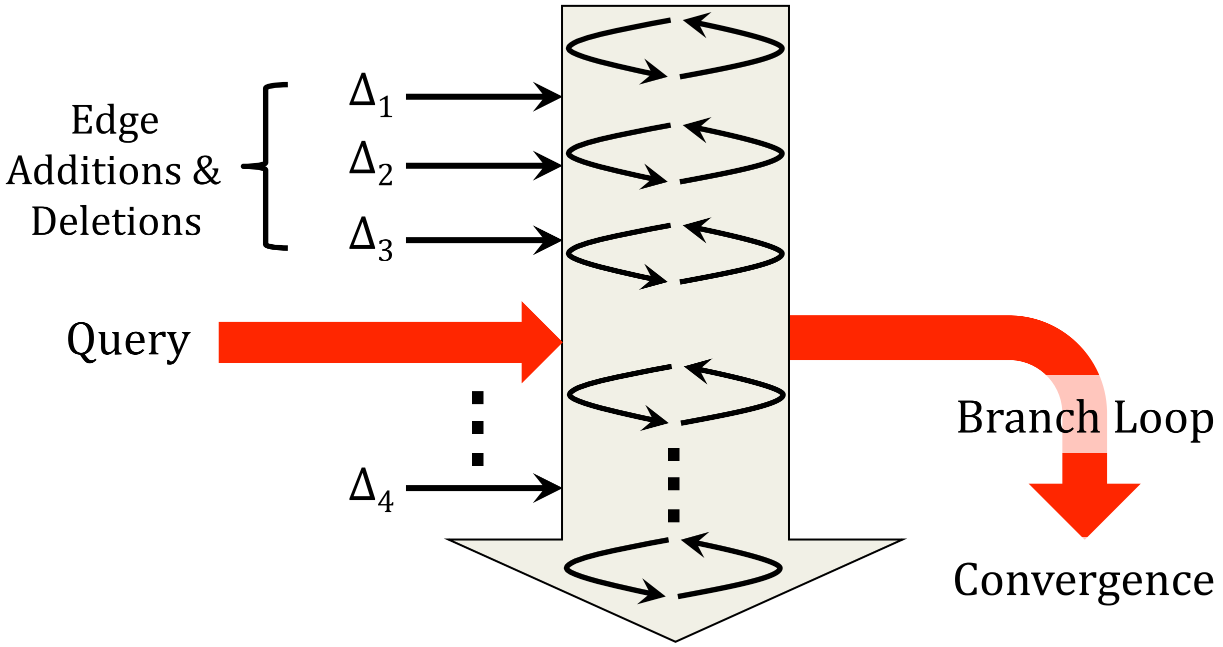
KickStarter: Fast and Accurate Computations on Streaming Graphs via Trimmed Approximations

Keval Vora, Rajiv Gupta and Guoqing Xu

Streaming Graph Processing

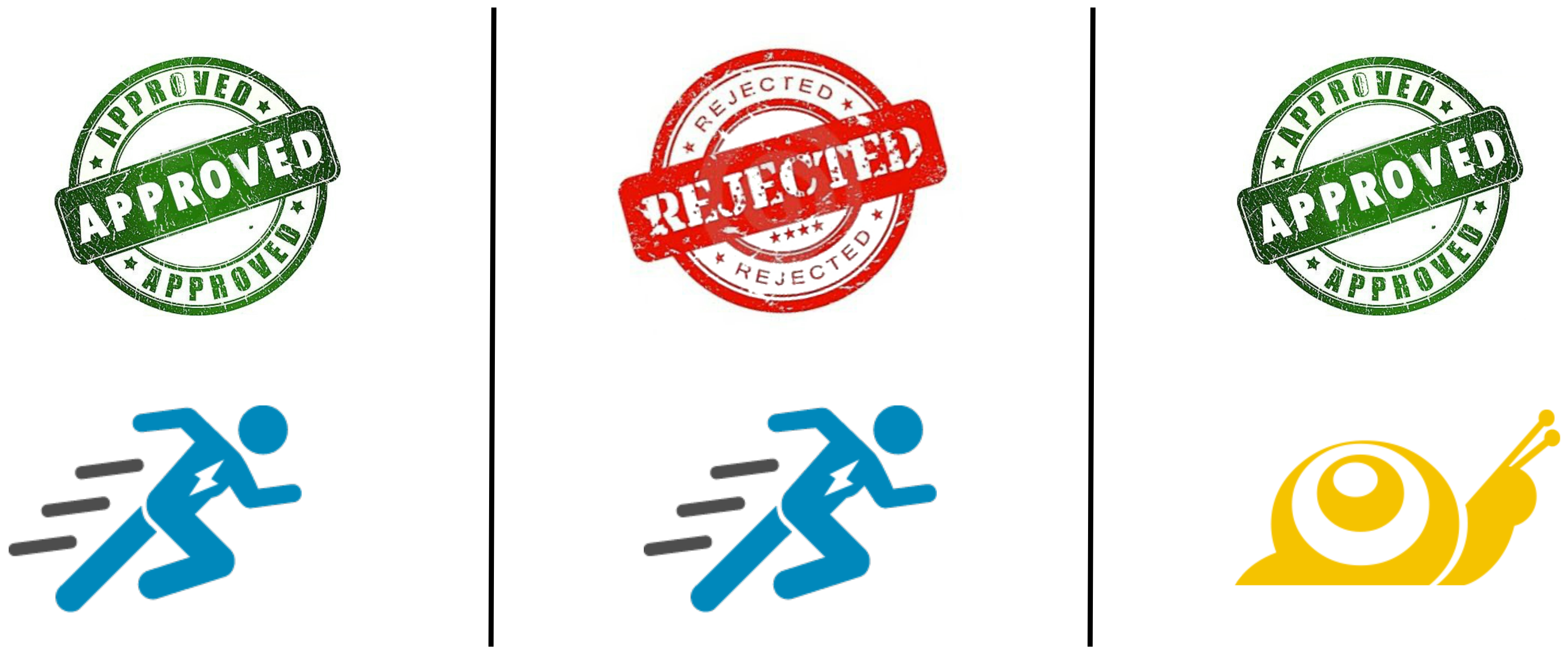
- Graph changes rapidly as computation proceeds
- Incremental processing
  - Maintain "profitable" approximation



Shao, Xiaogang Shi Bin Cui Yingxia, and Yunhai Tong, "Tornado: A System For Real-Time Iterative Analysis Over Evolving Data.", SIGMOD 2016.

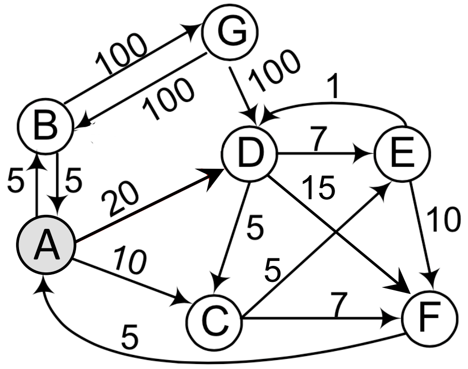
The Good, the Bad and the Ugly

- Correctness & performance



The Good Scenario

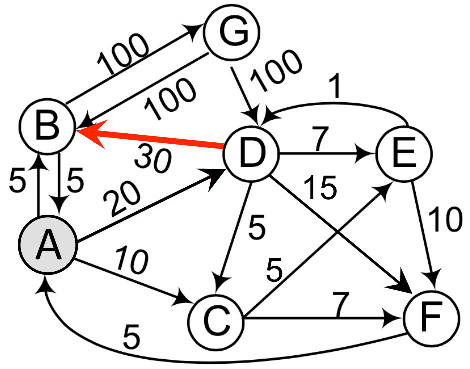
SSWP $v.path \leftarrow \max_{e \in \text{inEdges}(v)} (\min(e.source.path, e.weight))$



A	B	C	D	E	F	G
∞	5	10	20	7	15	5

The Good Scenario

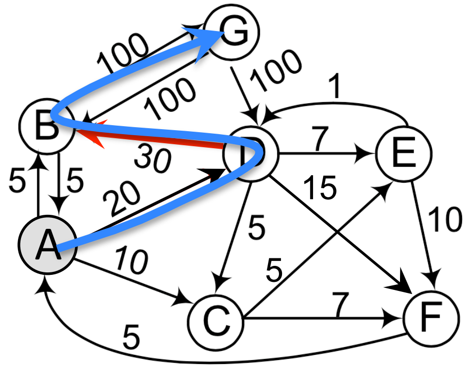
SSWP $v.path \leftarrow \max_{e \in \text{inEdges}(v)} (\min(e.source.path, e.weight))$



A	B	C	D	E	F	G
∞	5	10	20	7	15	5
Add D → B						

The Good Scenario

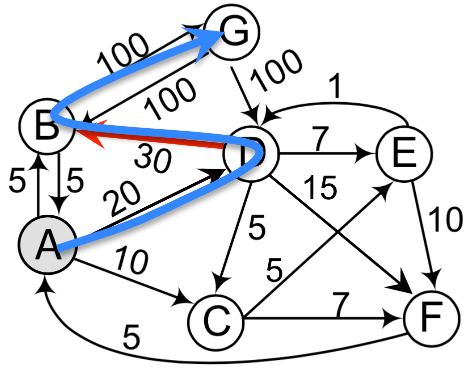
SSWP $v.path \leftarrow \max_{e \in \text{inEdges}(v)} (\min(e.source.path, e.weight))$



A	**B**	**C**	**D**	**E**	**F**	**G**
∞	5	10	20	7	15	5
Add D $\rightarrow$ B						

The Good Scenario

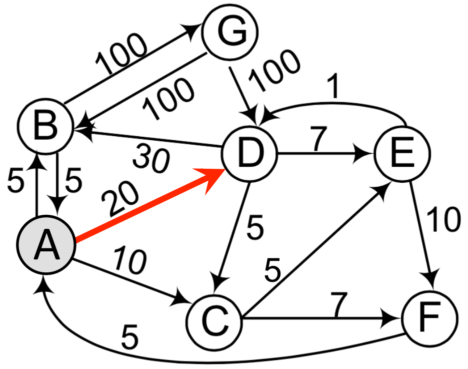
SSWP $v.path \leftarrow \max_{e \in \mathbf{inEdges}(v)} (\min(e.source.path, e.weight))$



A	B	C	D	E	F	G
∞	5	10	20	7	15	5
Add D → B						
∞	20	10	20	7	15	5
∞	**20**	10	20	7	15	**20**

The Bad Scenario

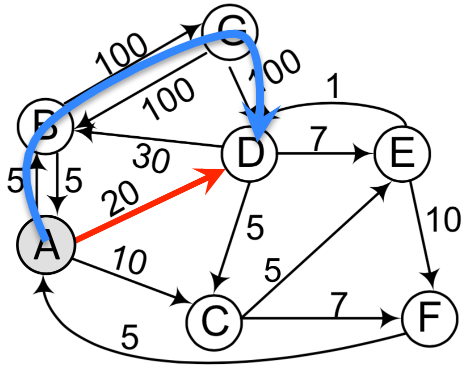
SSWP $v.path \leftarrow \max_{e \in \textbf{inEdges}(v)} (\min(e.source.path, e.weight))$



A	B	C	D	E	F	G
∞	20	10	20	7	7	20
Delete A → D						

The Bad Scenario

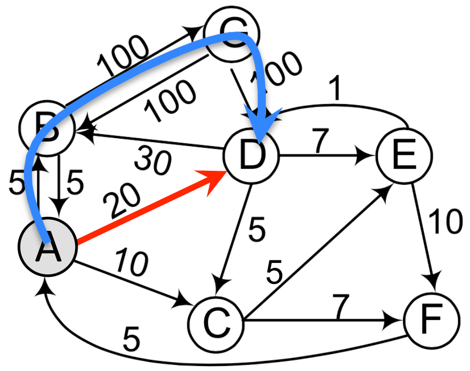
SSWP $v.path \leftarrow \max_{e \in \mathbf{inEdges}(v)} (\min(e.source.path, e.weight))$



A	B	C	D	E	F	G
∞	20	10	20	7	7	20
Delete A → D						

The Bad Scenario

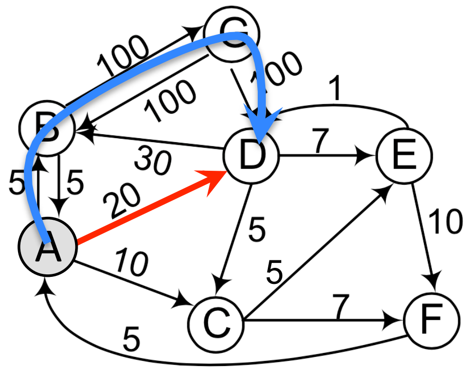
SSWP $v.path \leftarrow \max_{e \in \textbf{inEdges}(v)} (\min(e.source.path, e.weight))$



A	B	C	D	E	F	G
∞	20	10	20	7	7	20
Delete A → D						
…						
∞			20			

The Bad Scenario

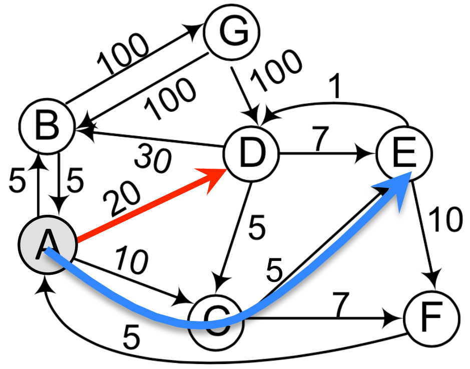
SSWP $v.path \leftarrow \max_{e \in \mathrm{inEdges}(v)} (\min(e.source.path, e.weight))$



A	B	C	D	E	F	G
∞	20	10	20	7	7	20
Delete A → D						
…						
∞	**20**		**20**			**20**

The Bad Scenario

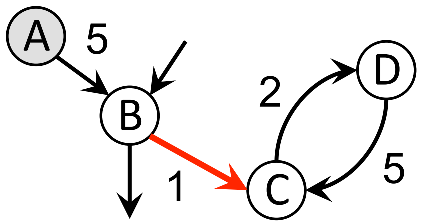
SSWP $v.path \leftarrow \max_{e \in \text{inEdges}(v)} (\min(e.source.path, e.weight))$



A	B	C	D	E	F	G
∞	20	10	20	7	7	20
Delete A → D						
…						
∞	20	10	20	7	7	20

The Ugly Scenario

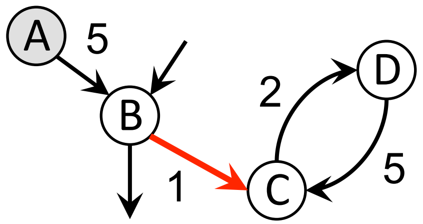
SSSP $v.path \leftarrow \min_{e \in \mathbf{inEdges}(v)} (e.source.path + e.weight)$



A	B	C	D
0	5	6	8
Delete B → C			

The Ugly Scenario

SSSP $$v.path \leftarrow \min_{e \in \textbf{inEdges}(v)} (e.source.path + e.weight)$$

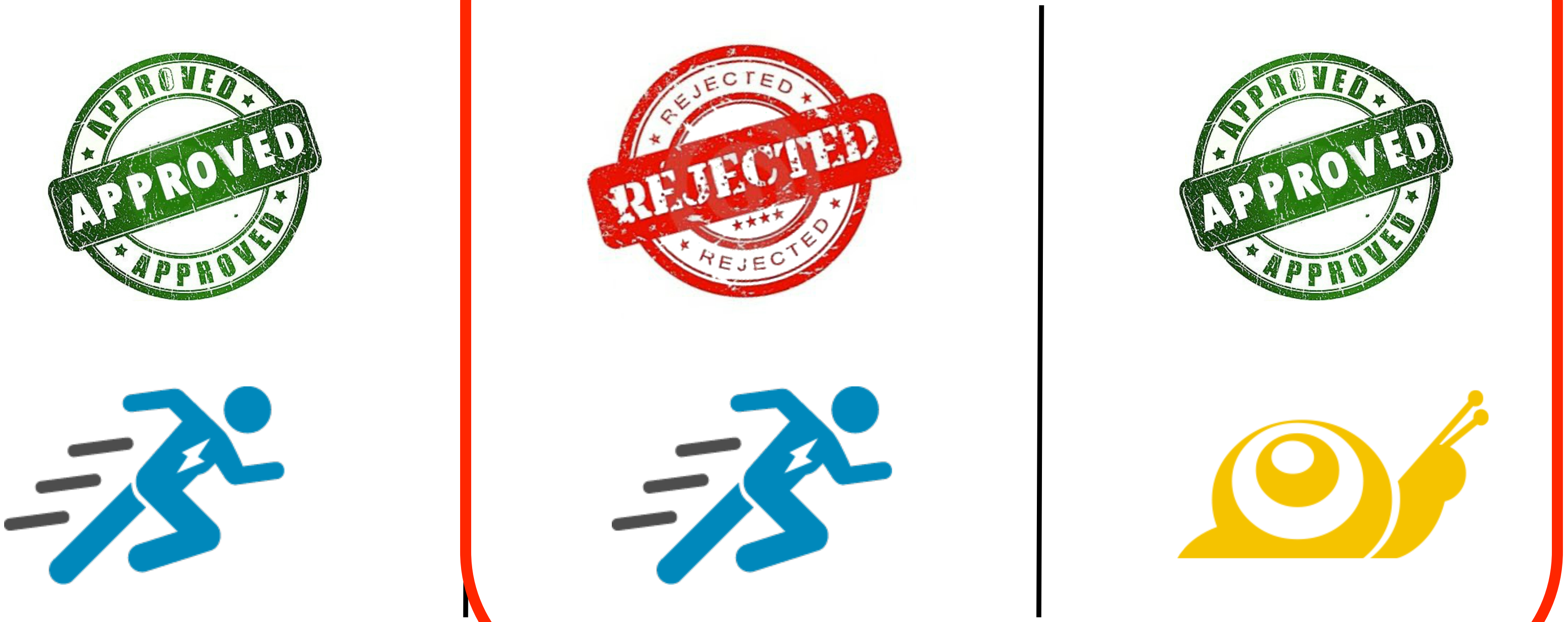


A	B	C	D
0	5	6	8
Delete B → C			
0	5	6	8
0	5	13	8
0	5	13	15
0	5	20	15
0	…	…	…
0	5	MAX	MAX

The Good, the Bad and the Ugly

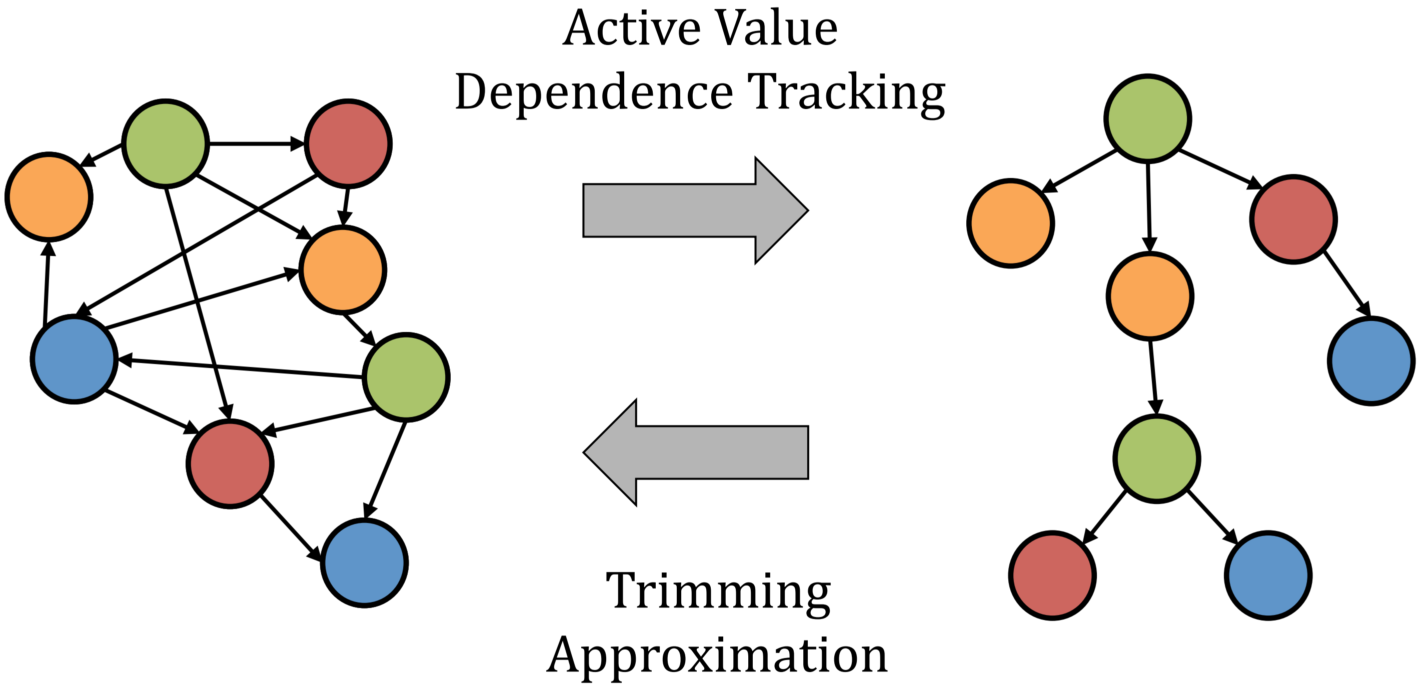
- Correctness & performance

Edge Deletions



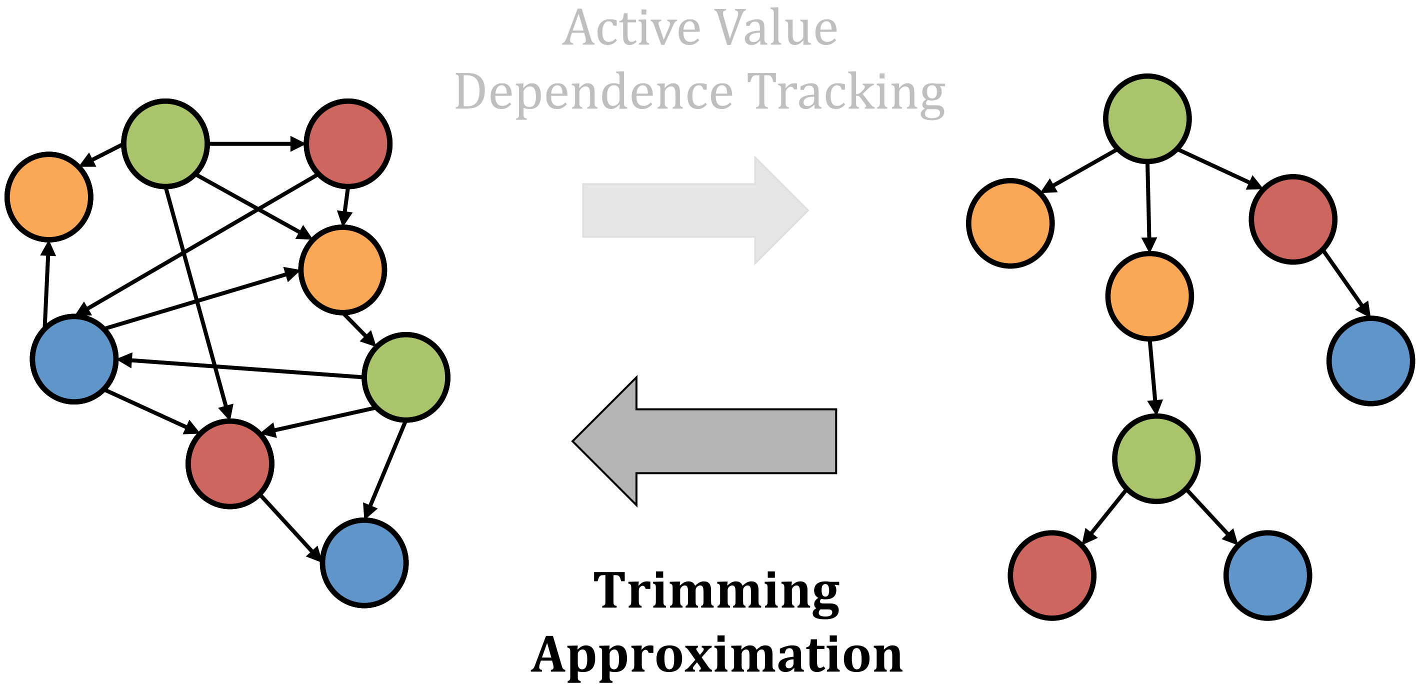
KickStarter

- Maintain value dependences during computation
    - $a \xrightarrow{\text{LT}} b$ iff b's value resulted from a



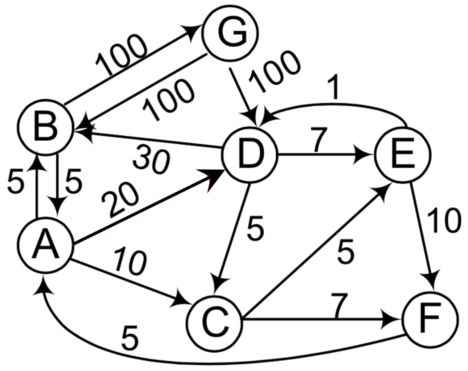
KickStarter

- Maintain value dependences during computation
  - $a \xrightarrow{\text{LT}} b$ iff b's value resulted from a



Trimming via Value Dependence

- Maintain value dependences during computation
  - $a \xrightarrow{\text{LT}} b$ iff *b*'s value resulted from *a*



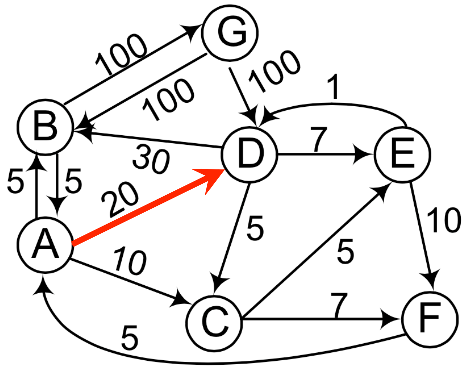
A	B	C	D	E	F	G
∞	20	10	20	7	7	20

$$\begin{tikzcd}
& A = \infty \arrow[dl] \arrow[dr] & & \\
C = 10 \arrow[d] & & D = 20 \arrow[d] \arrow[dr] & \\
F = 7 & & E = 7 & B = 20 \arrow[d] \\
& & & G = 20
\end{tikzcd}$$

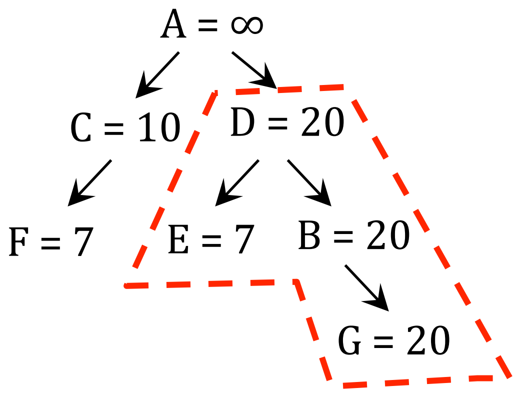
SSWP $v.path \leftarrow \max\limits_{e \in \textbf{inEdges}(v)} (\min(e.source.path, e.weight))$

Trimming via Value Dependence

- Compute safe approximations
  - Can be done using the same vertex function



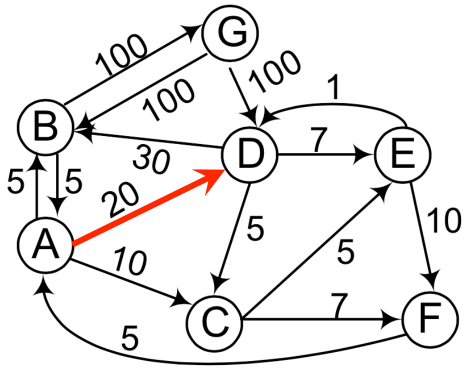
A	B	C	D	E	F	G
∞	20	10	20	7	7	20
Delete A→D						



SSWP $v.path \leftarrow \max\limits_{e \in \text{inEdges}(v)} (\min(e.source.path, e.weight))$

Trimming via Value Dependence

- Compute safe approximations
  - Can be done using the same vertex function



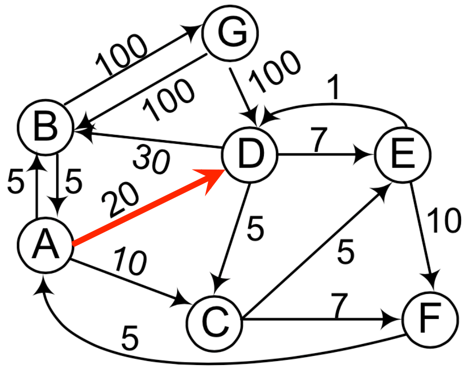
A	B	C	D	E	F	G
∞	20	10	20	7	7	20
Delete A → D						

$$\begin{tikzcd}
& & A = \infty \arrow[dl] \arrow[dr] & & & \\
& C = 10 \arrow[dl] & & \boxed{D = 20} \arrow[dl] \arrow[dr] & & \\
F = 7 & & E = 7 & & B = 20 \arrow[dr] & \\
& & & & & G = 20
\end{tikzcd}$$

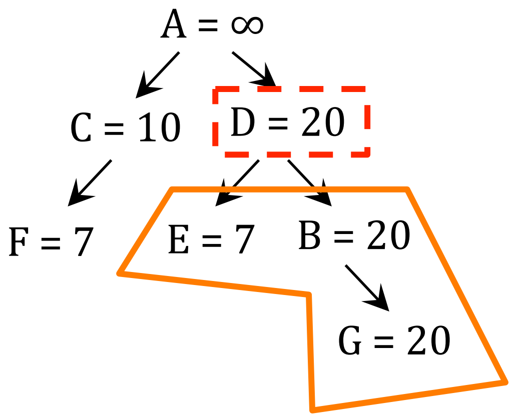
SSWP $$v.path \leftarrow \max_{e \in \text{inEdges}(v)} (\min(e.source.path, e.weight))$$

Trimming via Value Dependence

- Compute safe approximations
  - Can be done using the same vertex function



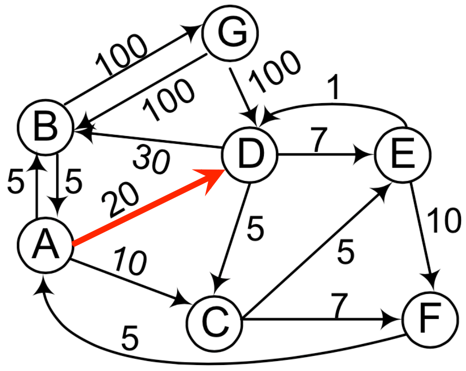
A	B	C	D	E	F	G
∞	20	10	20	7	7	20
Delete A → D						



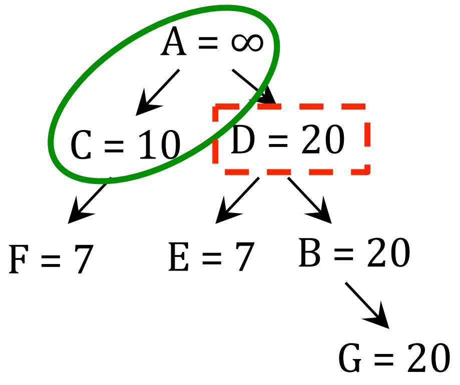
SSWP $v.path \leftarrow \max_{e \in \text{inEdges}(v)} (\min(e.source.path, e.weight))$

Trimming via Value Dependence

- Compute safe approximations
  - Can be done using the same vertex function



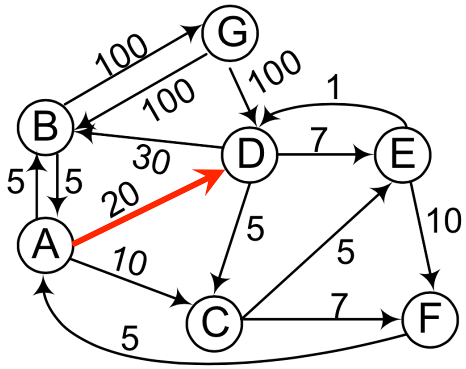
A	B	C	D	E	F	G
∞	20	10	20	7	7	20
Delete A → D						



SSWP $$v.path \leftarrow \max_{e \in \text{inEdges}(v)} (\min(e.source.path, e.weight))$$

Trimming via Value Dependence

- Compute safe approximations
  - Can be done using the same vertex function



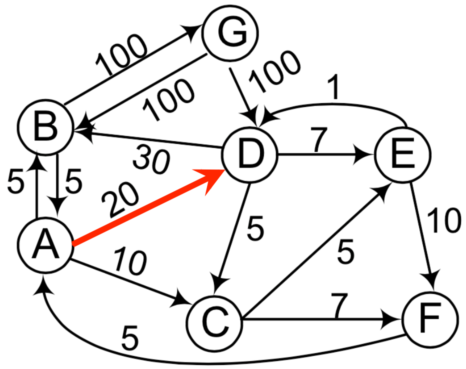
A	**B**	**C**	**D**	**E**	**F**	**G**
∞	20	10	20	7	7	20
Delete A → D						
∞	20	10	0	7	7	20

$$\begin{tikzcd}
 & & A = \infty \arrow[dl] & & D = 0 \arrow[dl] \arrow[dr] & & \\
 & C = 10 \arrow[dl] & & \boxed{E = 7} & & \boxed{B = 20} \arrow[dr] & \\
F = 7 & & & & & & G = 20
\end{tikzcd}$$

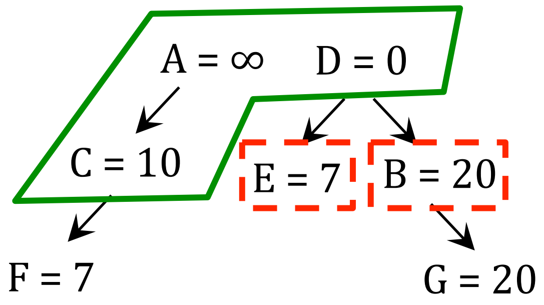
$$\textbf{SSWP} \quad v.path \leftarrow \max_{e \in \text{inEdges}(v)} (\min(e.source.path, e.weight))$$

Trimming via Value Dependence

- Compute safe approximations
  - Can be done using the same vertex function



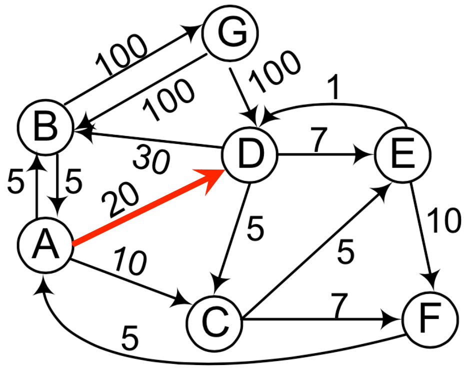
A	B	C	D	E	F	G
∞	20	10	20	7	7	20
Delete A → D						
∞	20	10	0	7	7	20



SSWP $$v.path \leftarrow \max_{e \in \text{inEdges}(v)} (\min(e.source.path, e.weight))$$

Trimming via Value Dependence

- Compute safe approximations
  - Can be done using the same vertex function



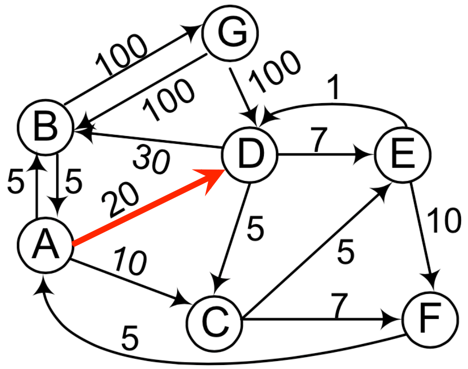
A	B	C	D	E	F	G
∞	20	10	20	7	7	20
Delete A → D						
∞	20	10	0	7	7	20
∞	5	10	0	5	7	20

$$\begin{tikzcd}
 & & A = \infty \arrow[dl] \arrow[dr] & & D = 0 \\
 & C = 10 \arrow[dl] \arrow[dr] & & B = 5 \arrow[dr] & \\
F = 7 & & E = 5 & & \boxed{G = 20}
\end{tikzcd}$$

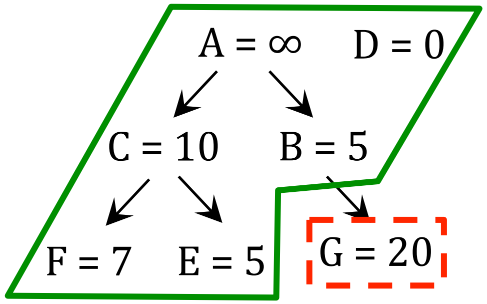
SSWP $v.path \leftarrow \max_{e \in \text{inEdges}(v)} (\min(e.source.path, e.weight))$

Trimming via Value Dependence

- Compute safe approximations
  - Can be done using the same vertex function



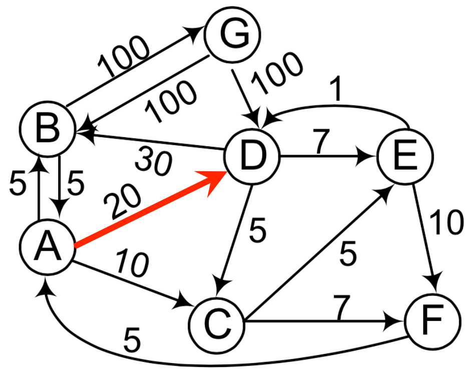
A	B	C	D	E	F	G
∞	20	10	20	7	7	20
Delete A → D						
∞	20	10	0	7	7	20
∞	5	10	0	5	7	20



SSWP $$v.path \leftarrow \max_{e \in \text{inEdges}(v)} (\min(e.source.path, e.weight))$$

Trimming via Value Dependence

- Compute safe approximations
  - Can be done using the same vertex function

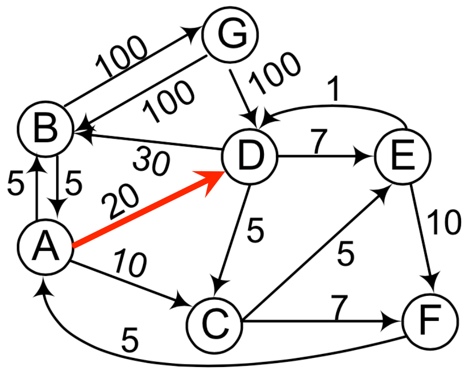


A	B	C	D	E	F	G
∞	20	10	20	7	7	20
Delete A → D						
∞	20	10	0	7	7	20
∞	5	10	0	5	7	20
∞	5	10	0	5	7	**5**
Trimming Complete						

$$\begin{tikzcd}
 & & A = \infty \arrow[dl] \arrow[dr] & & D = 0 \\
 & C = 10 \arrow[dl] \arrow[d] & & B = 5 \arrow[d] & \\
F = 7 & E = 5 & & G = 5 &
\end{tikzcd}$$

Trimming via Value Dependence

- Compute safe approximations
  - Can be done using the same vertex function

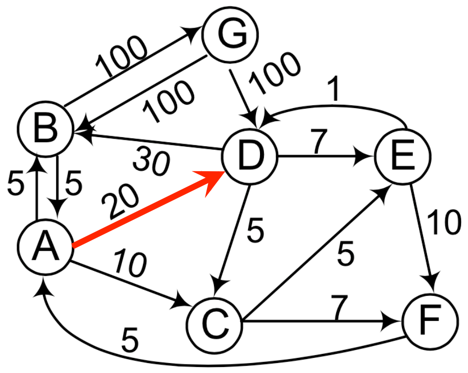


A	B	C	D	E	F	G
∞	20	10	20	7	7	20
Delete A → D						
∞	20	10	0	7	7	20
∞	5	10	0	5	7	20
∞	**5**	10	**0**	**5**	7	**5**
Trimming Complete						

$$\begin{tikzcd}
 & A = \infty \arrow[dl] \arrow[dr] & & D = 0 \\
C = 10 \arrow[d] \arrow[dr] & & B = 5 \arrow[dr] & \\
F = 7 & E = 5 & & G = 5
\end{tikzcd}$$

Trimming via Value Dependence

- Compute safe approximations
  - Can be done using the same vertex function

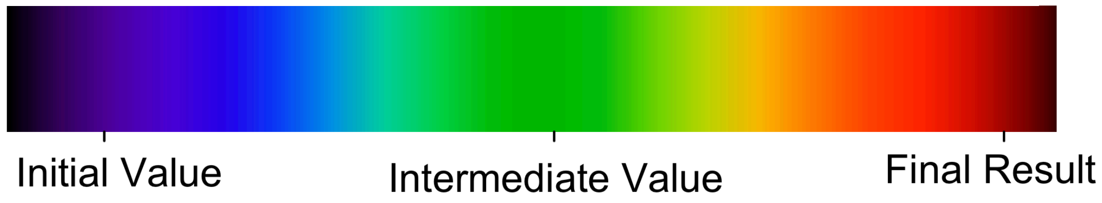


A	B	C	D	E	F	G
∞	20	10	20	7	7	20
Delete A → D						
∞	20	10	0	7	7	20
∞	5	10	0	5	7	20
∞	5	10	0	5	7	5
Trimming Complete						
∞	5	10	5	5	7	5

$$\begin{tikzcd} & & A = \infty \arrow[dl] \arrow[dr] & & D = 5 \\ & C = 10 \arrow[dl] \arrow[dr] & & B = 5 \arrow[dr] & \\ F = 7 & & E = 5 & & G = 5 \end{tikzcd}$$

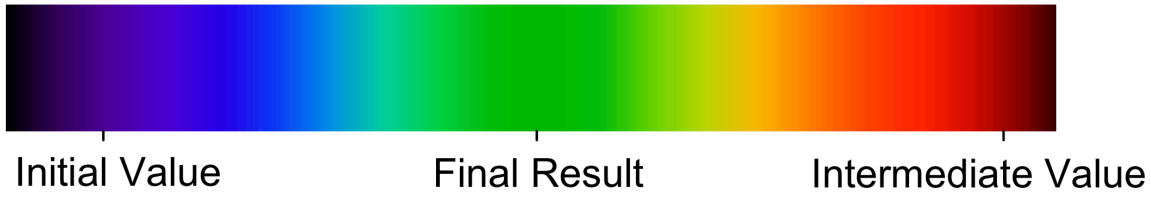
Safety

SSWP $v.path \leftarrow \max_{e \in \mathbf{inEdges}(v)} (\min(e.source.path, e.weight))$



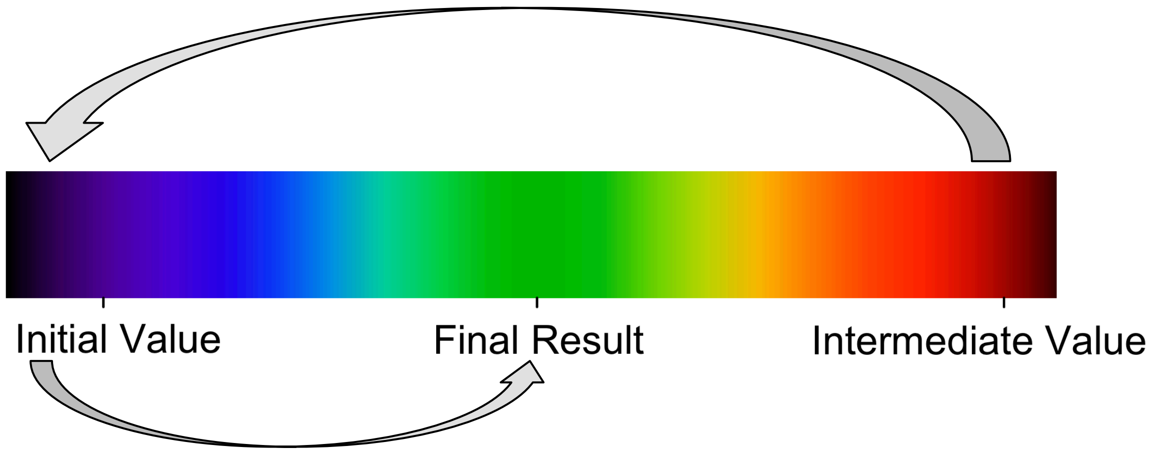
Safety

SSWP $v.path \leftarrow \max_{e \in \textbf{inEdges}(v)} (\min(e.source.path, e.weight))$



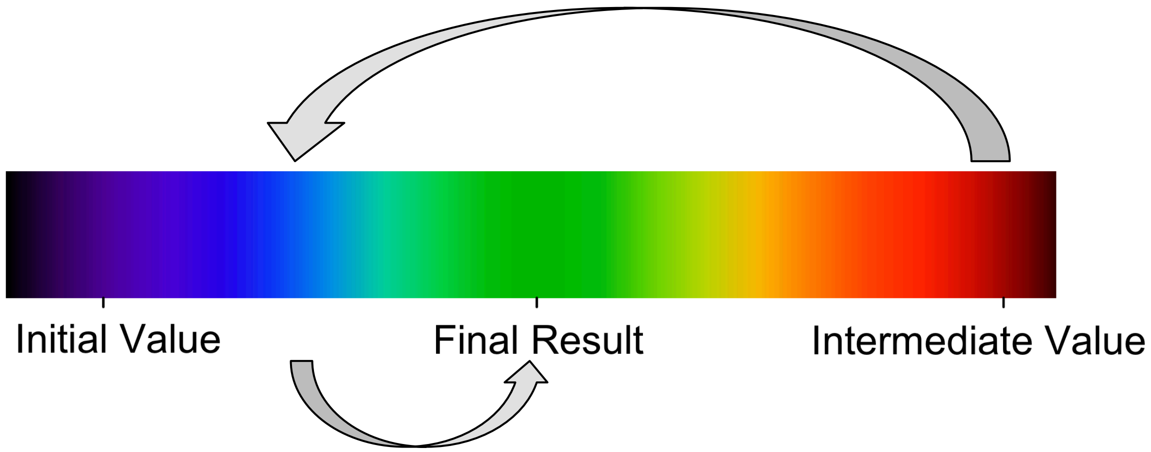
Safety

SSWP $v.path \leftarrow \max_{e \in \textbf{inEdges}(v)} (\min(e.source.path, e.weight))$



Safety

SSWP $$v.path \leftarrow \max_{e \in \textbf{inEdges}(v)} (\min(e.source.path, e.weight))$$



Monotonic Graph Algorithms

- Vertex values exhibit increasing/decreasing values

- WidestPaths (SSWP)

Monotonic Graph Algorithms

- Vertex values exhibit increasing/decreasing values

- ShortestPaths (SSSP)

- WidestPaths (SSWP)

Monotonic Graph Algorithms

- Vertex values exhibit increasing/decreasing values

▼

- ShortestPaths (SSSP)
- ConnectedComponents
- MinimumSpanningTree
- BreadFirstSearch
- FacilityLocation

▲

- WidestPaths (SSWP)
- Reachability

Experimental Setup

- 16-node EC2 cluster: 8-core/16GB nodes
- Monotonic algorithms

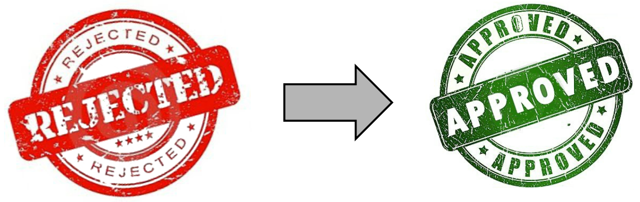
Bad Scenario	SingleSourceWidestPaths (SSWP) ConnectedComponents (CC)
Ugly Scenario	SingleSourceShortestPaths (SSSP) BreadthFirstSearch (BFS)

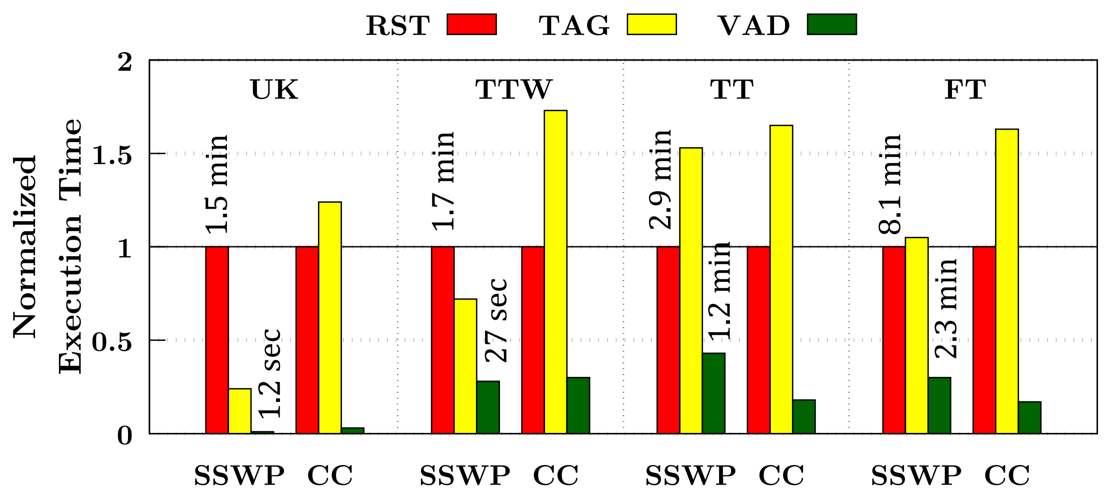
Experimental Setup

- Streaming graph datasets constructed using [SIGMOD'16]
  - Fixed point achieved at 50% edges
  - Remaining edges treated as edge additions
  - Edge deletions sampled from loaded graph
- Rate of update stream (100K-1M updates per query)
- Edge deletion ratio (10-50%)

Graphs	**#Edges**	**#Vertices**
Friendster (FT)	2.5B	68.3M
Twitter (TT)	2.0B	52.6M
Twitter (TTW)	1.5B	41.7M
UKDomain (UK)	1.0B	39.5M

Trimming for Safety



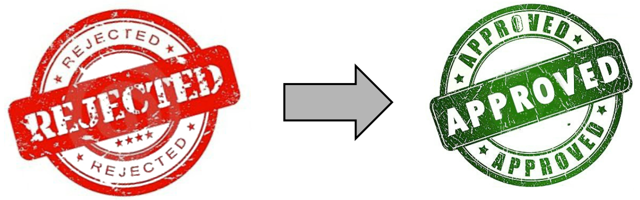


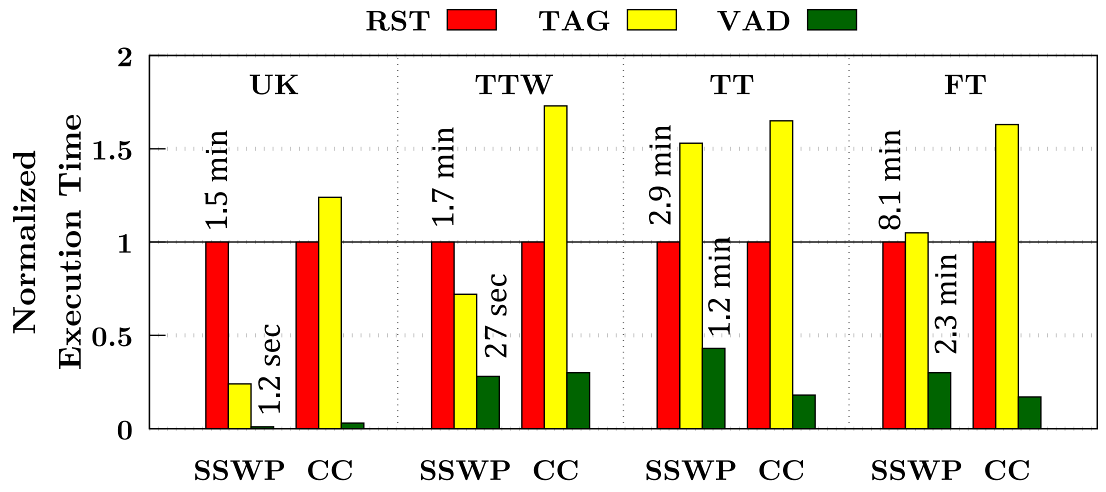
	SSWP	CC
RST/ VAD	17.7x	10x
TAG/ VAD	6.2x	13.7x

RST: Reset all vertex values

- 100K updates per query
- 30% deletion rate

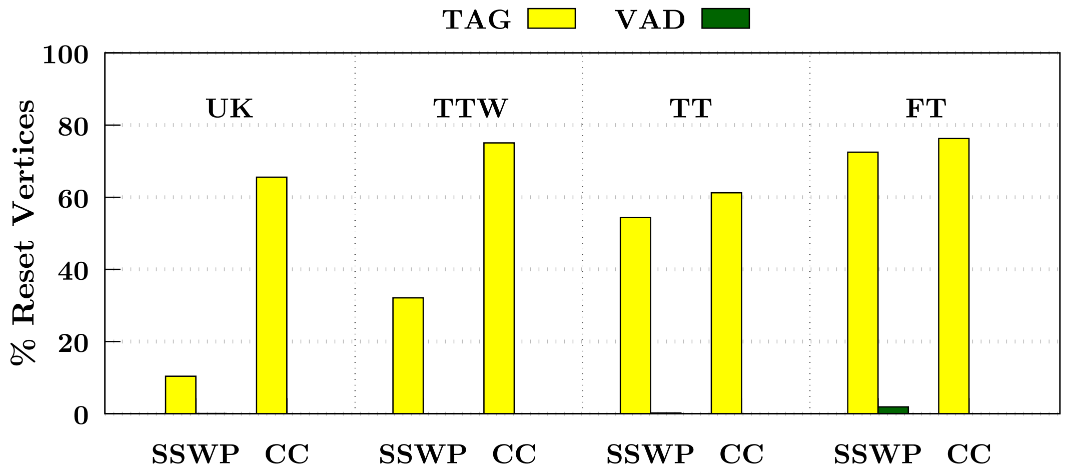
Trimming for Safety





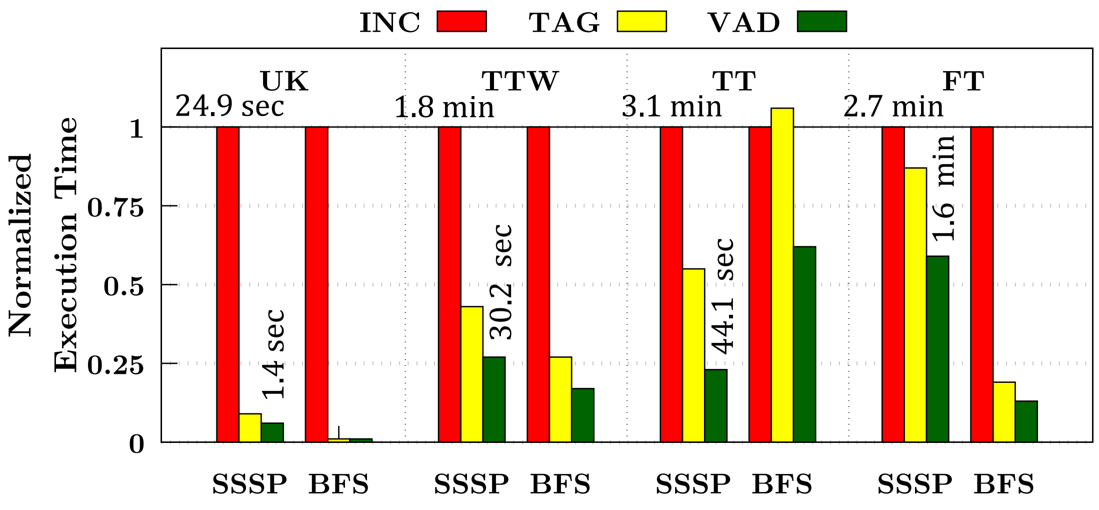
	SSWP	CC
RST/ VAD	17.7x	10x
TAG/ VAD	6.2x	13.7x

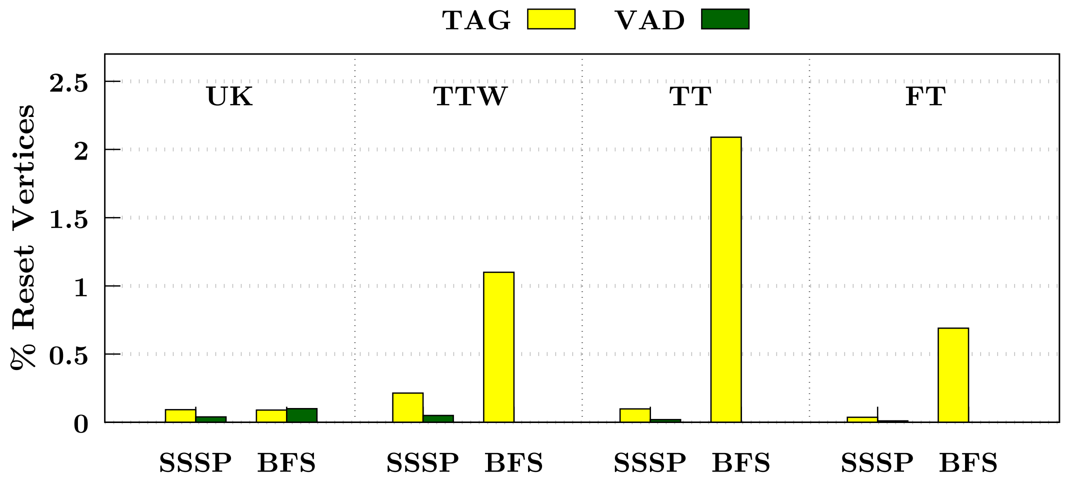
RST: Reset all vertex values



- 100K updates per query
- 30% deletion rate

Trimming for Performance





	SSSP	BFS
INC/ VAD	23.7x	8.5x
TAG/ VAD	1.5x	1.7x

INC: Incremental processing (no resets)

- ➢ 100K updates per query
- ➢ 30% deletion rate

More Results

- Individual query performance
  - Trimming v/s computation time
- Effectiveness of trimming over resetting
- Varying update rate
- Varying edge deletion ratio
- Dependence tracking overhead

Summary

- Incremental processing in presence of edge deletions
- Trimming approximation
  - Reuse safe and profitable values
- Active dependence tracking based trimming
  - Up to 8.5-23.7x speedups

Thanks