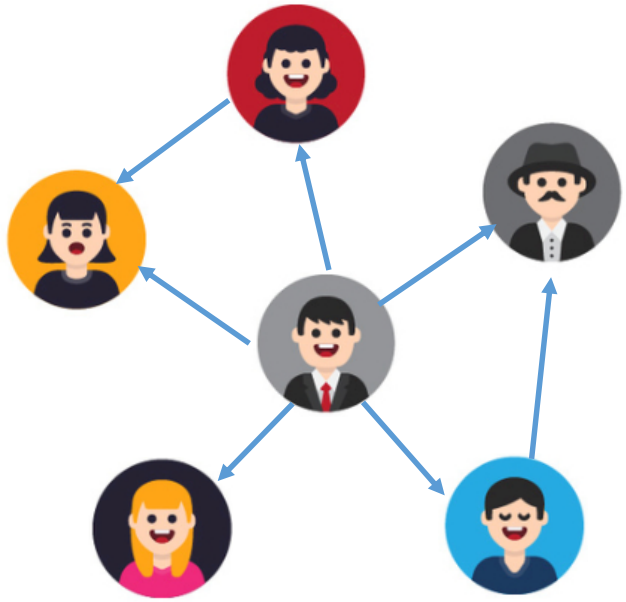
Iteration ends

The states of all vertices converge.

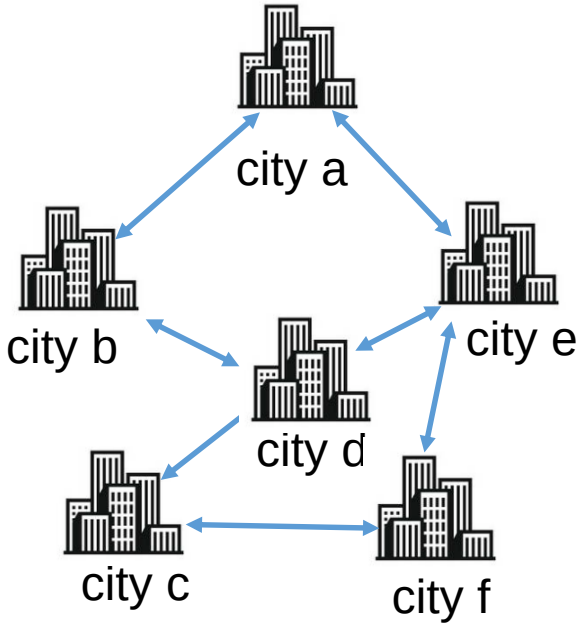
Evolving Graph



Web network



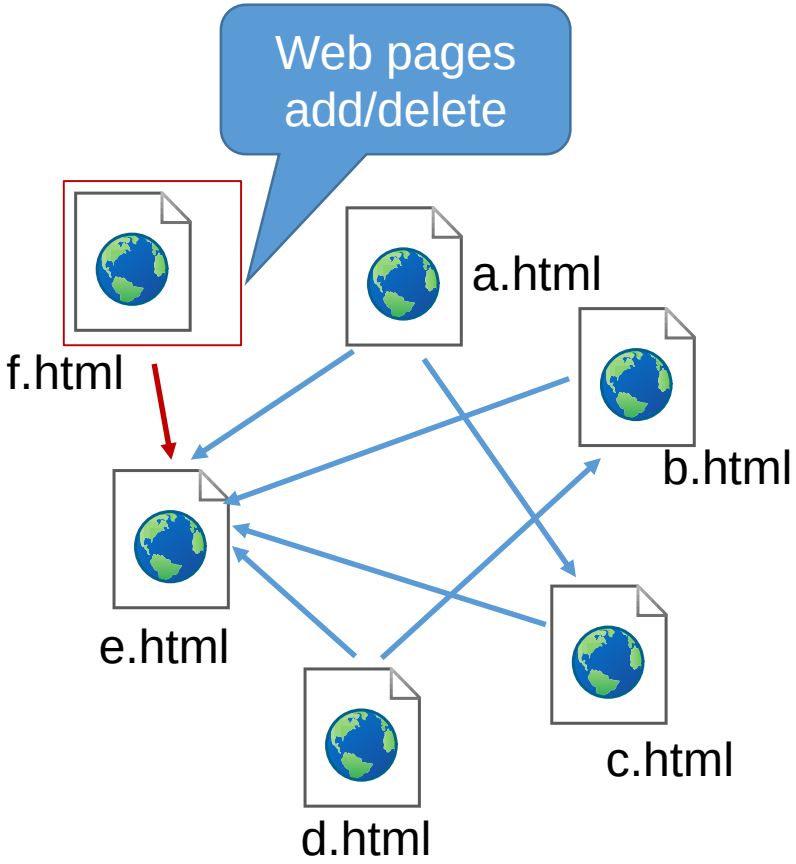
Social network



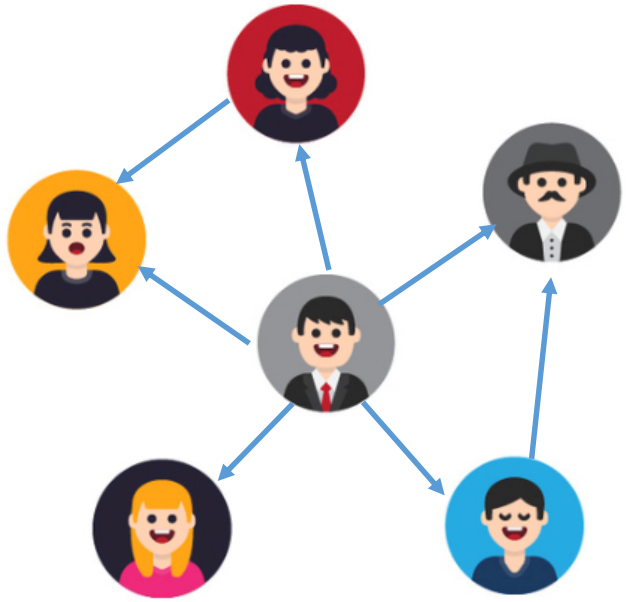
Road network

The graphs are constantly evolving in real time!

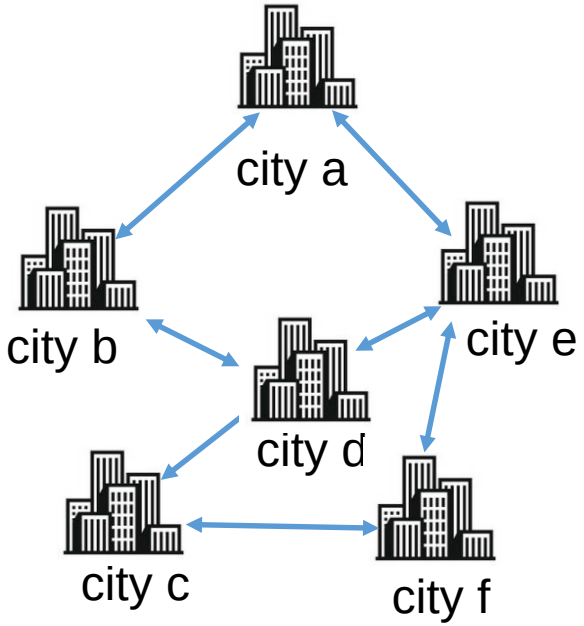
Evolving Graph



Web network



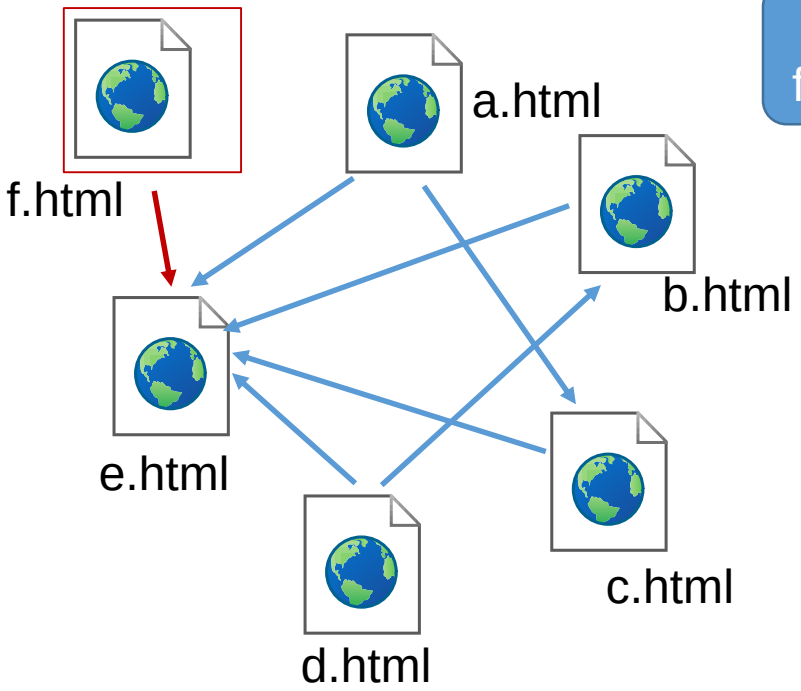
Social network



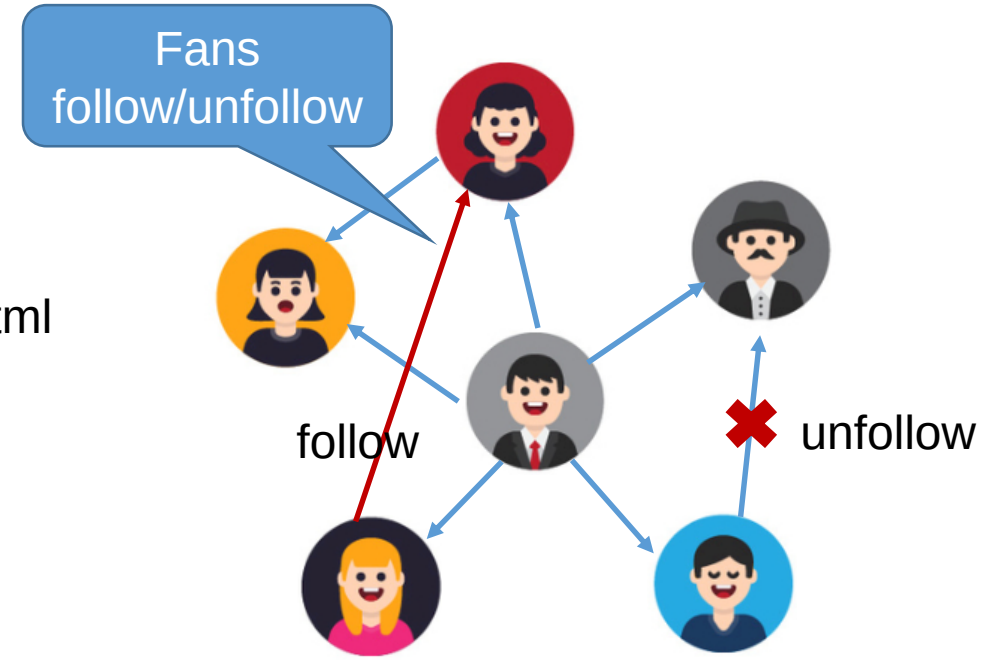
Road network

The graphs are constantly evolving in real time!

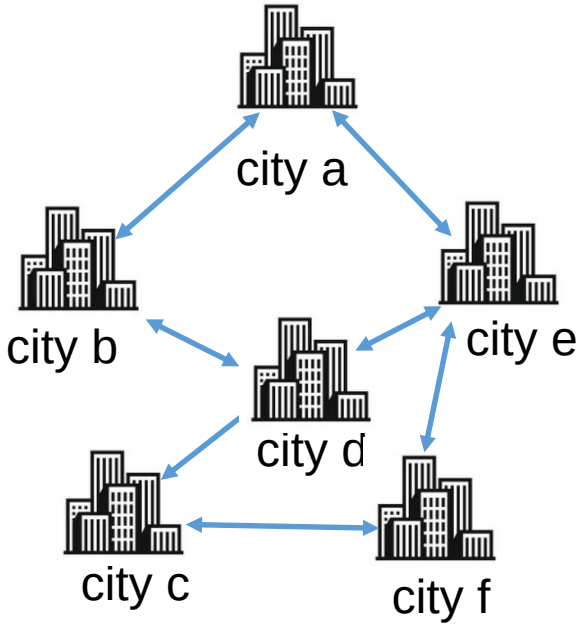
Evolving Graph



Web network



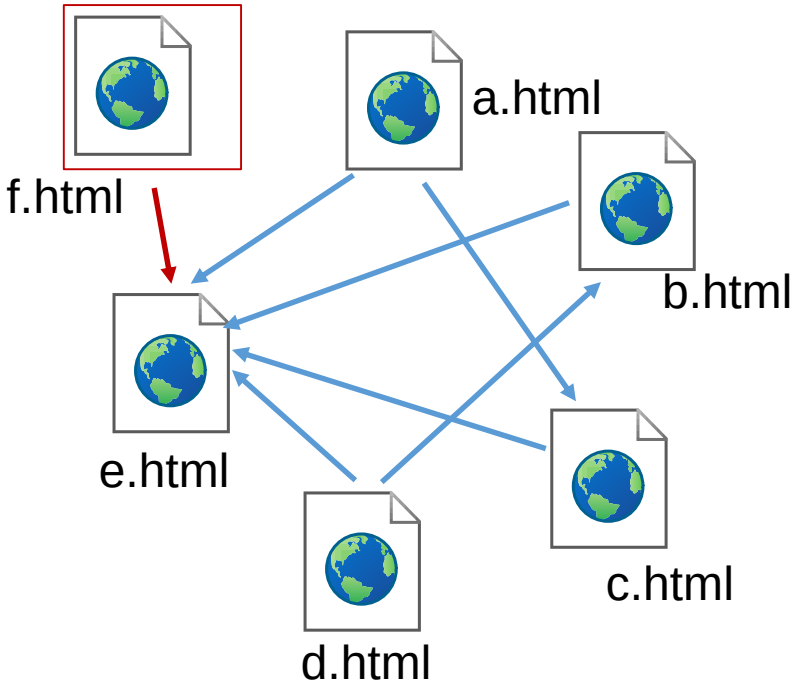
Social network



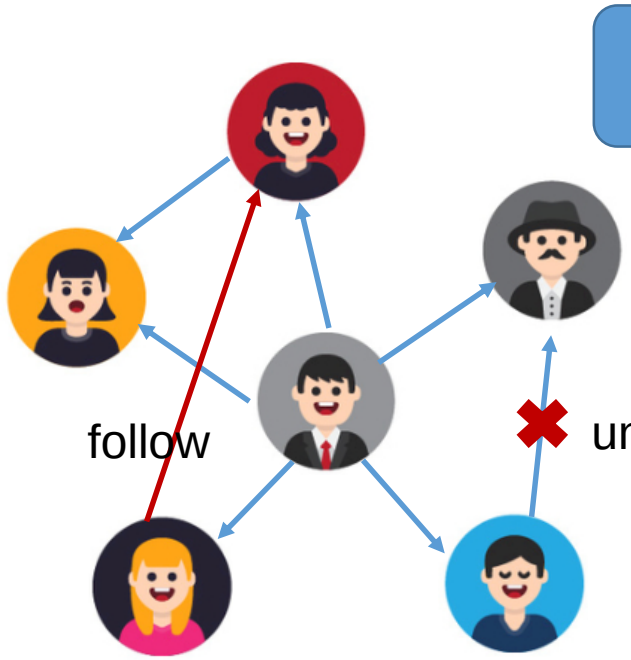
Road network

The graphs are constantly evolving in real time!

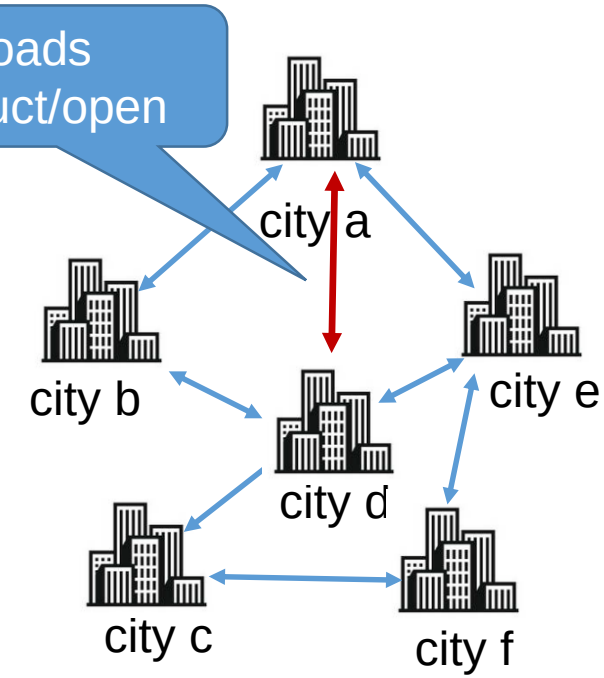
Evolving Graph



Web network



Social network

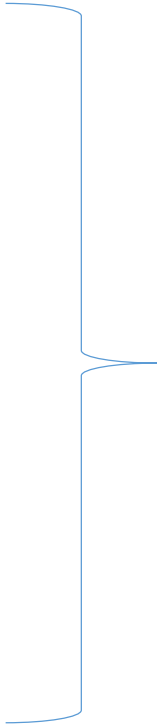


Road network

The graphs are constantly evolving in real time!

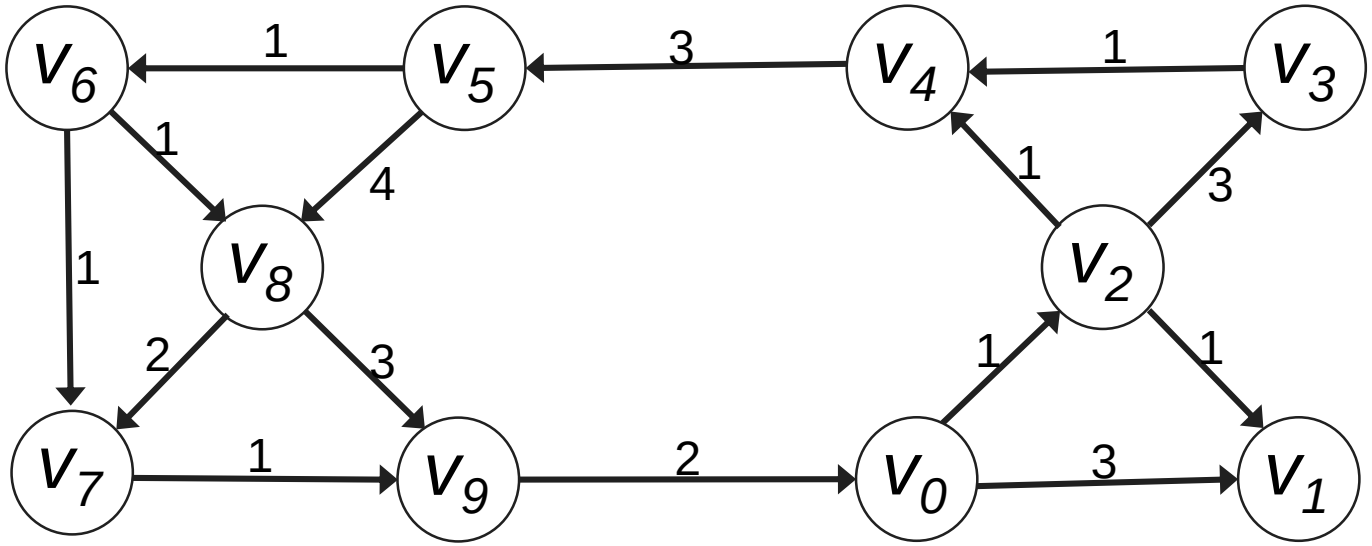
Incremental graph processing systems

- Tornado [SIGMOD'16]
- KickStarter [ASPLOS'17]
- GraphBolt [EuroSys'19]
- Ingress [VLDB'21]
- DZiG [EuroSys'21]
- RisGraph [SIGMOD'21]
- GraphFly [SC'22]
- **...**



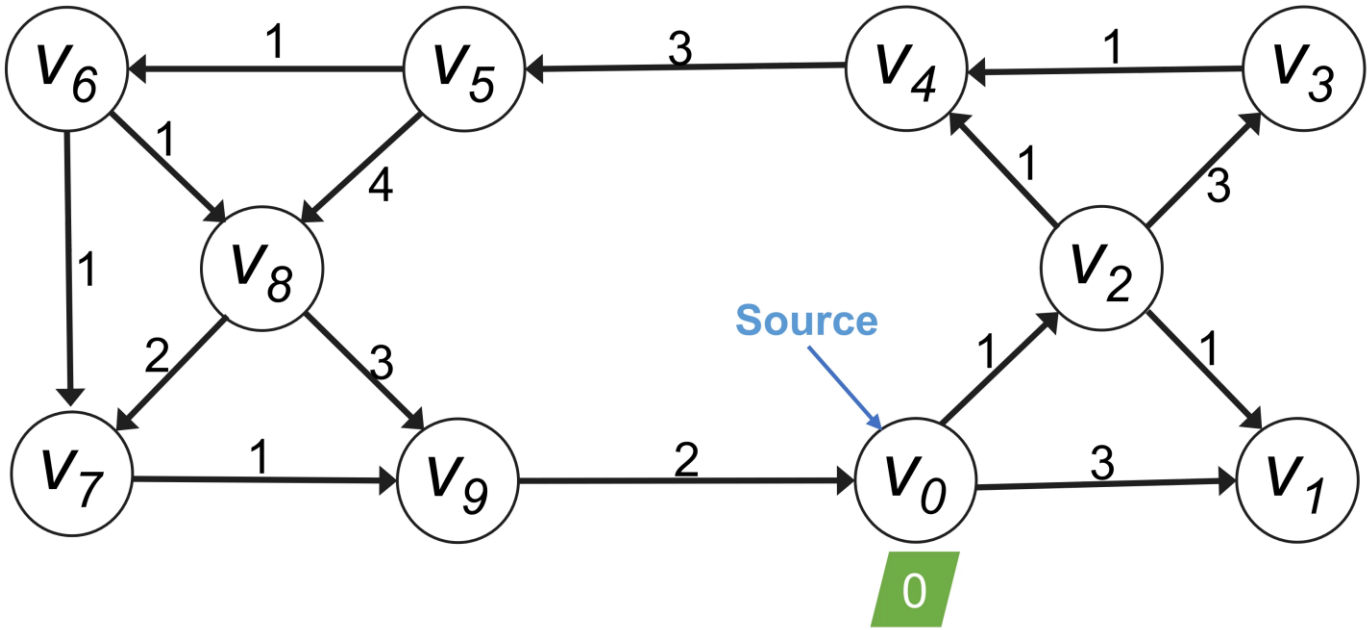
Try to avoid redundant computations

Example：Incremental SSSP



(a) A simple graph G

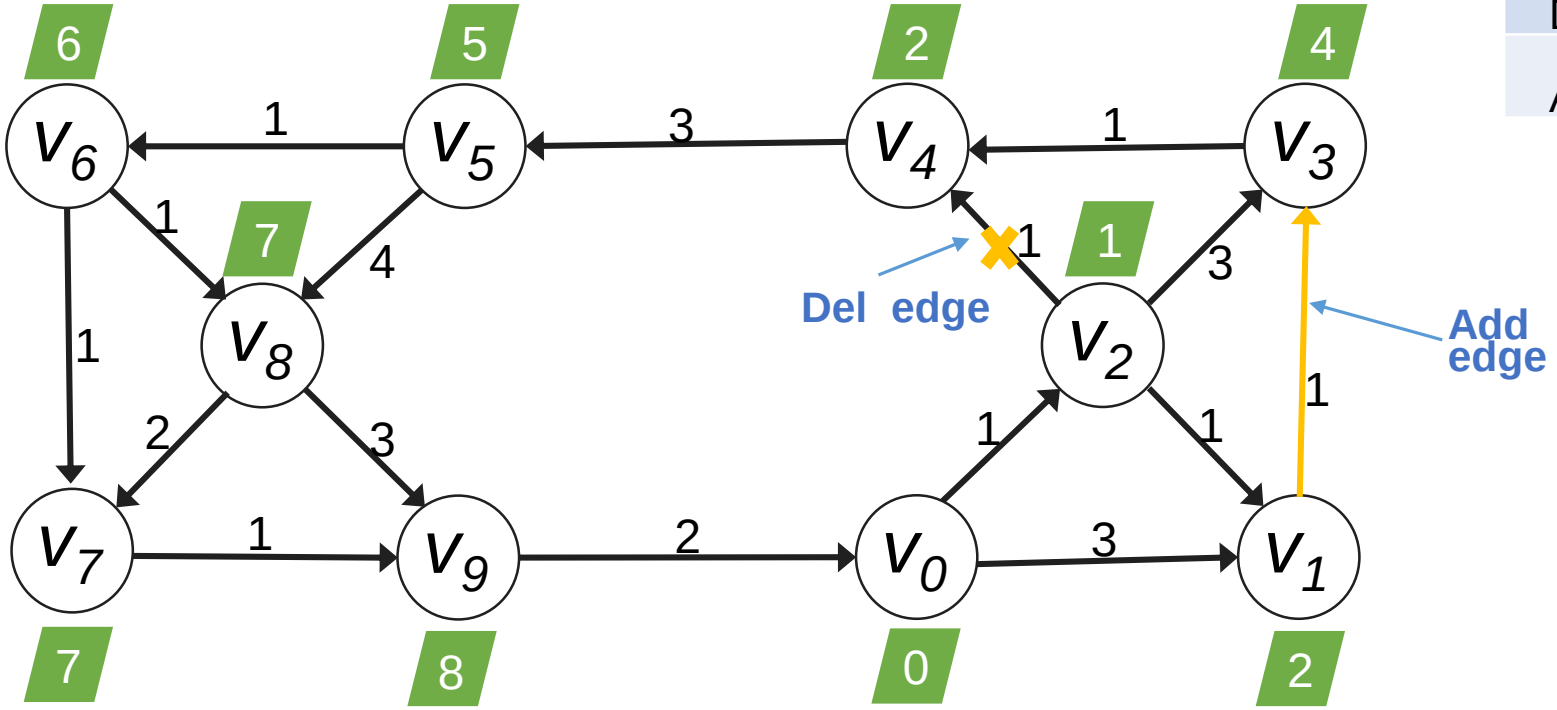
Example：Incremental SSSP



(a) A simple graph G

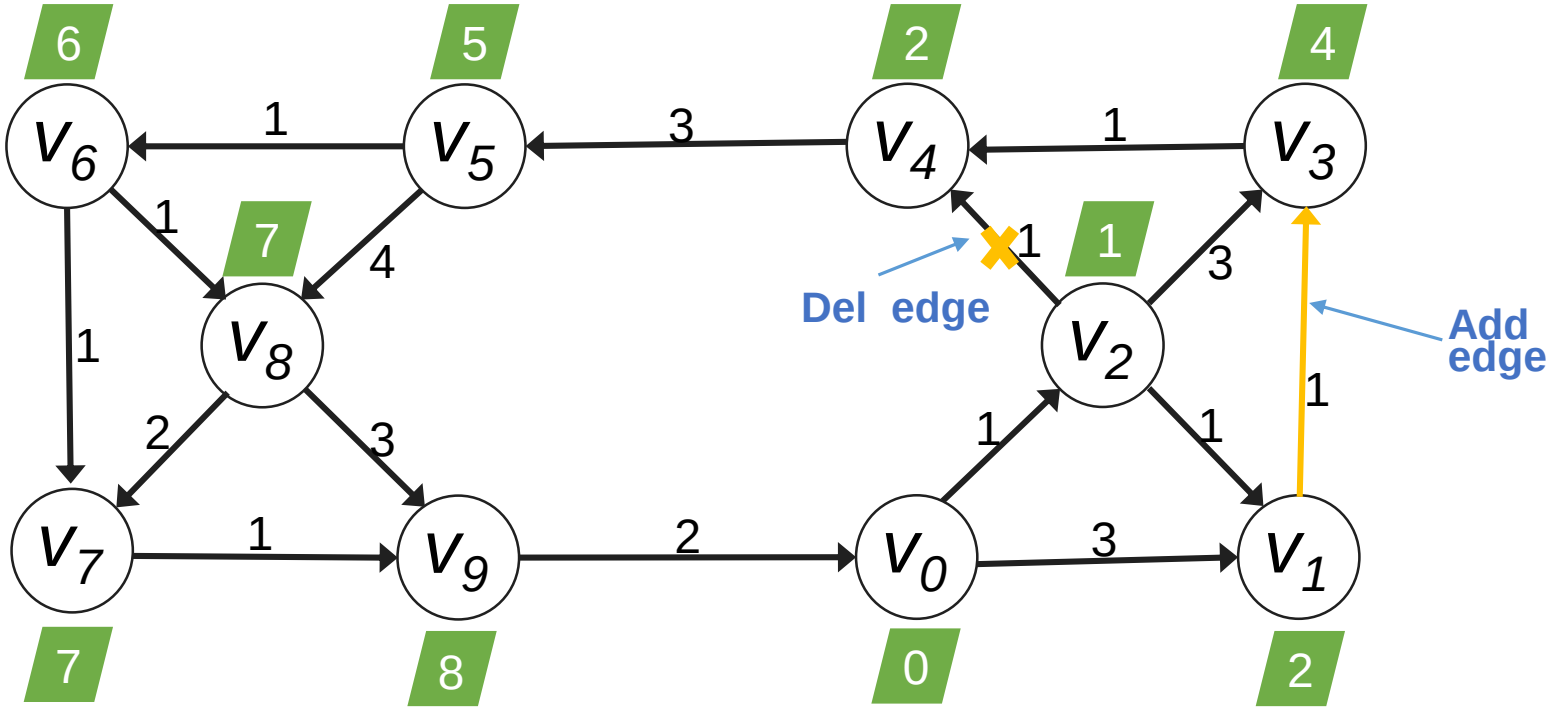
Example：Incremental SSSP

Graph updates
Del edge (v_2, v_4, 1)
Add edge (v_1, v_3, 1)



(b) Updated graph G’

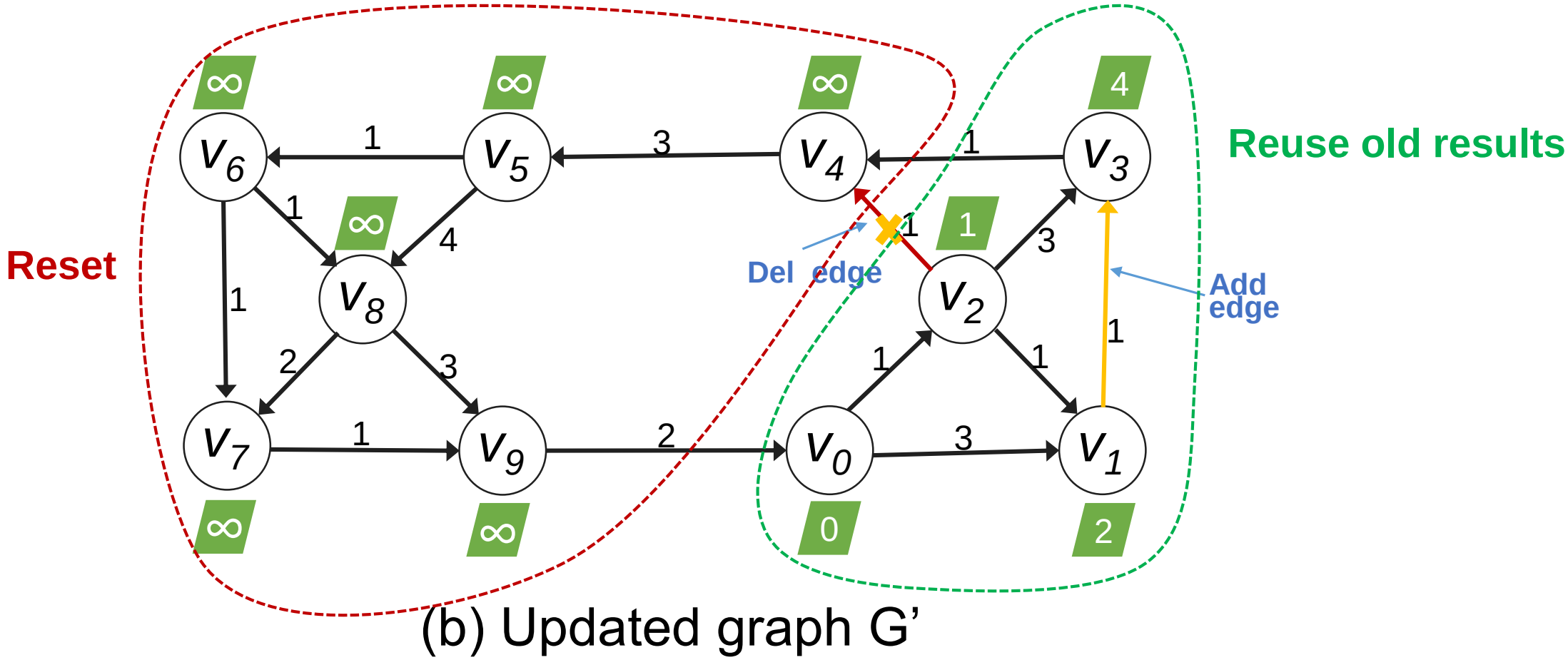
Example：Incremental SSSP



(b) Updated graph G’

Old results should be updated

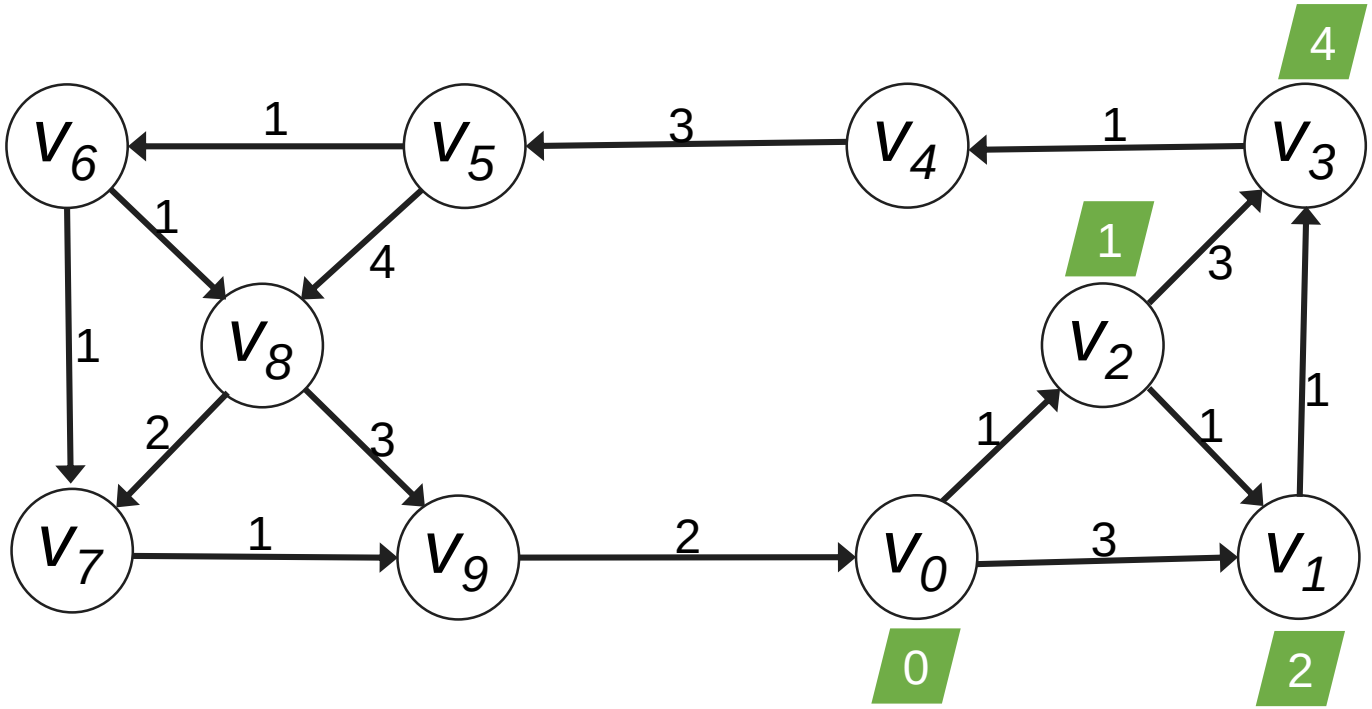
Example：Incremental SSSP



(b) Updated graph G’

Old results should be updated

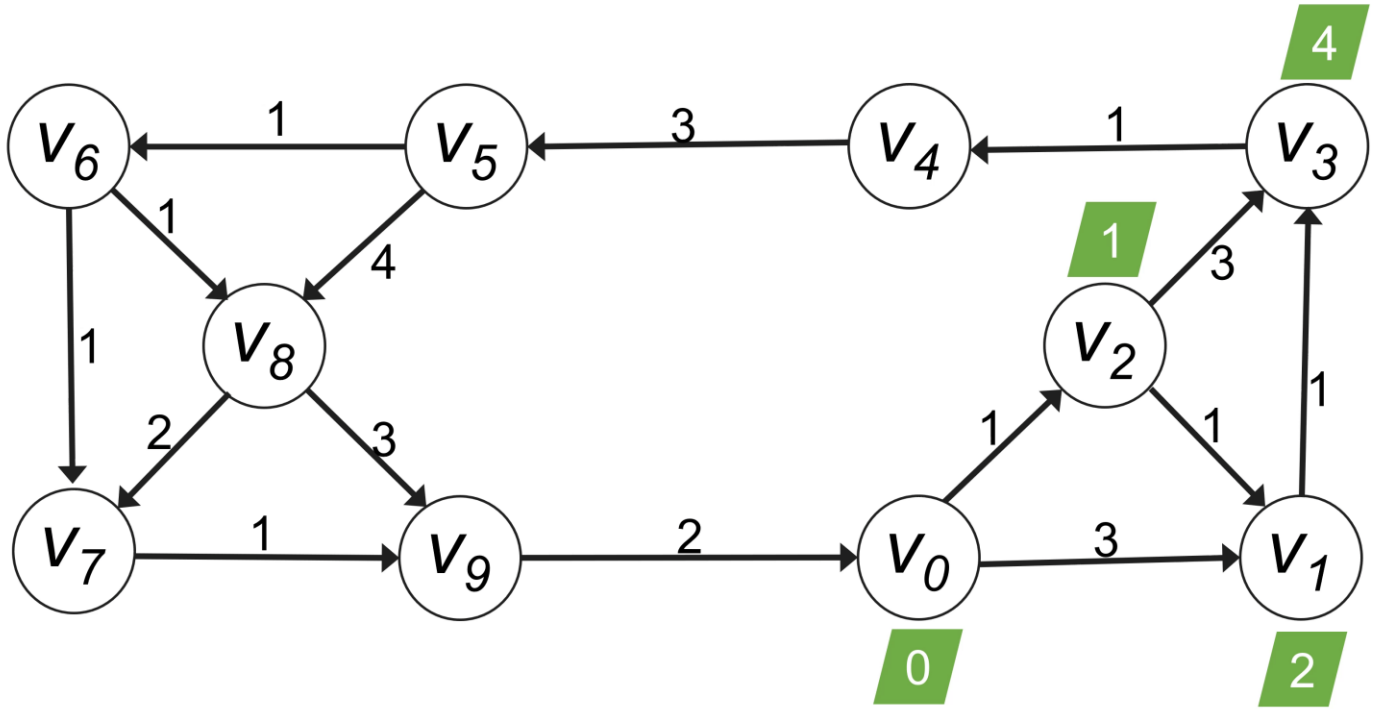
Example：Incremental SSSP



(b) Updated graph G’

Recompute to reach convergence on the new graph

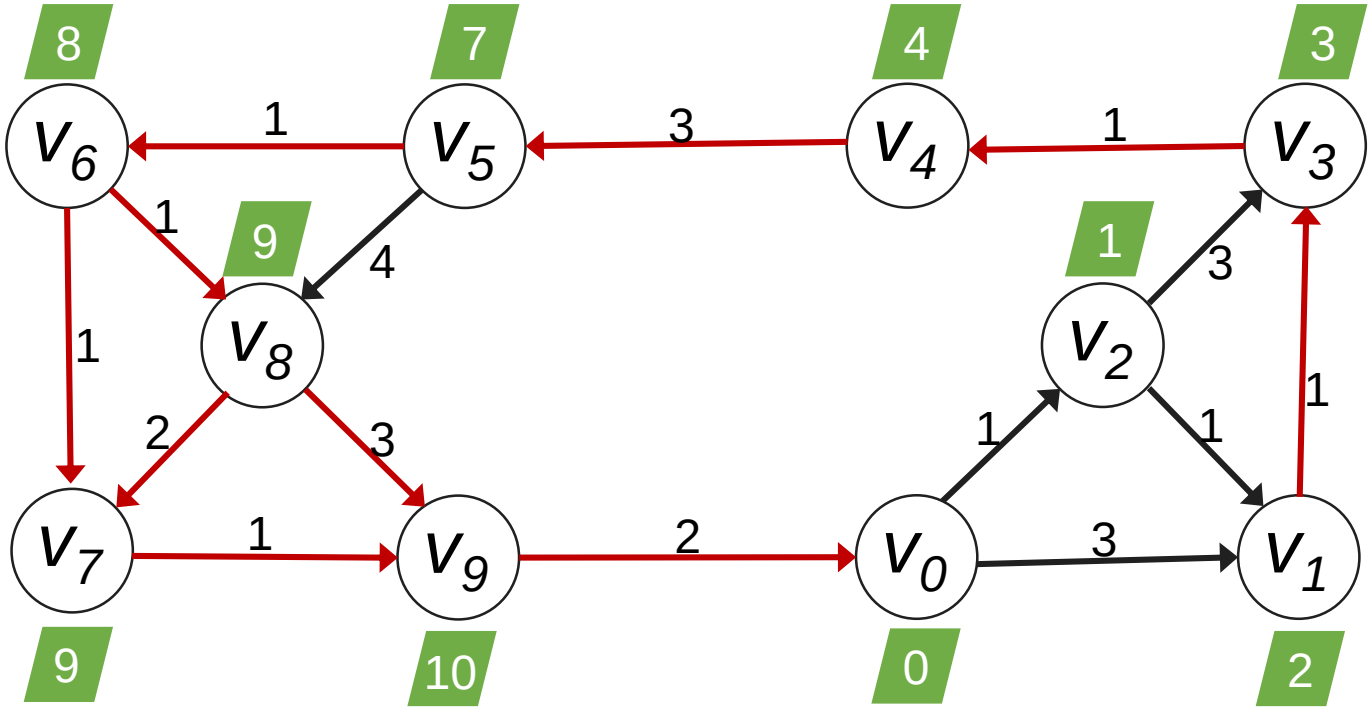
Example：Incremental SSSP



(b) Updated graph G’

Recompute to reach convergence on the new graph

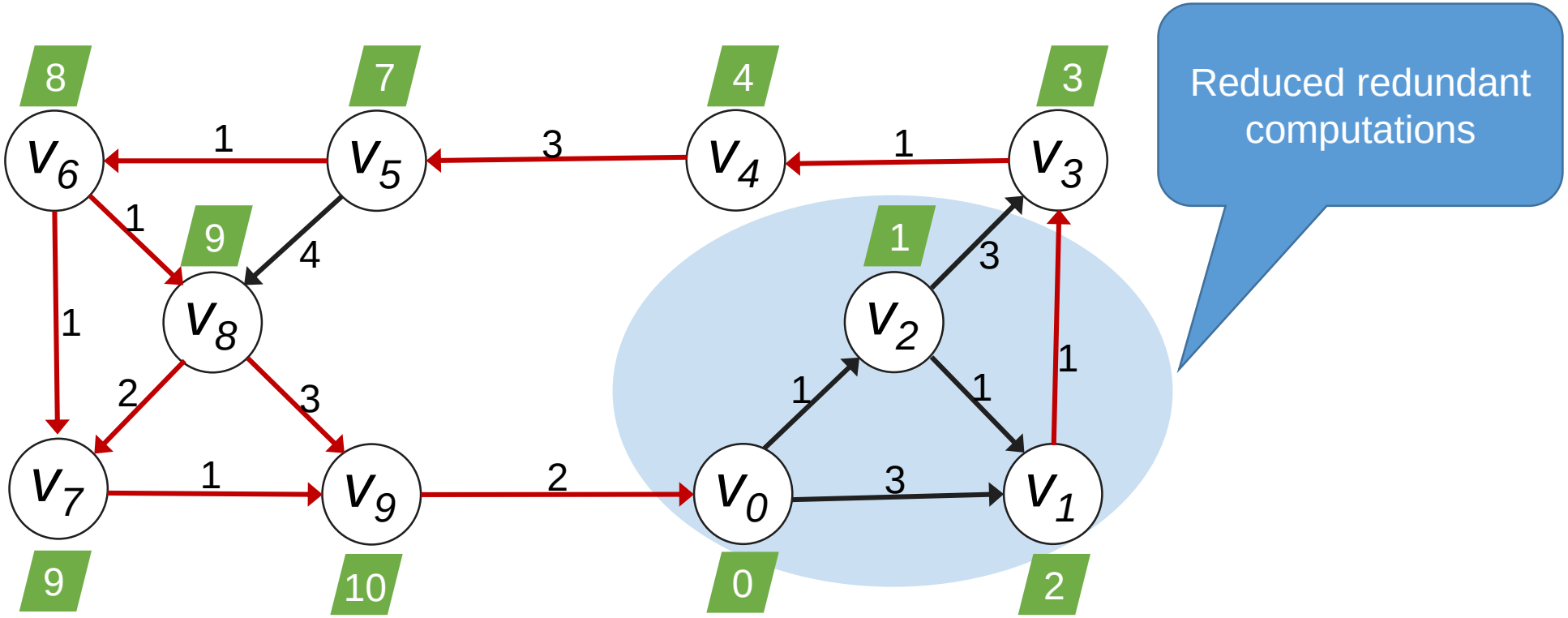
Example：Incremental SSSP



(b) Updated graph G’

The red edges mean they have been activated at least once.

Example：Incremental SSSP



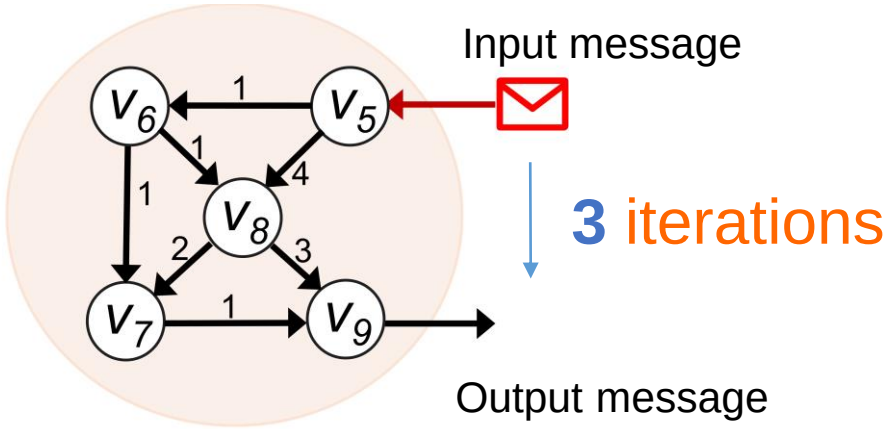
(b) Updated graph G’

The red edges mean they have been activated at least once.

Drawback

- **Update messages go through dense regions requires multiple iterations.**

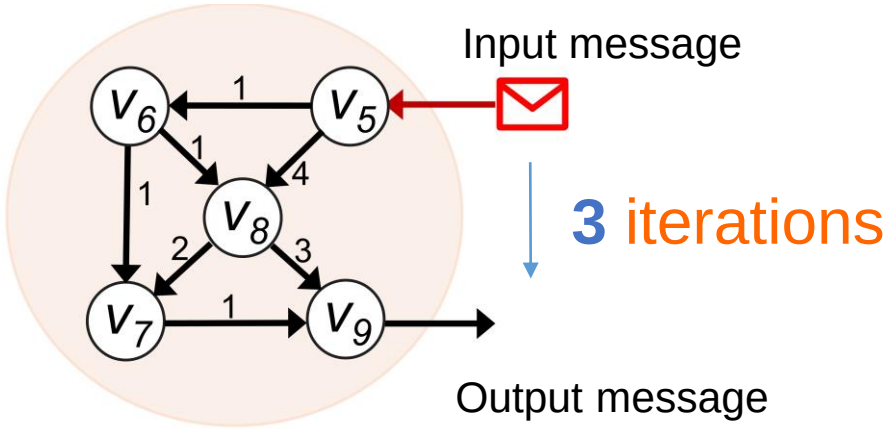
Require multiple iterations



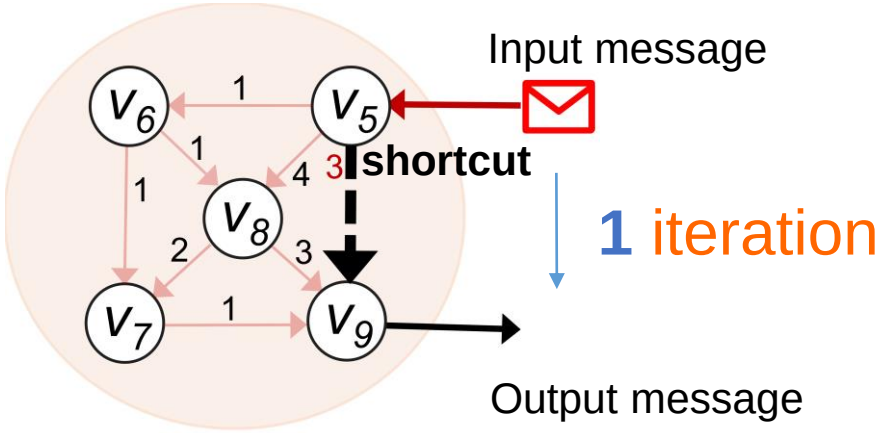
Solution

- **Update messages go through dense regions requires multiple iterations.**

Require multiple iterations

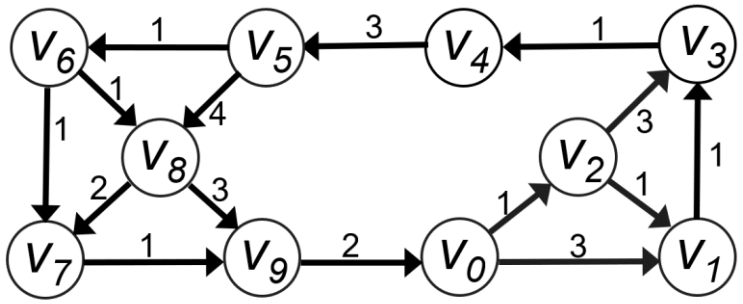


Require One iteration



Drawback

- **Update messages are propagated widely during iterations.**

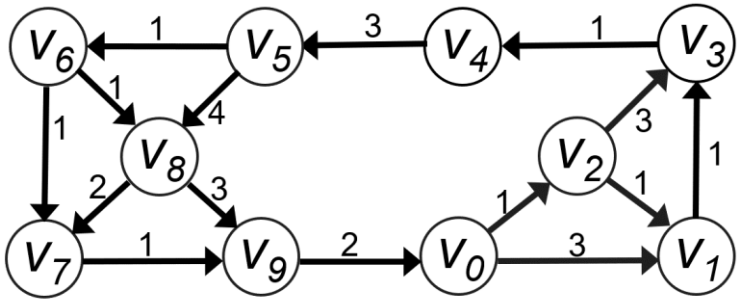


(a) Results of running SSSP on G'

Most vertices

Drawback

- **Update messages are propagated widely during iterations.**



(a) Results of running SSSP on G'

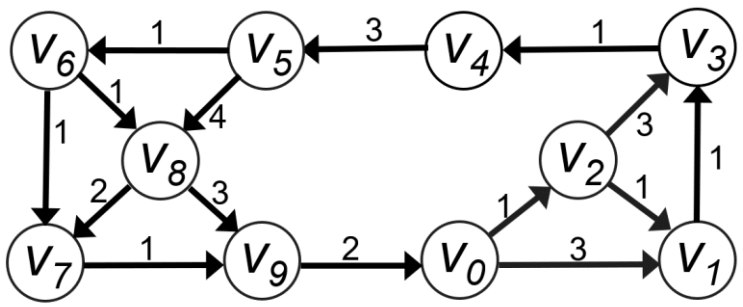
Most vertices

Can iteration be limited to a small range?

Solution

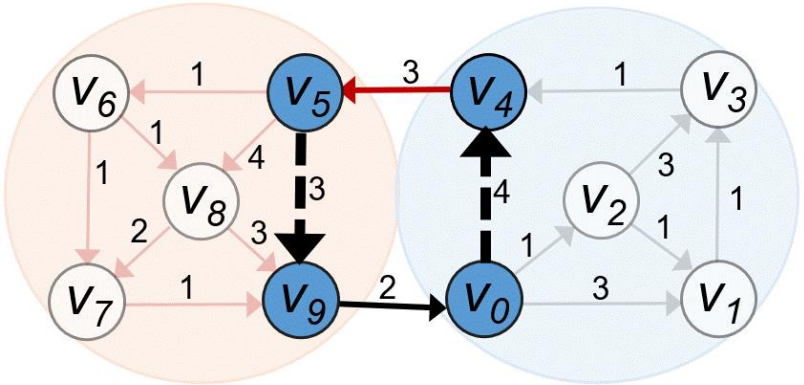
- **Update messages are propagated widely during iterations.**

Iteration



Most vertices

Iteration

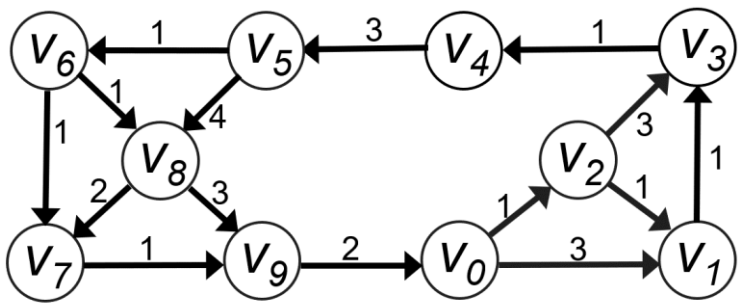


Only key vertices

Solution

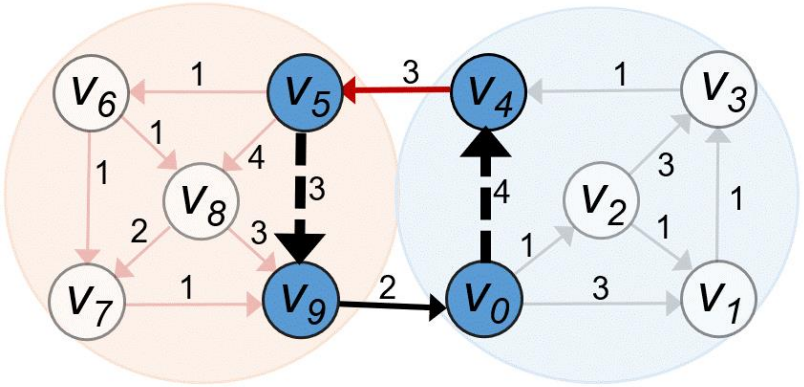
- **Update messages are propagated widely during iterations.**

Iteration



Most vertices

Iteration

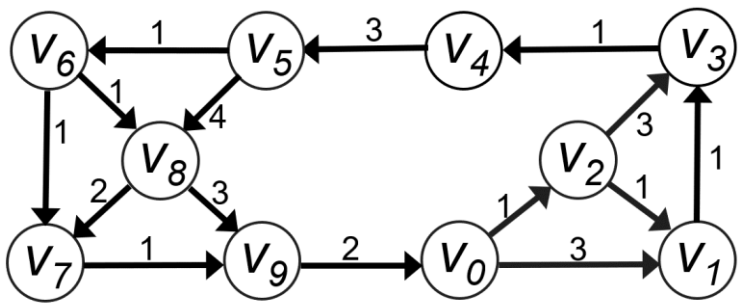


Only key vertices

Solution

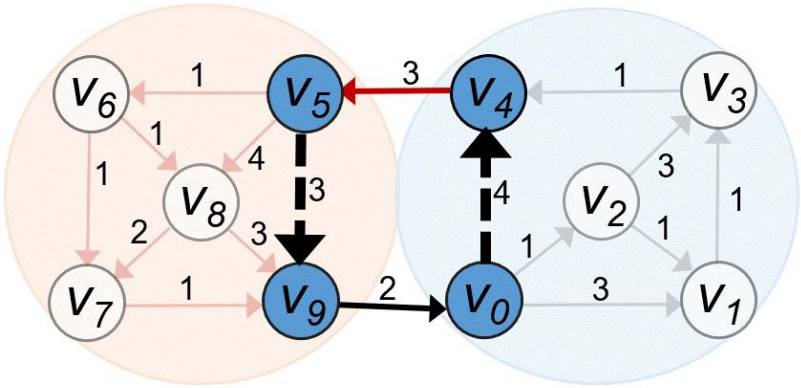
➢ **Update messages are propagated widely during iterations.**

Iteration



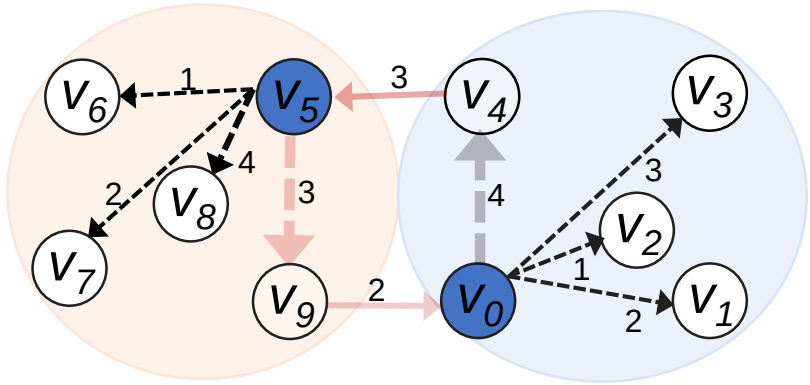
Most vertices

Iteration



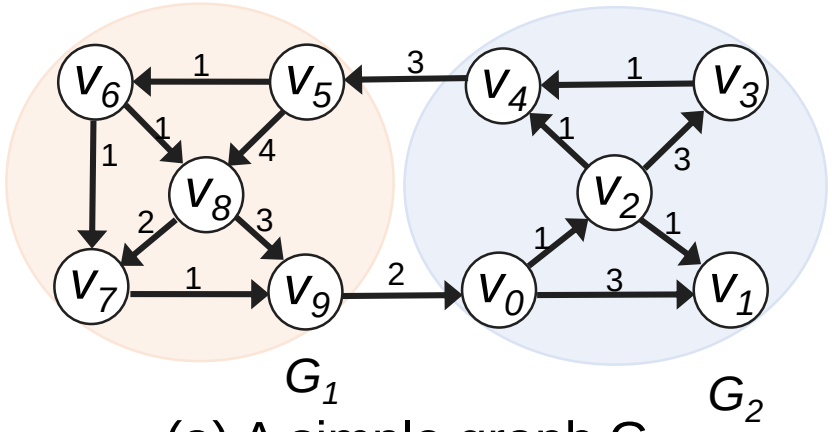
Only key vertices

Non-iteration

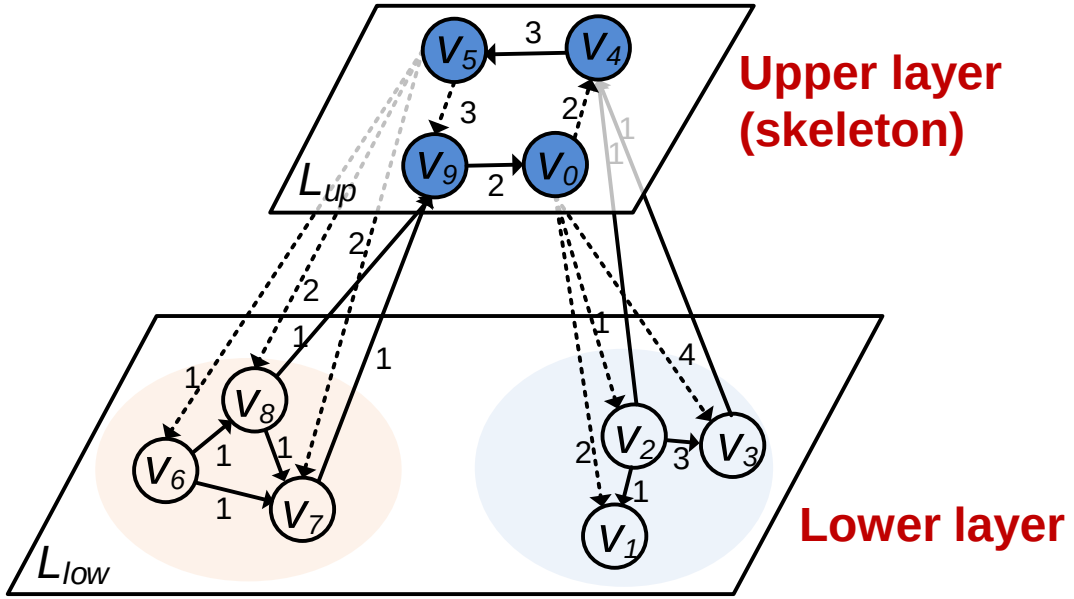


Internal vertices

Layph

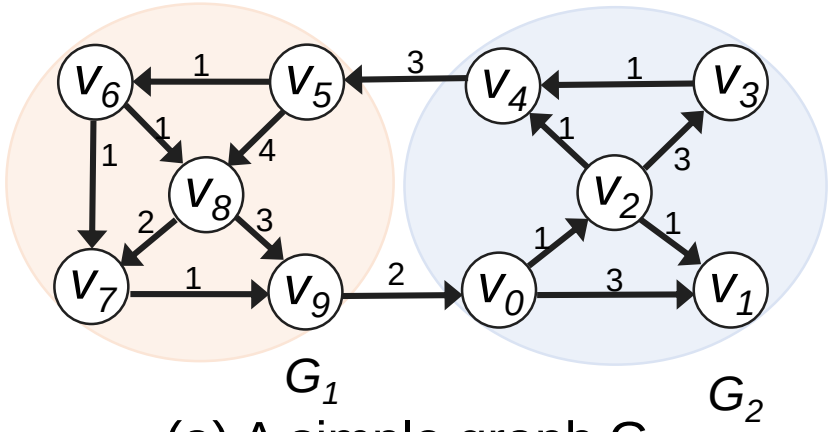


(a) A simple graph G

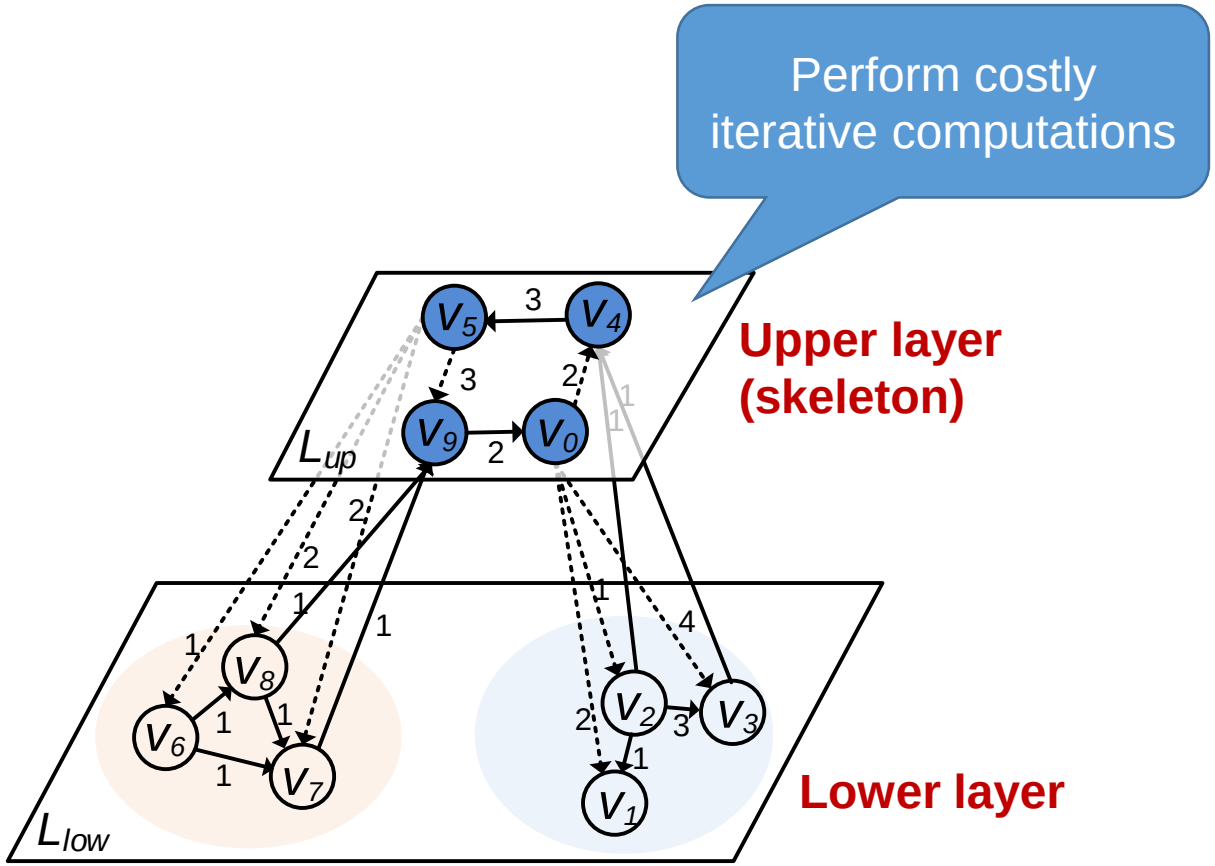


(b) Layered G

Layph

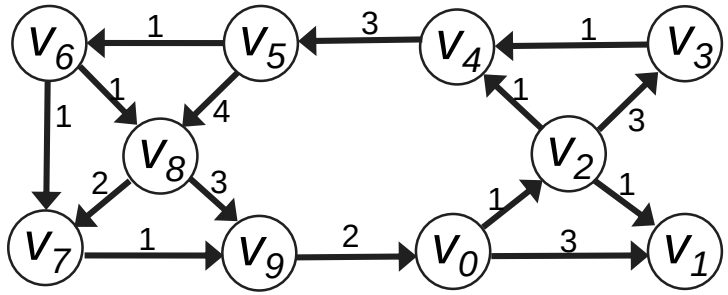


(a) A simple graph G

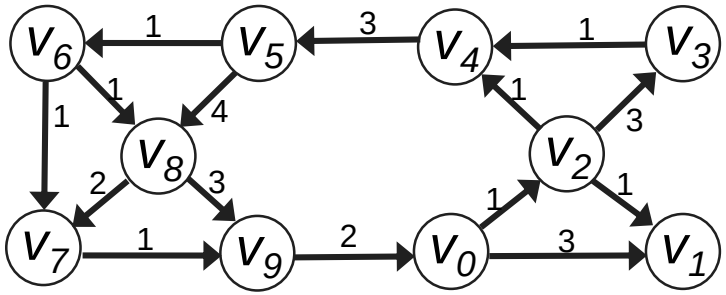


(b) Layered G

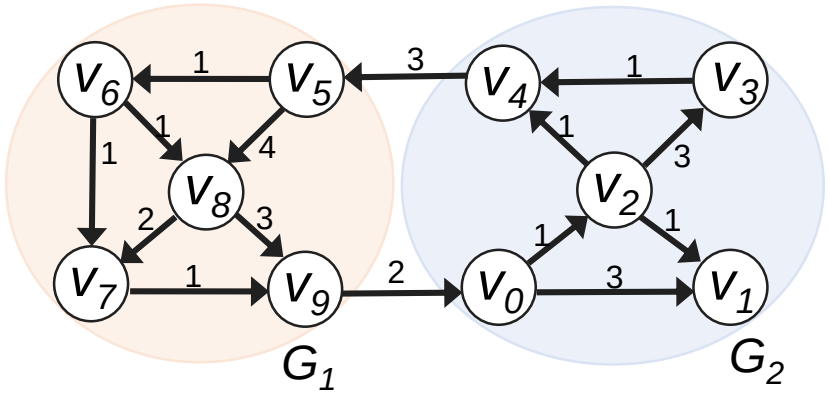
Layered graph construction



Layered graph construction

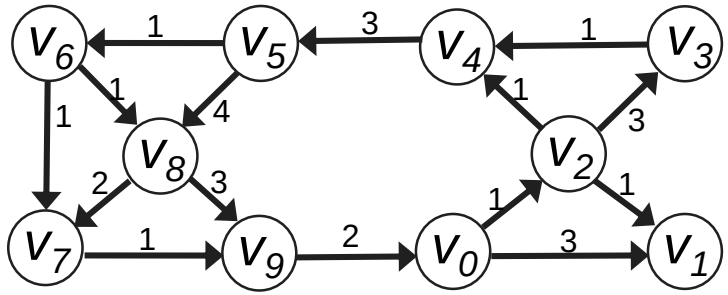


Find dense subgraphs

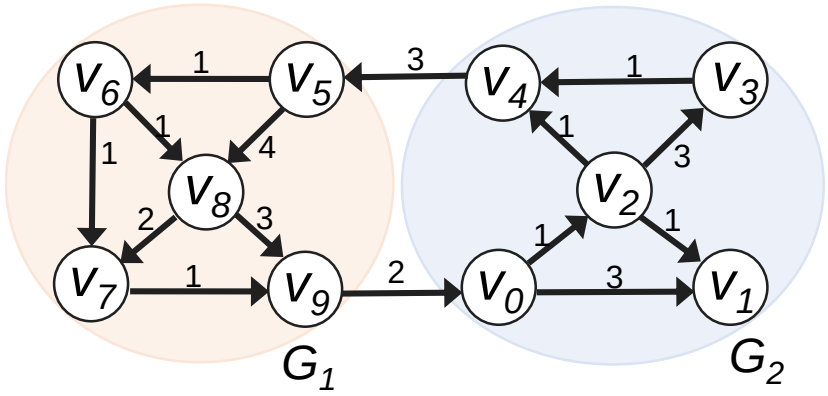


Existing subgraph discovery or community discovery algorithms

Layered graph construction

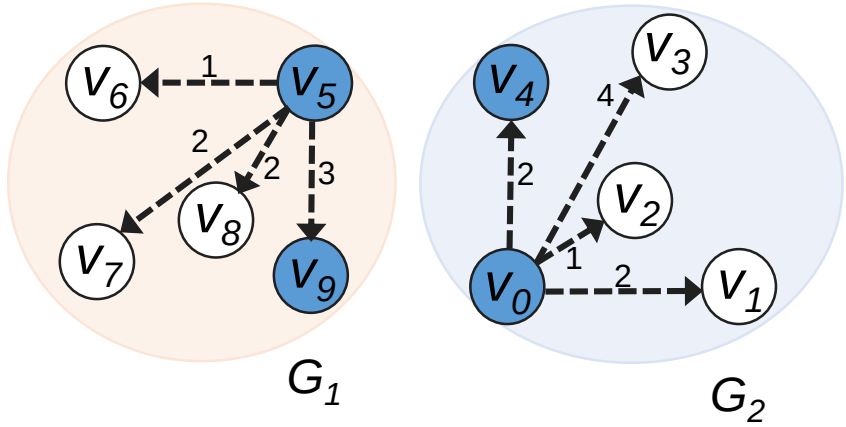


Find dense subgraphs



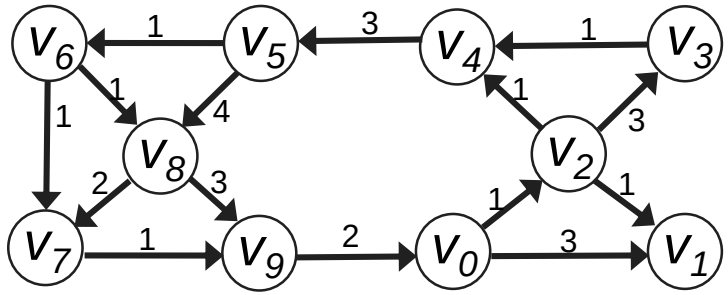
Existing subgraph discovery or community discovery algorithms

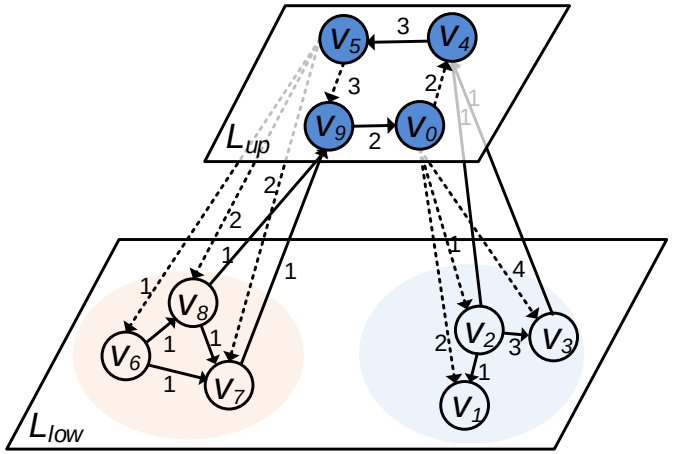
Compute shortcuts



$$\vec{w}_{u,v} = \mathcal{G}_v\Big(\bigcup_{k=1}^{\infty}(\mathcal{G}_{V_i} \circ \mathcal{F}_{E_i})^k(m_u)\Big)$$

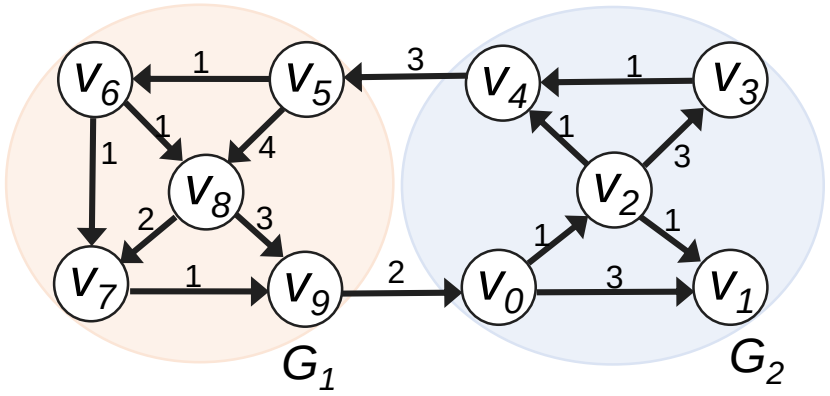
Layered graph construction



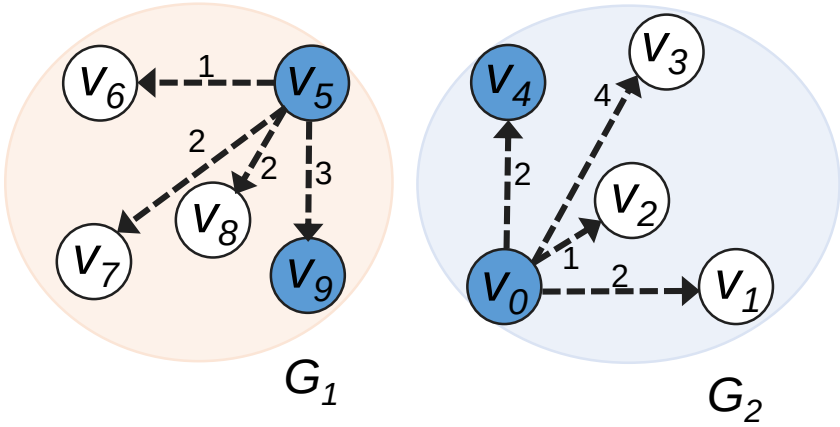


Find dense subgraphs

Layer graph



Compute shortcuts



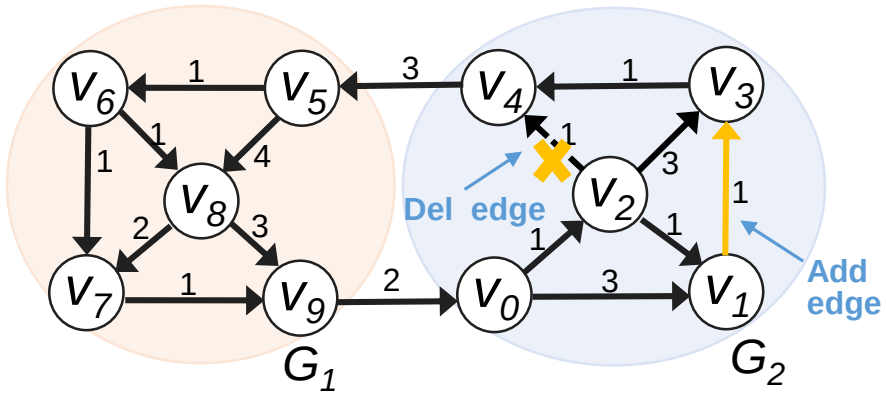
Existing subgraph discovery or community discovery algorithms

$$\vec{w}_{u,v} = \mathcal{G}_v\Big(\bigcup_{k=1}^{\infty}(\mathcal{G}_{V_i} \circ \mathcal{F}_{E_i})^k(m_u)\Big)$$

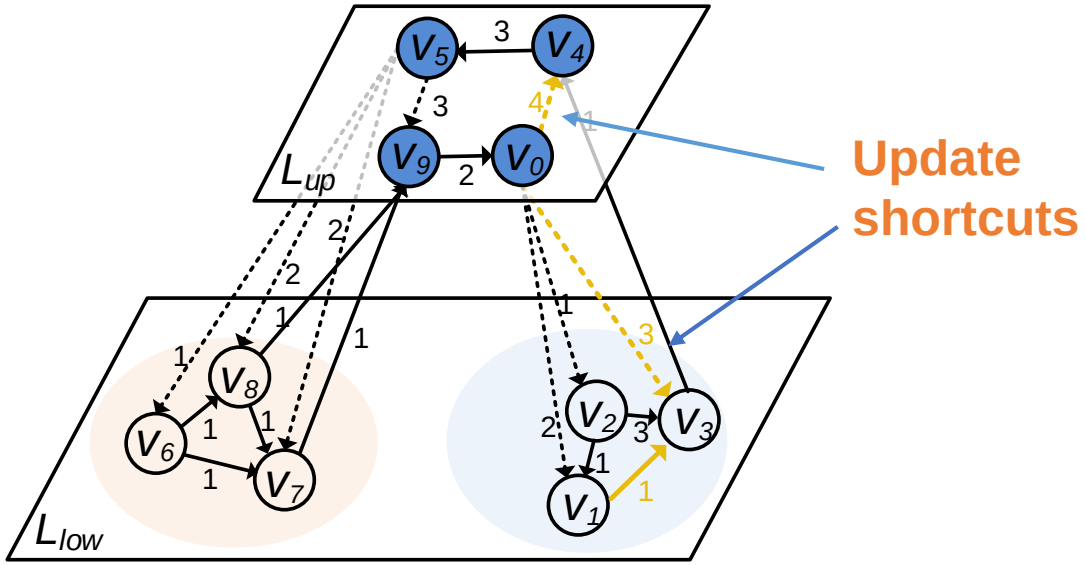
Incremental processing with layered graph

- Layered graph update
- Revision messages deduction
- Messages Upload
- Iterative Computation On The Upper Layer
- Revision Messages Assignment

Layered graph update



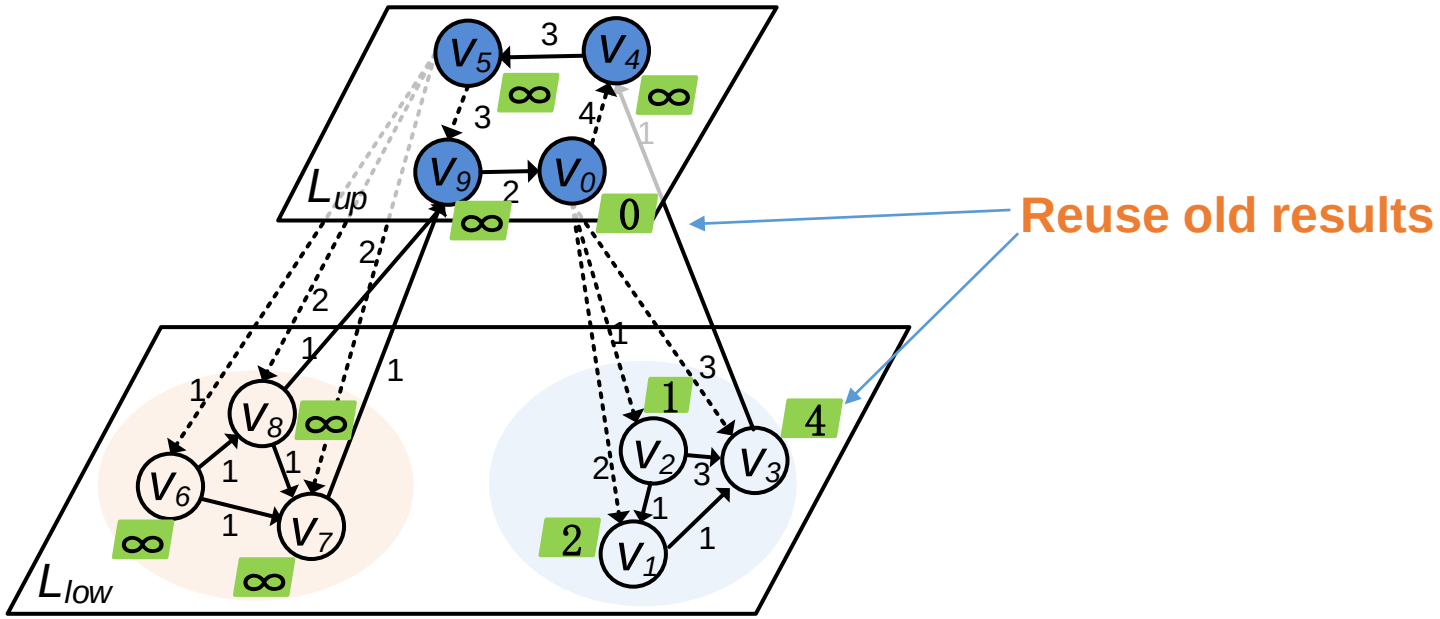
(a) Updated graph G'



(b) Layered G'

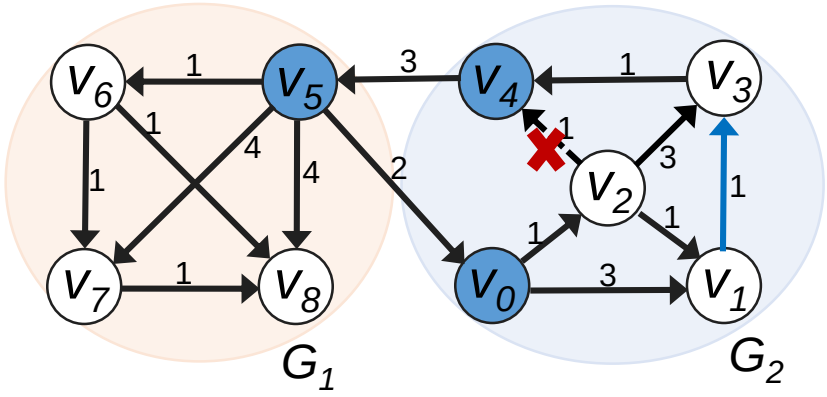
Incrementally and **independently** update affected regions based on graph changes.

Revision messages deduction

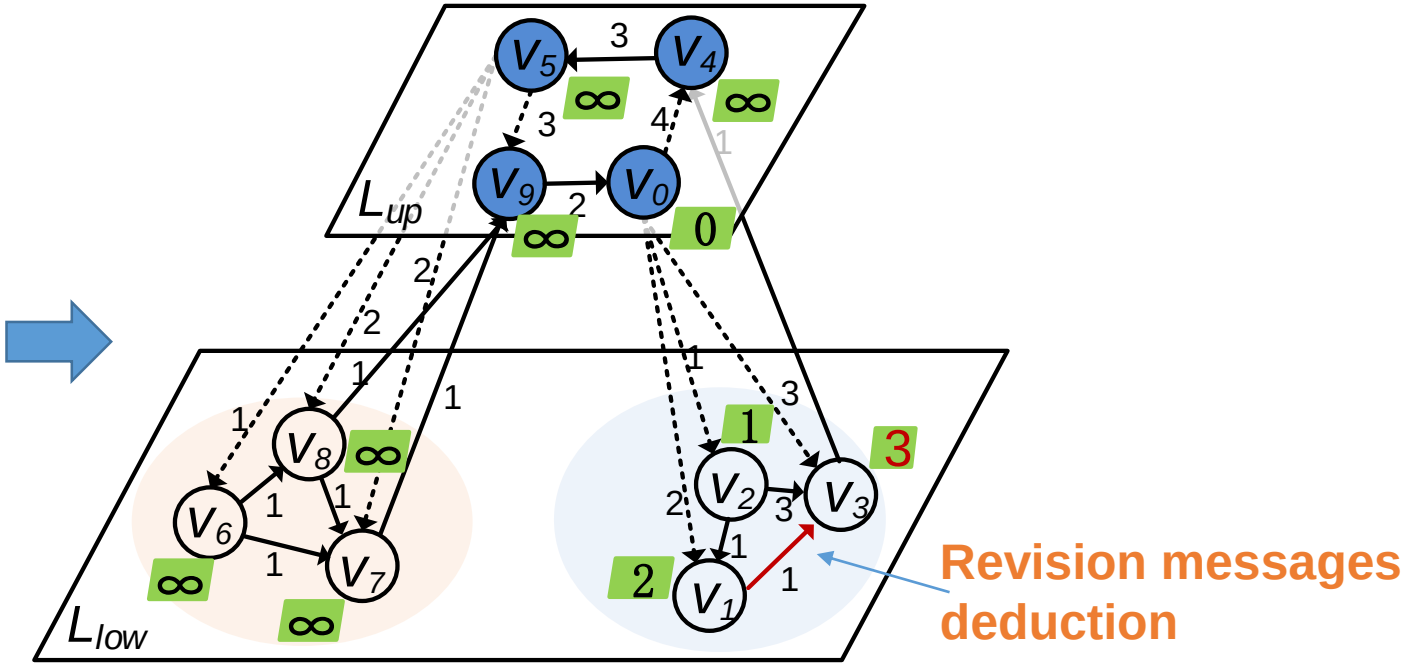


The previous results are reused when the graph is updated.

Revision messages deduction



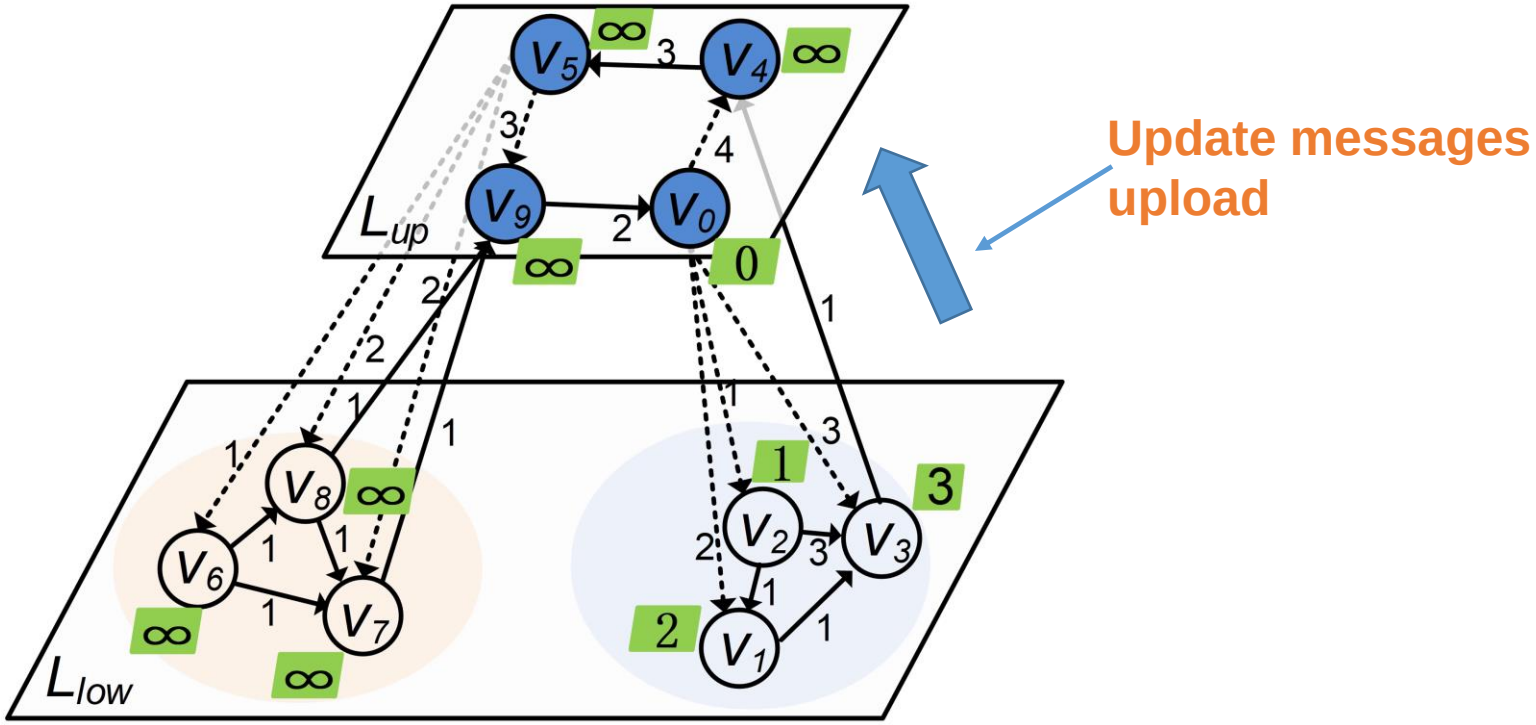
(a) Updated graph G'



(b) Layered G' for SSSP

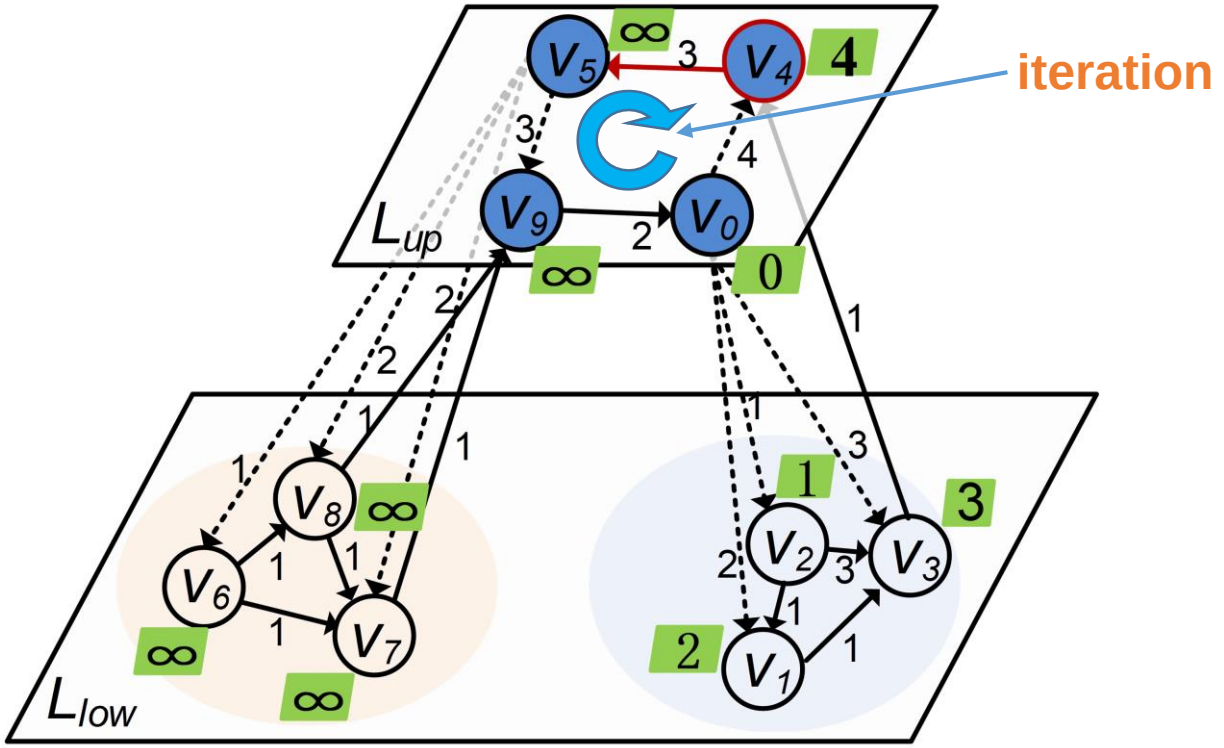
Deduce revision messages using methods from existing incremental systems such as KickStarter [ASPLOS'17], Ingress [VLDB'21], RisGraph [SIGMOD'21], etc.

Messages Upload



Update messages are uploaded from the lower layer to the upper layer.

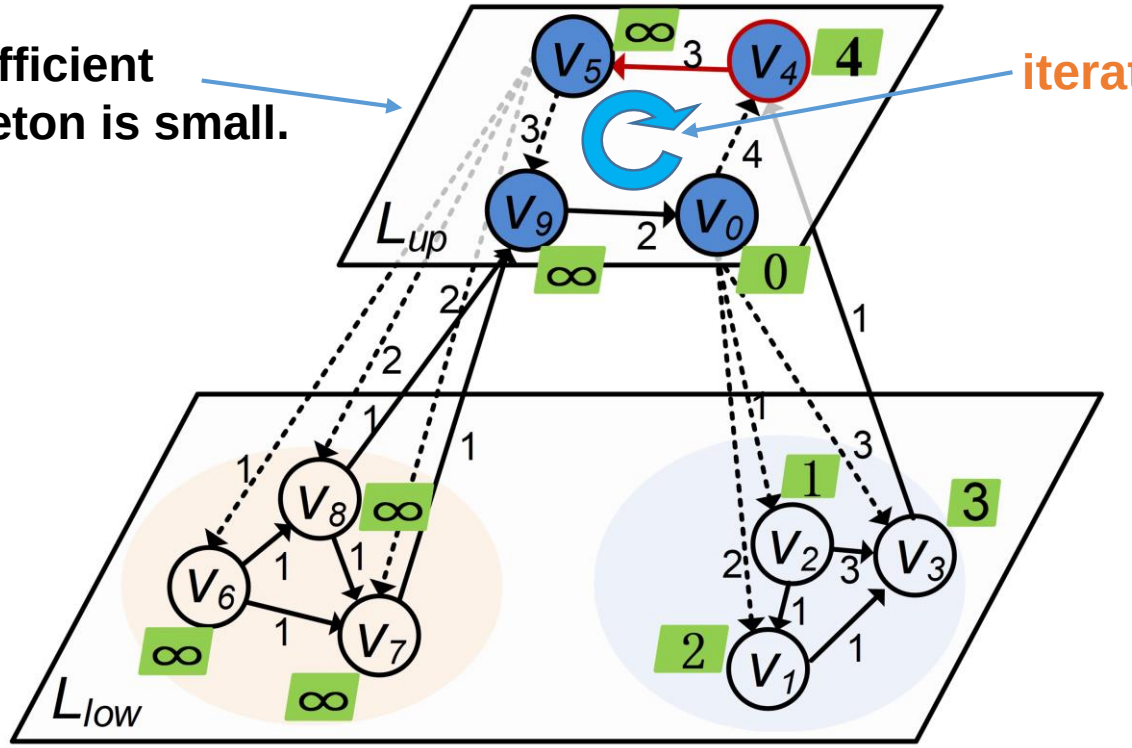
Iterative Computation On The Upper Layer



Apply all update messages to all key vertices on the upper layer.

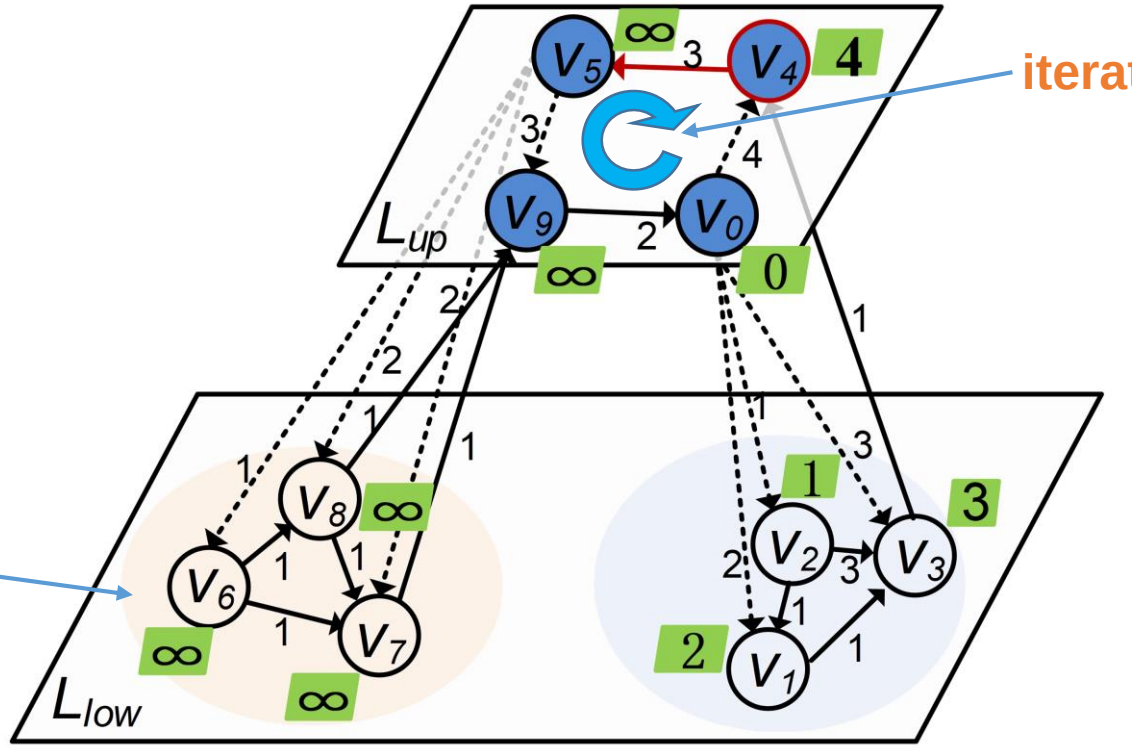
Iterative Computation On The Upper Layer

This iteration is efficient because the skeleton is small.



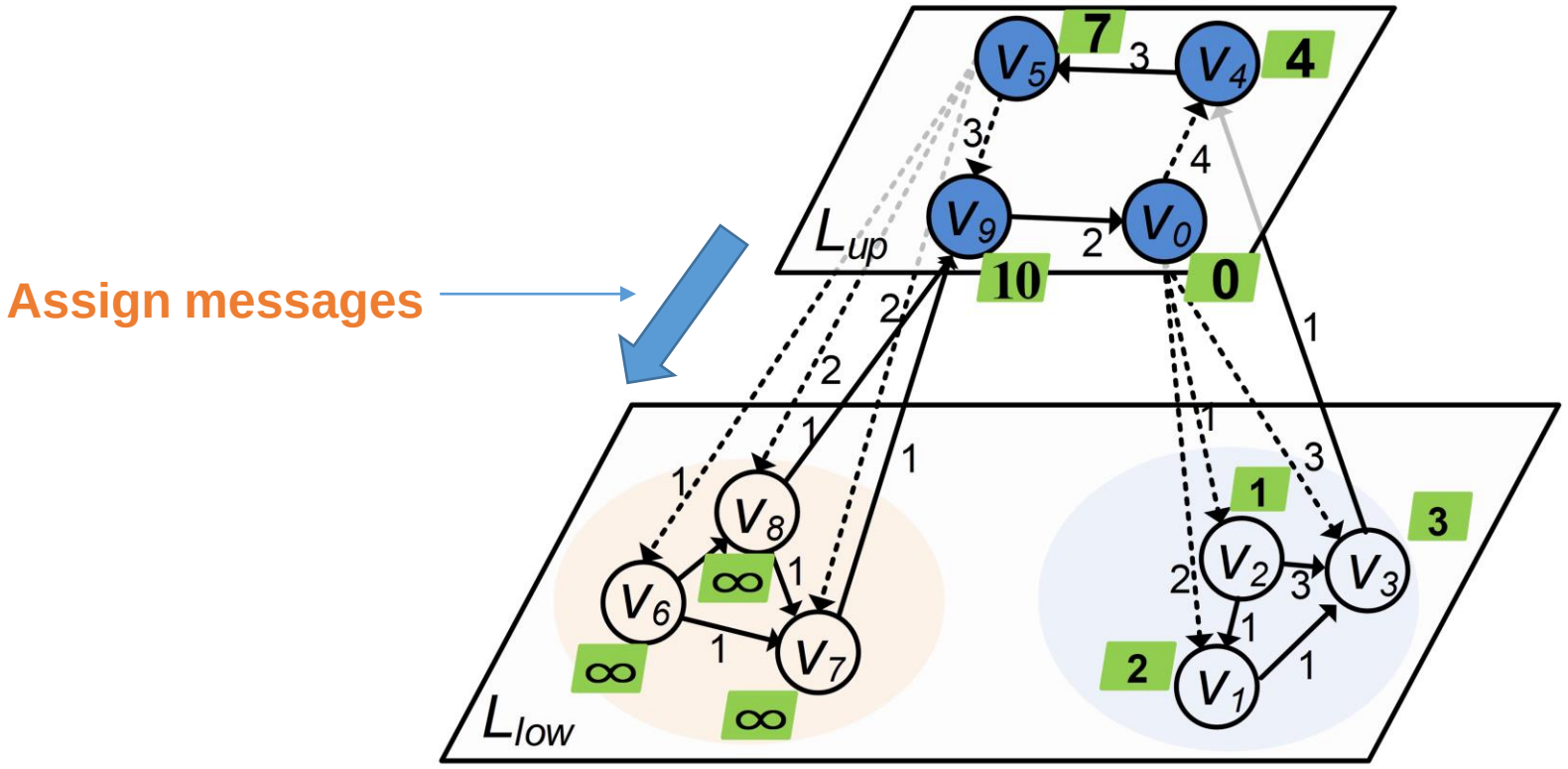
Apply all update messages to all key vertices on the upper layer.

Iterative Computation On The Upper Layer



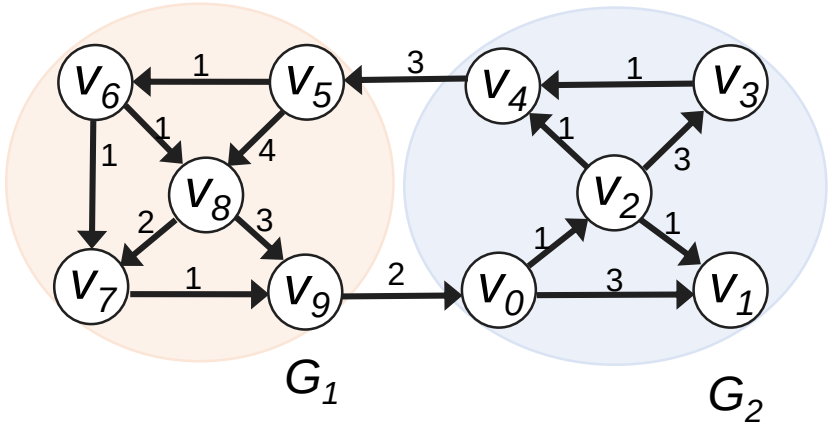
Apply all update messages to all key vertices on the upper layer.

Revision Messages Assignment

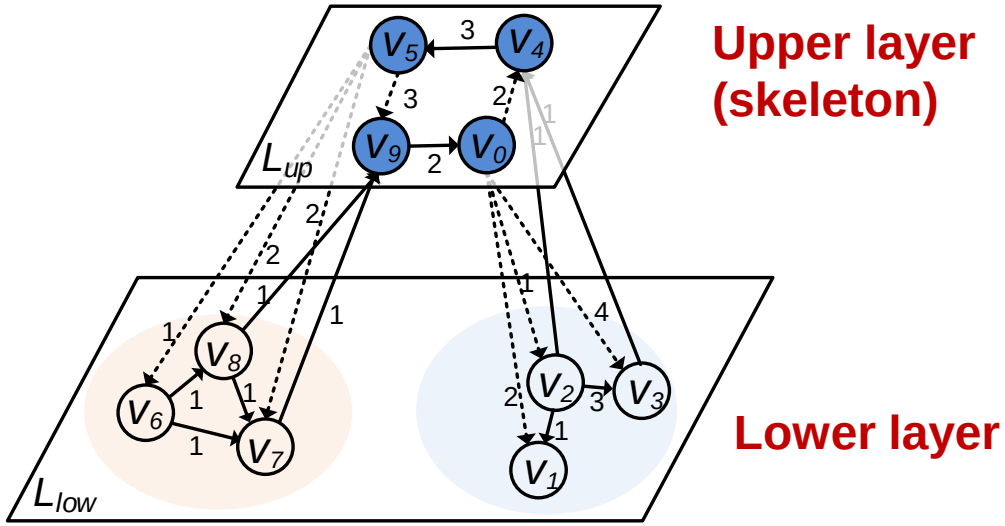


Assign update messages to the vertices on the lower layer.

Advantages of Layph



(a) A simple graph G



(b) Layered G for SSSP

- The iterative computation is transferred from **the original graph** to **a small skeleton**.
- **Limit the propagation range** of update messages in iterations effectively.

Experimental Setting

- Competitors

  KickStarter[ASPLOS'17], **GraphBolt**[EuroSys'19], **DZiG**[EuroSys'21], **Ingress**[VLDB'21], **RisGraph**[SIGMOD'21]

- Workloads

  SSSP, BFS, PageRank, PHP

- Environment

  AliCloud 1ecs.r6.13xlarge (52vCPU, 384G Memory)

- Datasets

Graph	Vertices	Edges	Size
UK-2005 (UK) [24]	39,459,925	936,364,282	16GB
IT-2004 (IT) [25]	41,291,594	1,150,725,436	19GB
SK-2005 (SK) [26]	50,636,154	1,949,412,601	33GB
Sinaweibo (WB) [27]	58,655,850	261,323,450	3.8GB

Overall Performance

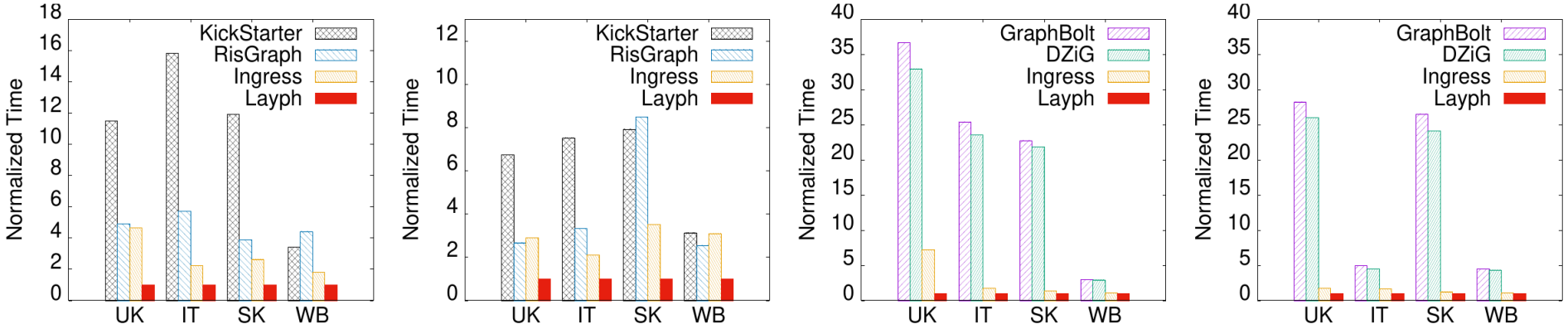


Fig. 1: Response time comparison

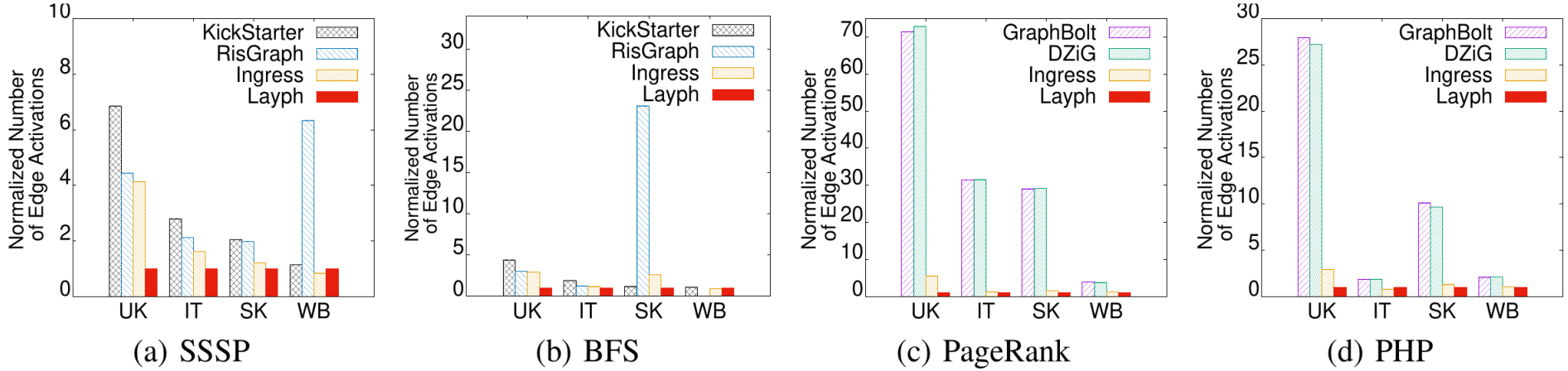
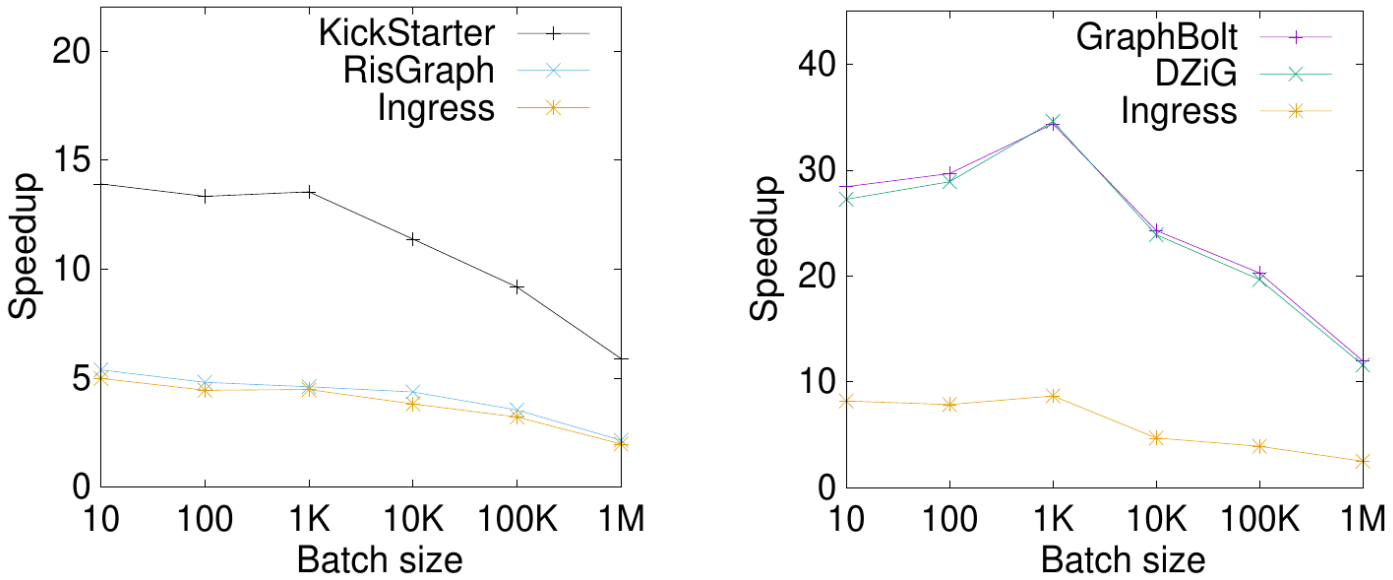


Fig. 2: Number of edge activations comparison

Layph outperforms state-of-the-art incremental graph systems by **9.08×** on average.

Varying Amount of Updates

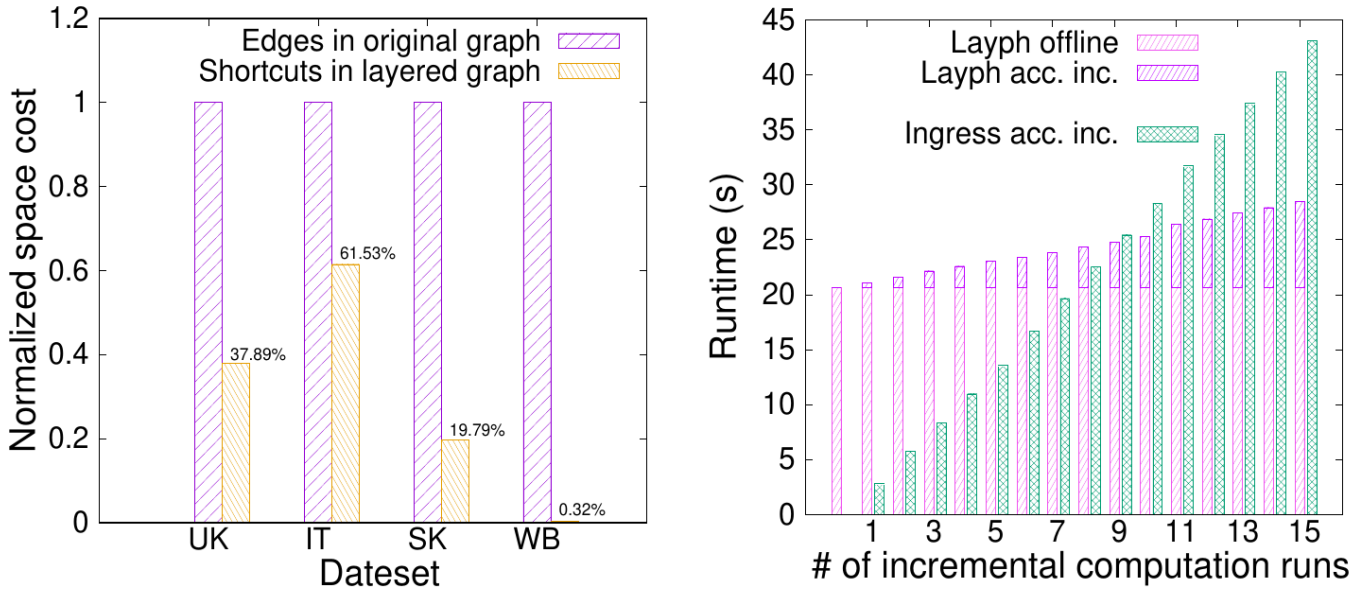


(a) SSSP (b) PageRank

Fig.3: Speedup over competitors when varying batch size

Layph has a significant speedup when the updates are small.

Analysis of Additional Overhead



(a) Space cost (b) Offline preprocessing time

Fig.4: Additional space cost and offline preprocessing time

The offline operation is performed only once but can bring a significant performance gain on each incremental computation.

Conclusion

- Design a layered incremental graph processing framework.

Conclusion

- Design a layered incremental graph processing framework.
- Constrain message propagation in incremental graph processing via layering graph.

Conclusion

- Design a layered incremental graph processing framework.
- Constrain message propagation in incremental graph processing via layering graph.
- Implement a high-performance runtime engine for incremental graph processing.

Thank you for listening