UNIVERSITY OF CALIFORNIA
UC RIVERSIDE

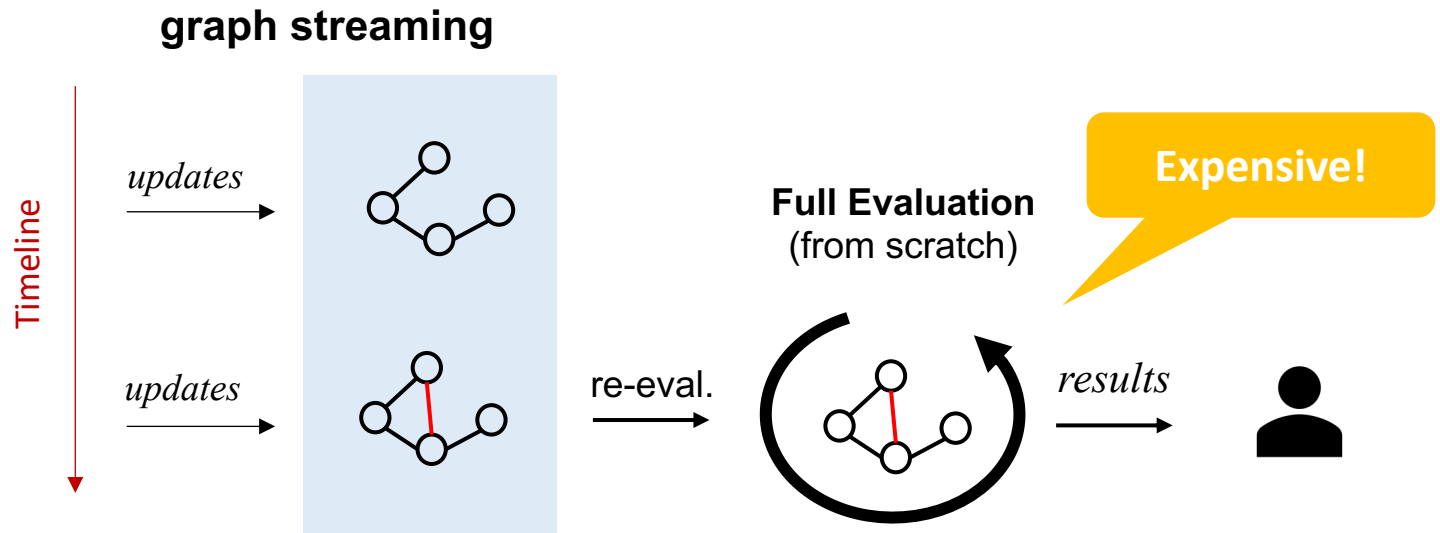
Tripoline: Generalized Incremental Graph Processing via Graph Triangle Inequality

Xiaolin Jiang*, Chengshuo Xu*, Xizhe Yin*, Zhijia Zhao, Rajiv Gupta
Computer Science and Engineering
University of California Riverside

∗First three authors contributed equally to this research.

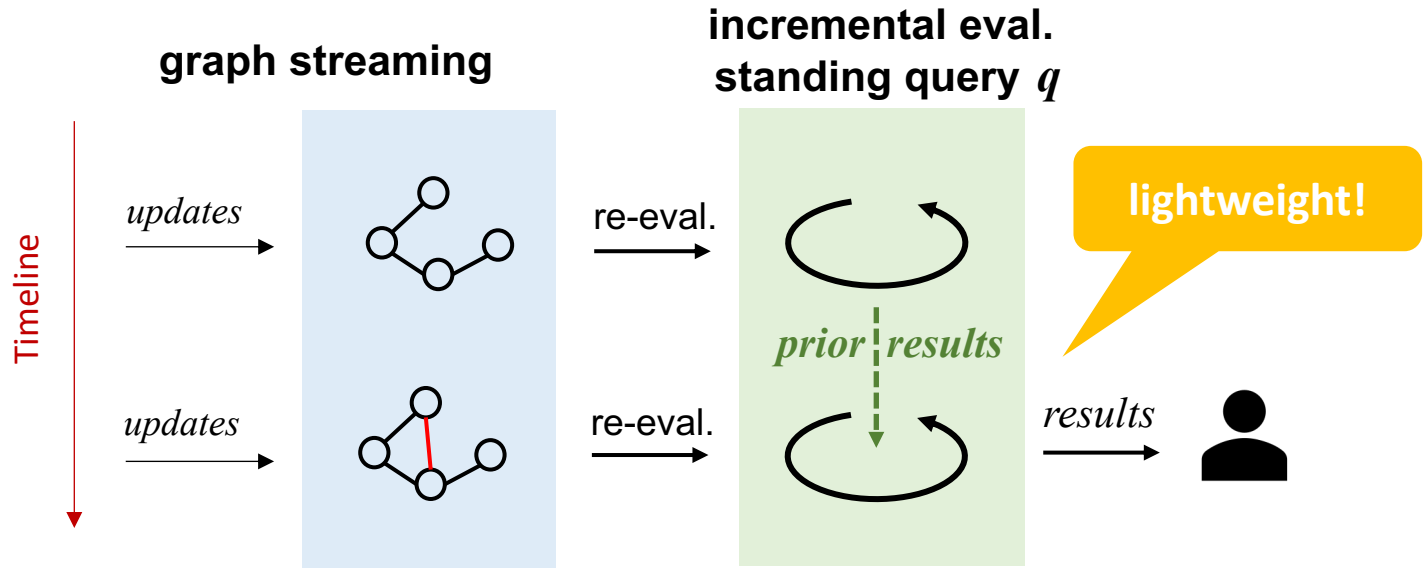
Background

- **Streaming graph processing**
  - A stream of updates (e.g., new edges) are continuously applied to the graph
  - E.g., social network, online shopping



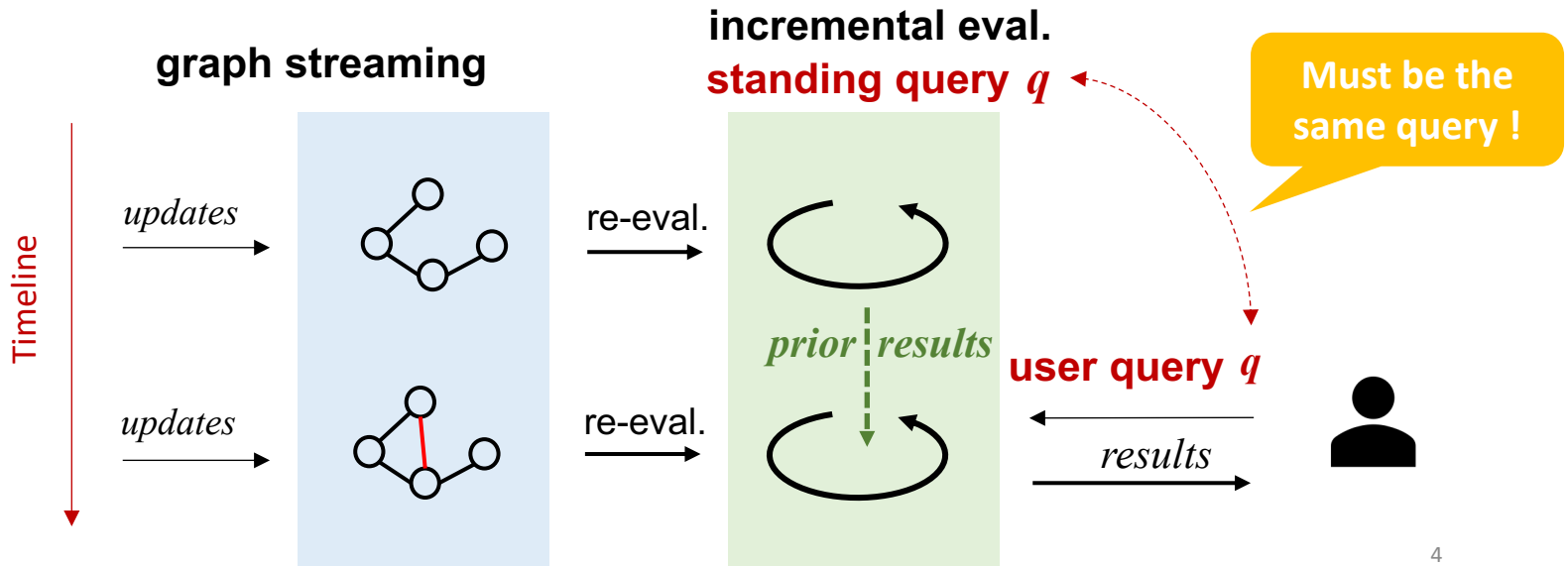
Existing Solutions to Streaming Graph Processing

- **Incremental evaluation**
  - Kineograph [EuroSys'12], Naiad [SOSP'13], Tornado [SIGMOD'16], Kickstarter [ASPLOS'17]
  - Start the evaluation from results of a prior evaluation, rather than from scratch



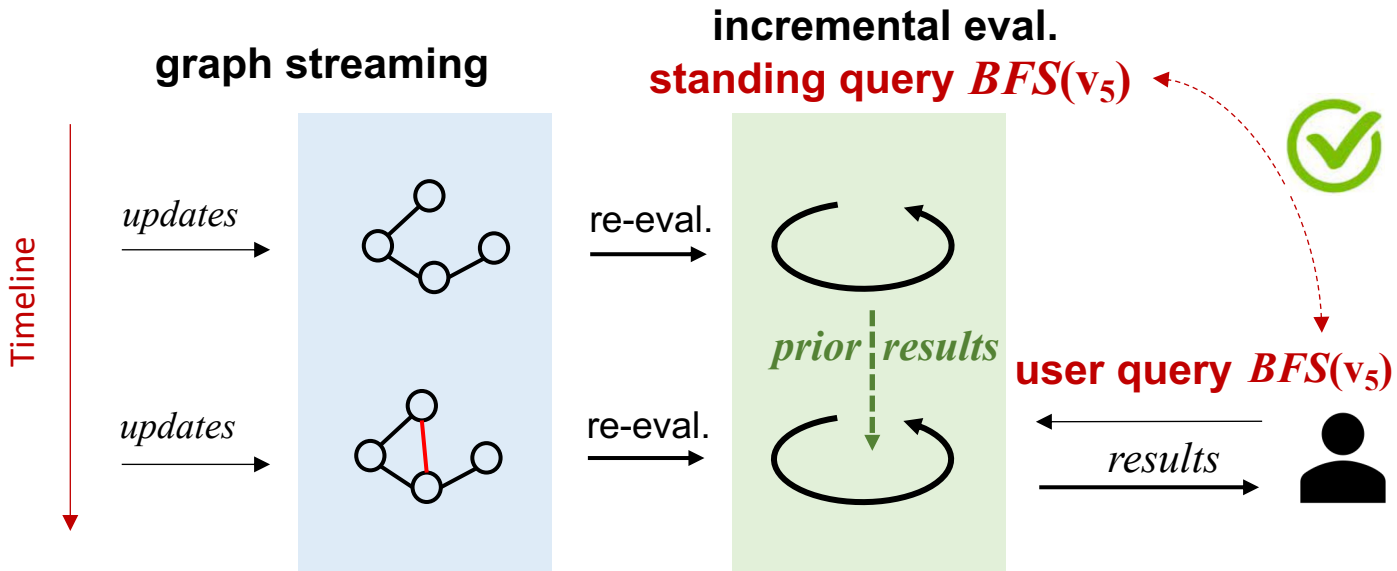
Limitation in Existing Incremental Graph Processing

- **Require a priori knowledge of (standing) query**



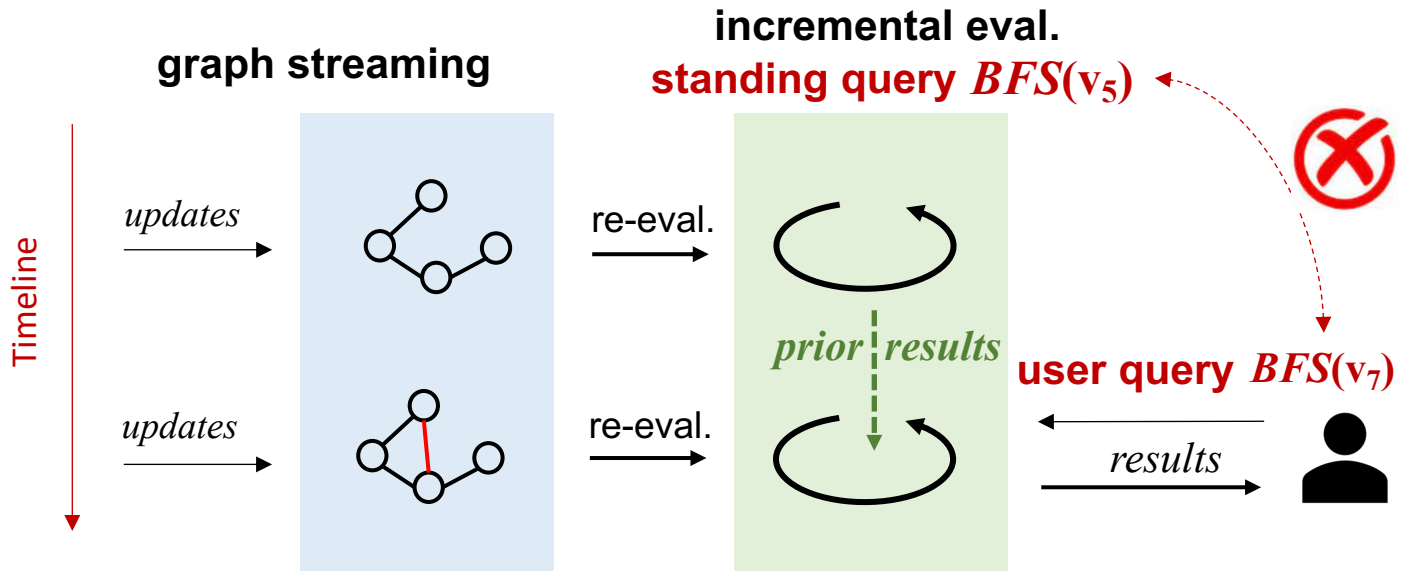
Limitation in Existing Incremental Graph Processing

- **Require a priori knowledge of (standing) query**
  - A serious issue for vertex-specific queries (e.g., BFS, SSSP, SSWP, etc.)



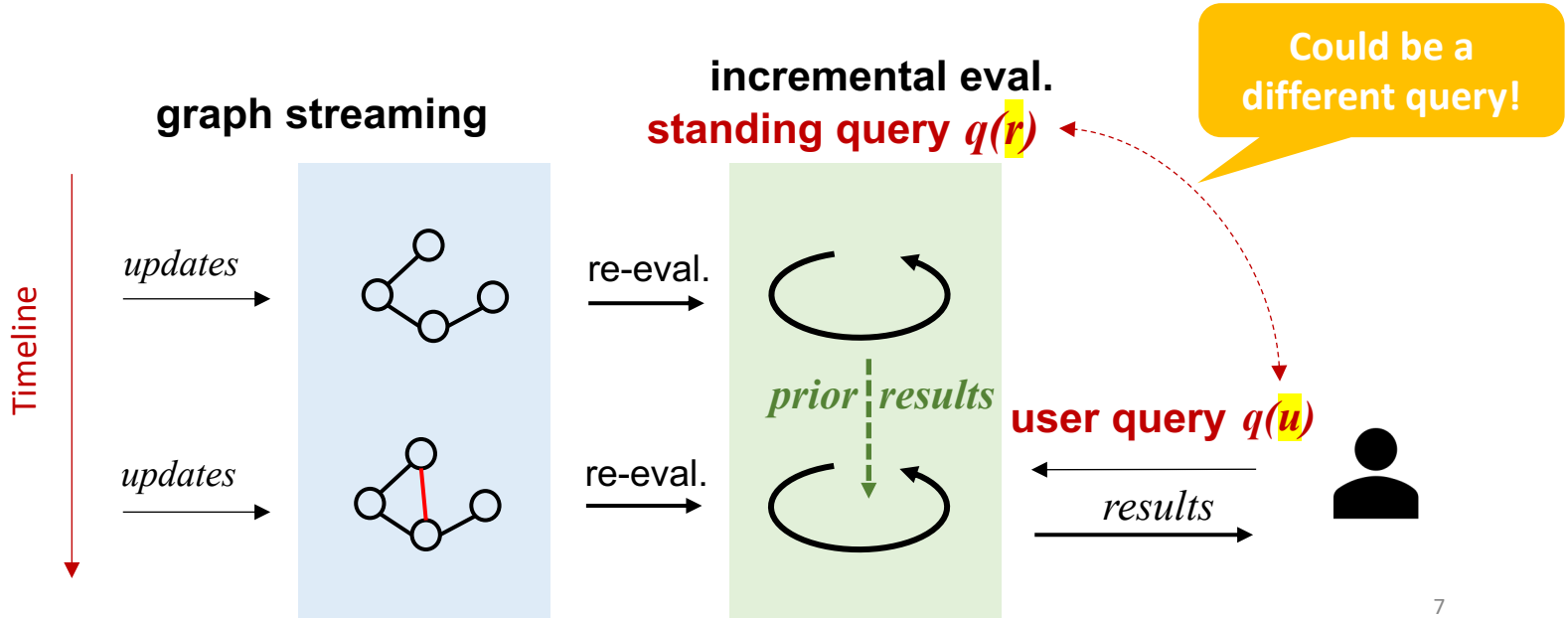
Limitation in Existing Incremental Graph Processing

- **Require a priori knowledge of (standing) query**
  - A serious issue for vertex-specific queries (e.g., BFS, SSSP, SSWP, etc.)



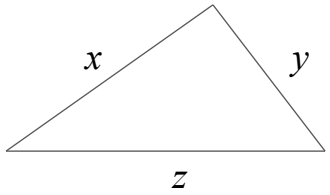
Goal of This Work

- ~~**Require a prior knowledge of (standing) query**~~
- **Incremental evaluation of an arbitrary user query (of the same type)**



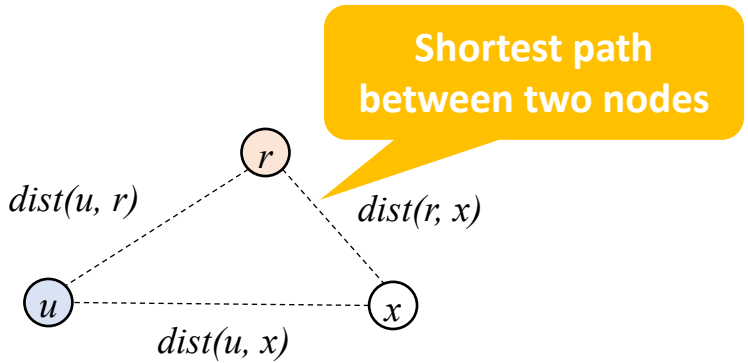
Key Idea: **Graph Triangle Inequality**

- **<u>Properties regarding three different vertices form a triangle</u>**
  - Inspired the classic triangle inequality



for any x, y, and z $\quad x + y \geq z$

Triangle Inequality in Euclidean Space



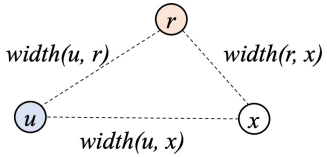
for any u, r, and x $\quad dist(u, r) + dist(r, x) \geq dist(u, x)$

Distance-based Triangle Inequality in Graphs*

*A sketch-based distance oracle for web-scale graphs [Sarma'10]

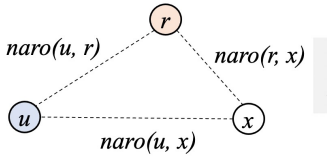
Key Idea: **Graph Triangle Inequality**

- **<u>Graph triangle inequalities exist for many vertex-specific properties</u>**



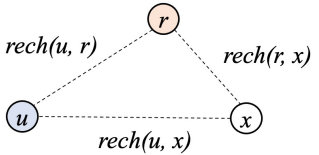
for any u, r, and x
$min(width(u, r), width(r, x)) \leq width(u, x)$

(a) SSWP Triangle



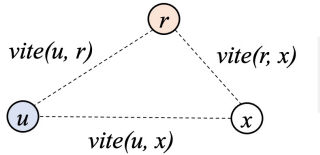
for any u, r, and x
$max(naro(u, r), naro(r, x)) \geq naro(u, x)$

(b) SSNP Triangle



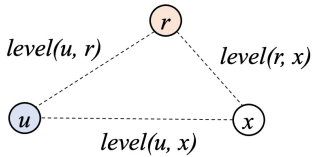
for any u, r, and x
$rech(u, r)$ *AND* $rech(r, x) \leq rech(u, x)$

(c) SSR Triangle



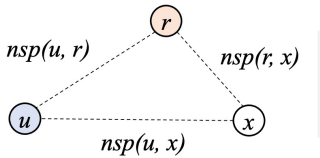
for any u, r, and x
$vite(u, r) * vite(r, x) \leq vite(u, x)$

(d) Viterbi Triangle



for any u, r, and x
$level(u, r) + level(r, x) \geq level(u, x)$

(e) BFS Triangle

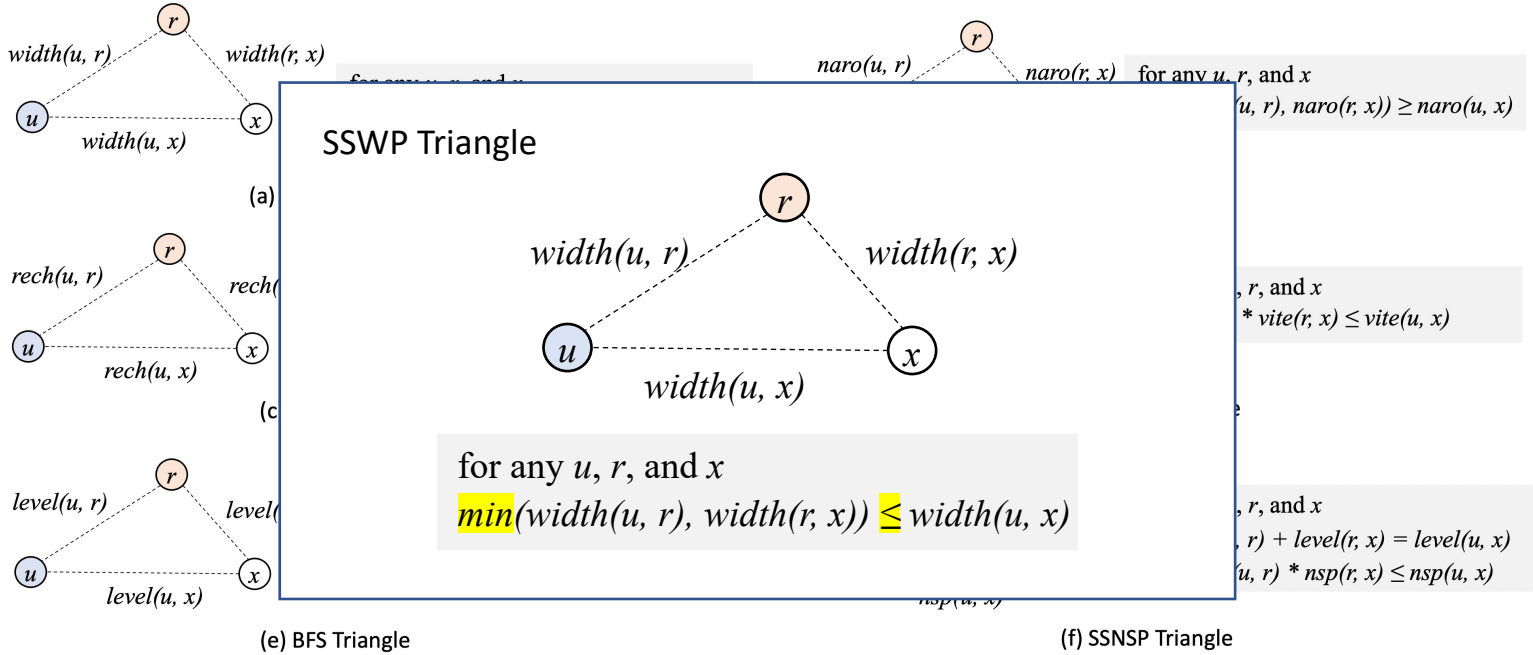


for any u, r, and x
if $level(u, r) + level(r, x) = level(u, x)$
then $nsp(u, r) * nsp(r, x) \leq nsp(u, x)$

(f) SSNSP Triangle

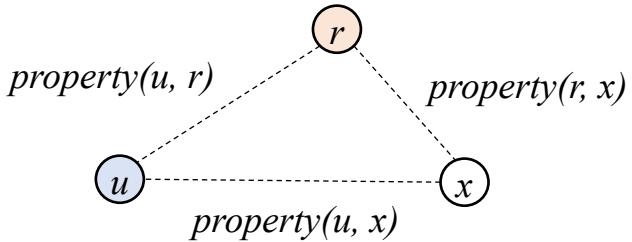
Key Idea: **Graph Triangle Inequality**

- **Graph triangle inequalities exist for many vertex-specific properties**



Key Idea: **Graph Triangle Inequality**

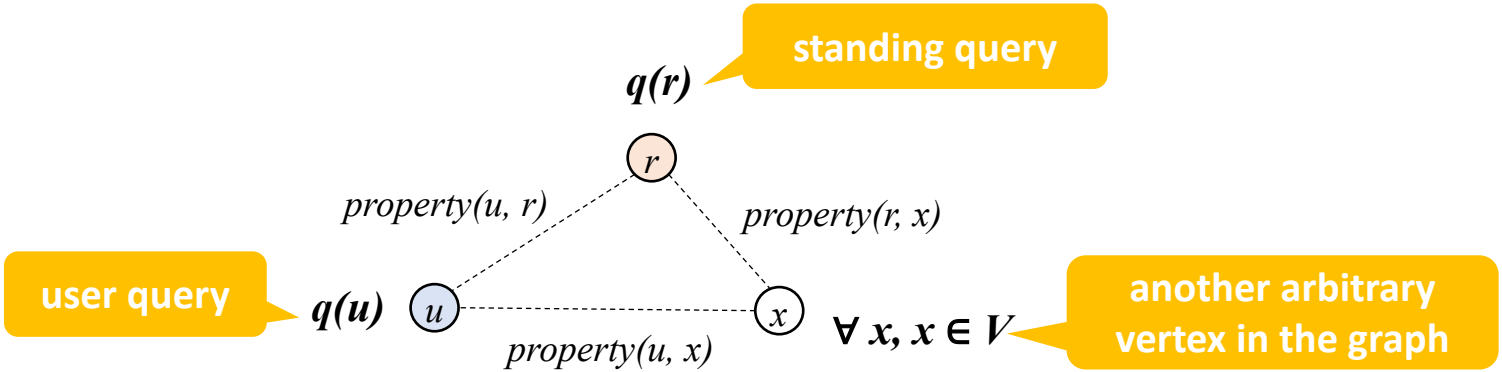
- **<u>Abstraction with a generalized “Addition” and “Greater Than”</u>**



for any u, r, and x $\quad property(u, r) \oplus property(r, x) \succcurlyeq property(u, x)$

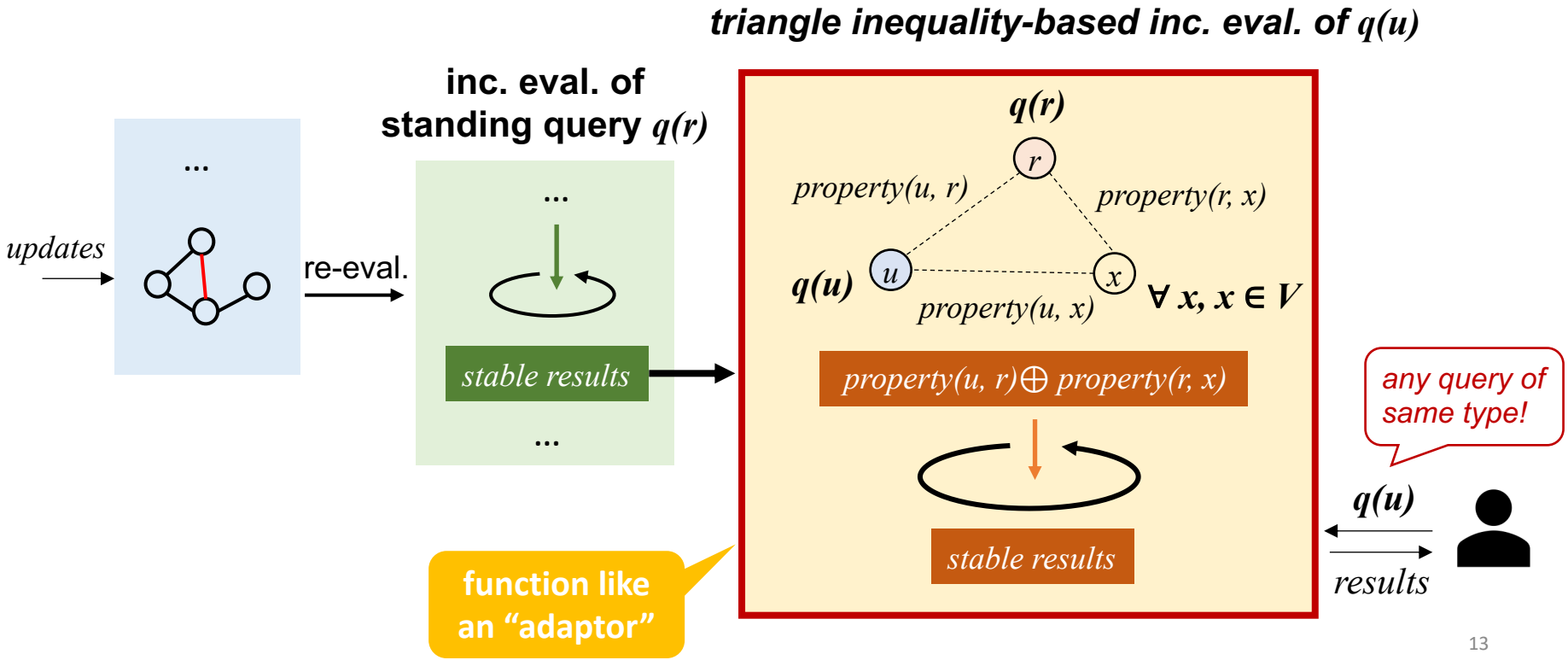
Key Idea: Connect Queries with Graph Triangle Inequality

- **Build constraints between Standing Query and User Query**



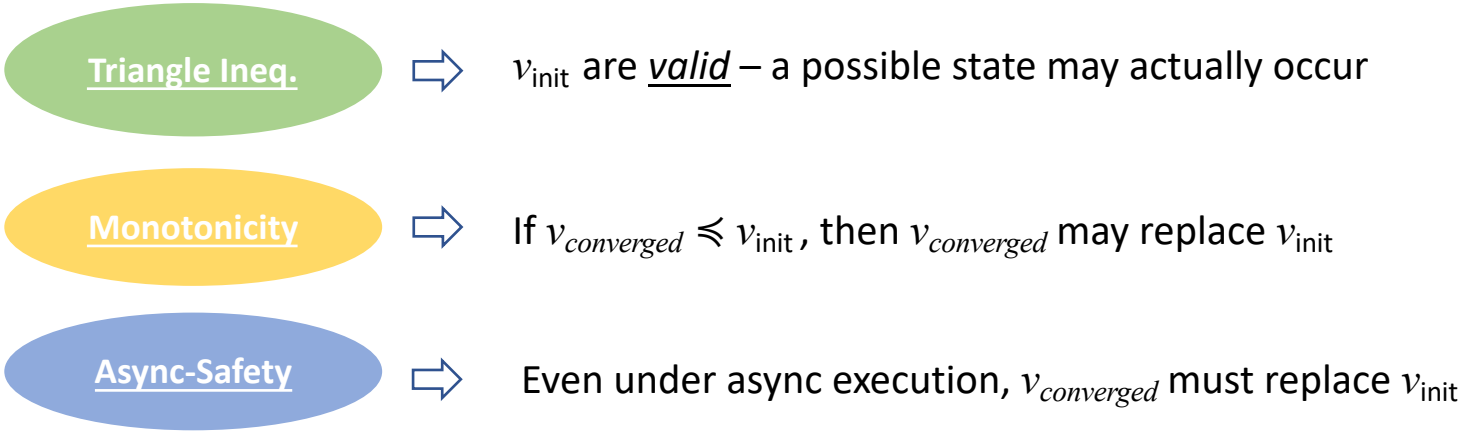
for any u, r, and x $\quad property(u, r) \oplus property(r, x) \succcurlyeq property(u, x)$

Graph Triangle Inequality-based Incremental Evaluation

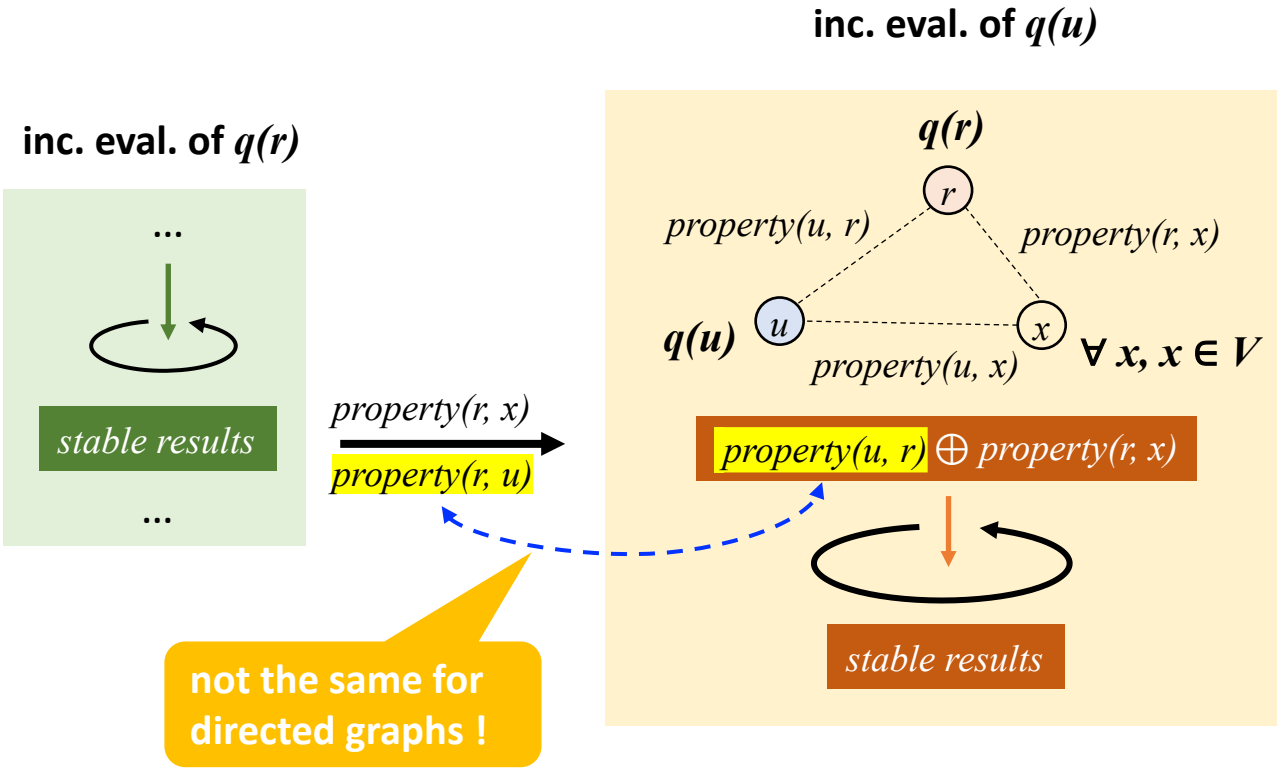


Assumptions and Correctness

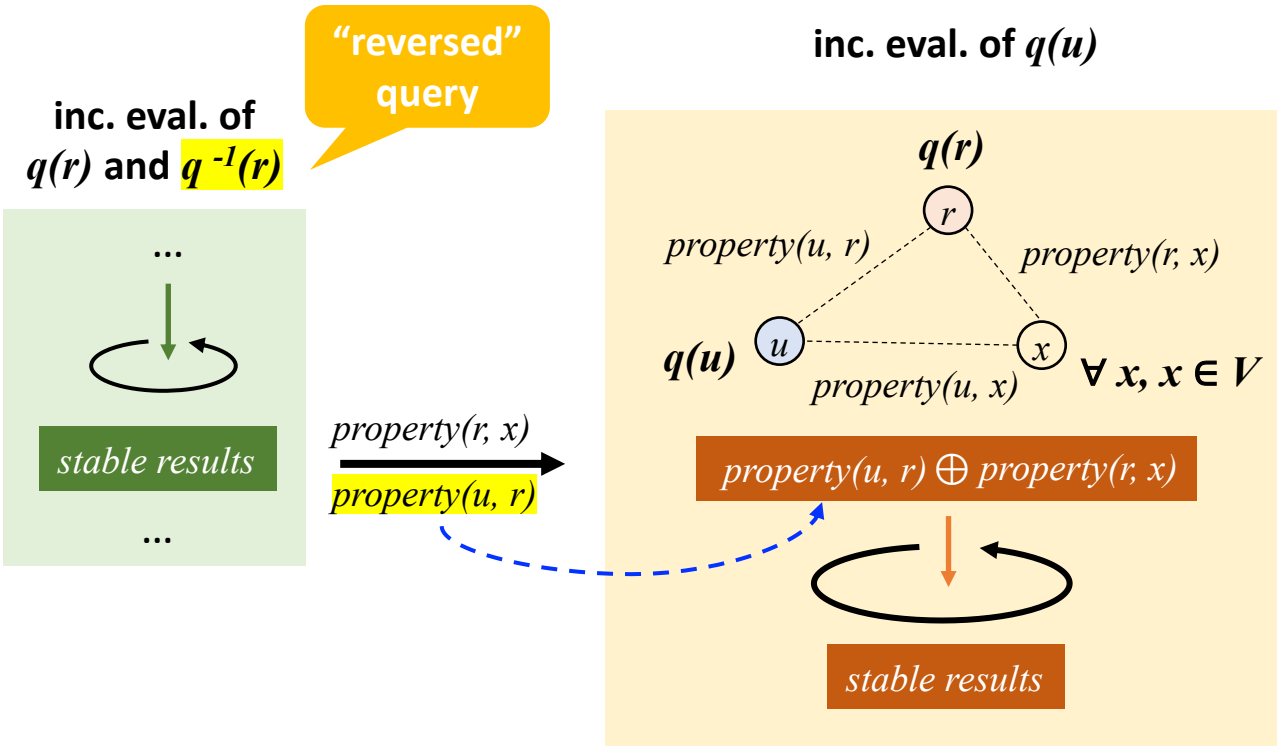
- Assumptions: **Monotonicity** and **Async-Safety**
  - from GraphLab [UAI'10], GRAPE [VLDB'17], Subway [Eurosys'20], Kickstarter [ASPLOS'17], etc.



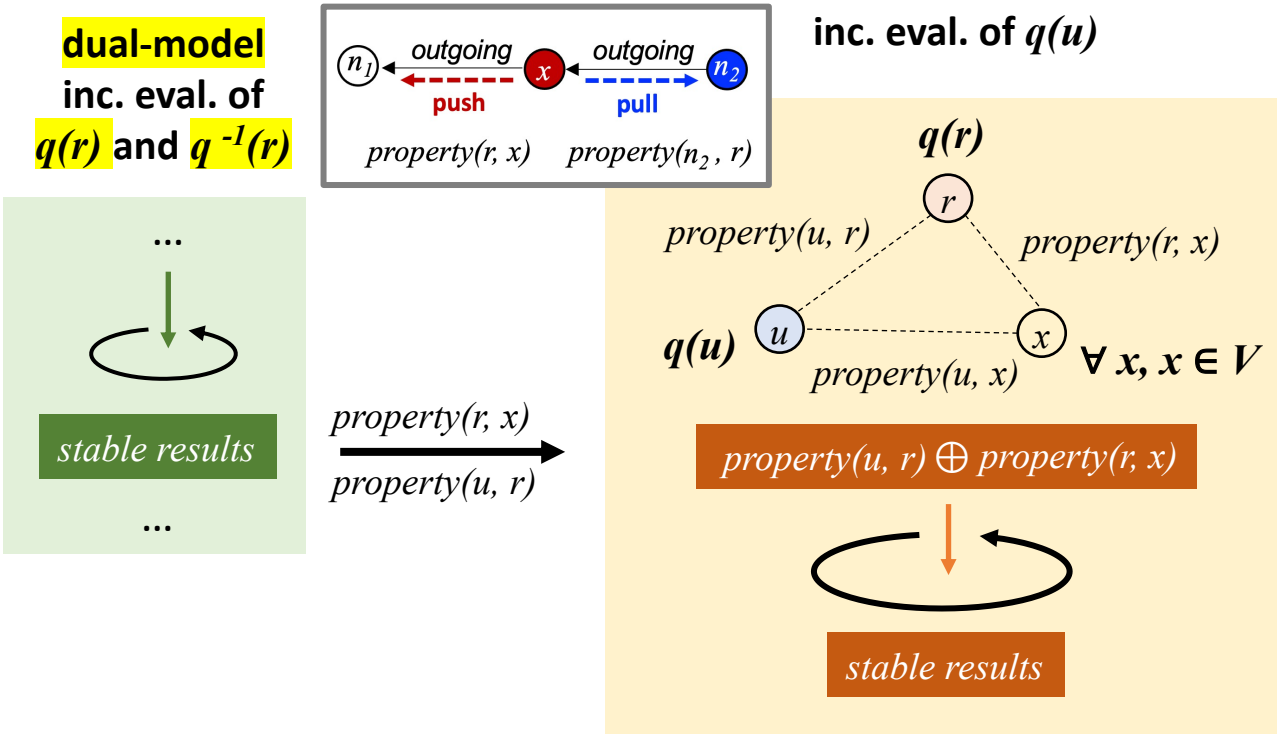
Complexity: Handling Directed Graphs



Complexity: Handling Directed Graphs



Complexity: Handling Directed Graphs



Complexity: Standing Query Selection

- **Triangle-based Selection**
  - select the query *q(r*)* that can minimize the **initial values** for all possible user queries

$$r^* = \arg\min_{r \in V} \sum_{\forall x \in V} property(u, r) \oplus property(r, x)$$

$property(u, r) \oplus property(r, x) \succcurlyeq property(u, x)$

- **Topology-based Selection**
  - select the query *q(r*)* that can maximize the **reachability** to all possible user queries

$$r^* = \arg\max_{r \in V} degree(r)$$

Complexity: Standing Query Selection

- **<u>Two-Step Standing Query Selection</u>**

  - Step-1: maintaining K high-degree standing queries continunously and incrementally

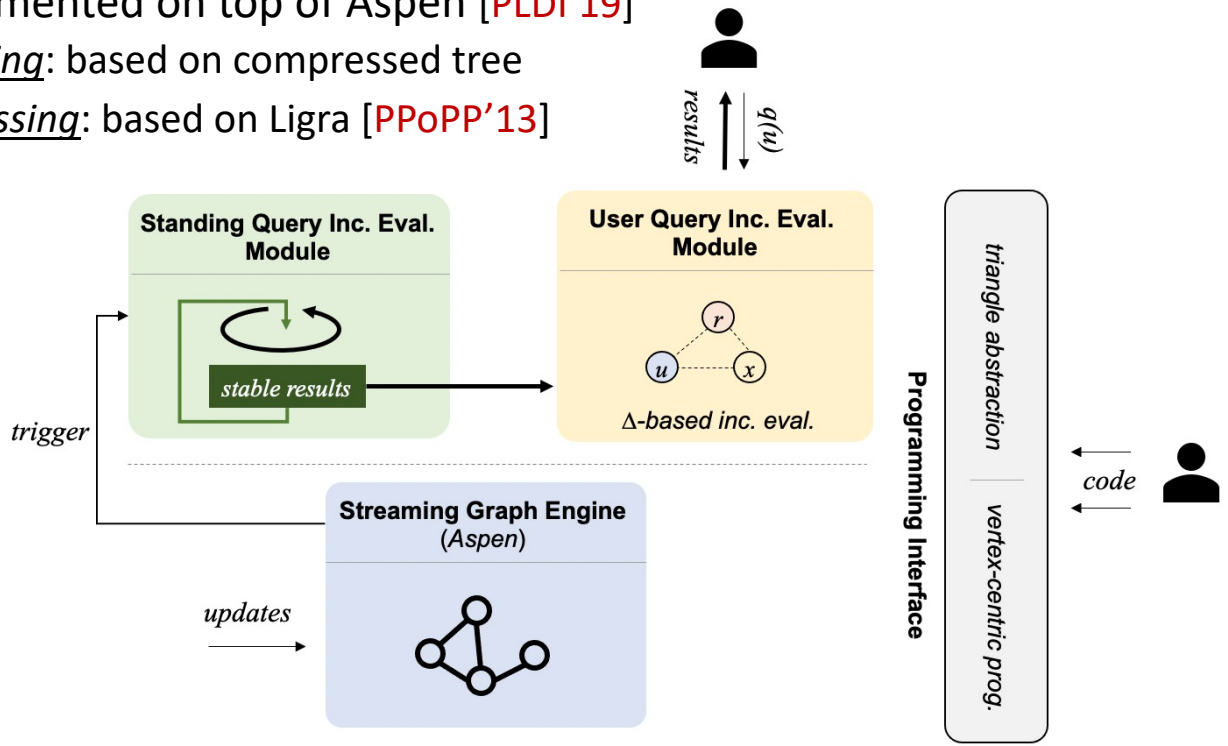
$$Standing_K = \{q(r_1), q(r_2), \cdots, q(r_K)\}$$

  - Step-2: given a user query *q(u)*, select one standing query to apply triangle inequality

$$r^* = \underset{r \in Standing_K}{\arg\min}\ property(u, r)$$

Implementation

- **Tripoline**: implemented on top of Aspen [PLDI'19]
  - *graph streaming*: based on compressed tree
  - *parallel processing*: based on Ligra [PPoPP'13]



Evaluation Setup

- Machine environment
  - Intel Xeon CPU E5-2683 v4 CPU (32 cores) and 512GB memory
  - CentOS 7.9 and g++ 8.3.
- Graph queries
  - BFS, SSSP, SSWP, Viterbi, SSNP, SSNSP, SSR, Radii
  - source vertex: 256 randomly selected vertices
- Real-world large graphs
  - LiveJournal (LJ), Twitter (TW), Orkut (OR), Friendster (FR)
- Streaming scenario
  - starting ratio of edges: 50%, 60%, 70%
  - graph update batch size: 10K
  - num. of standing queries: 1-64 (16 by default)

Experimental Results

- Speedups of **Tripoline** over the non-incremental evaluation

Graph	SSSP	SSWP	Viterbi	BFS	SSNP	SSR	Radii	SSNSP
OR-60	2.42x [0.17s]	33.91x [0.01s]	37.94x [0.01s]	1.25x [0.13s]	29.06x [0.01s]	10.86x [0.01s]	1.22x [2.43s]	1.09x [0.27s]
FR-60	1.34x [12.26s]	35.23x [0.38s]	41.48x [0.38s]	1.02x [7.16s]	18.77x [0.45s]	10.44x [0.45s]	1.18x [56.43s]	1.00x [9.58s]
LJ-60	1.81x [0.13s]	11.56x [0.01s]	26.88x [0.02s]	1.12x [0.07s]	11.53x [0.02s]	5.50x [0.02s]	1.16x [1.31s]	1.03x [0.20s]
TW-60	1.95x [1.45s]	15.97x [0.13s]	19.14x [0.13s]	1.56x [0.94s]	13.42x [0.14s]	8.32x [0.15s]	1.15x [11.85s]	1.18x [2.27s]
Avg.	**1.89x**	**23.28x**	**30.52x**	**1.24x**	**18.24x**	**8.83x**	**1.18x**	**1.08x**

Experimental Results

- Reduction in vertex function activations
  - higher activation ratio → lower speedup

$$R_{act} = \frac{Num_{act}\ triangle - based\ inc.eval.}{Num_{act}\ non - inc.eval.}$$

Table: Vertex Function Activation Ratio

	OR-60	FR-60	LJ-60	TW-60
SSSP	44.4%	61.7%	56%	52.8%
SSWP	1.9e-9	1.3e-8	0.79%	4.0e-8
Viterbi	3.5			-7
BFS	82.2%	98%	89.4%	65.8%
SSNP	1.9e-7	1.4e-8	0.78%	3.6e-8
SSR	3.3e-7	1.7e-8	0.78%	3.2e-8
Radii	98.9%	91.9%	92.21%	93.9%
SSNSP	98.9%	99.97%	98.58%	94.9%

min-max nature of the SSWP

Experimental Results

- Speedup distribution across queries

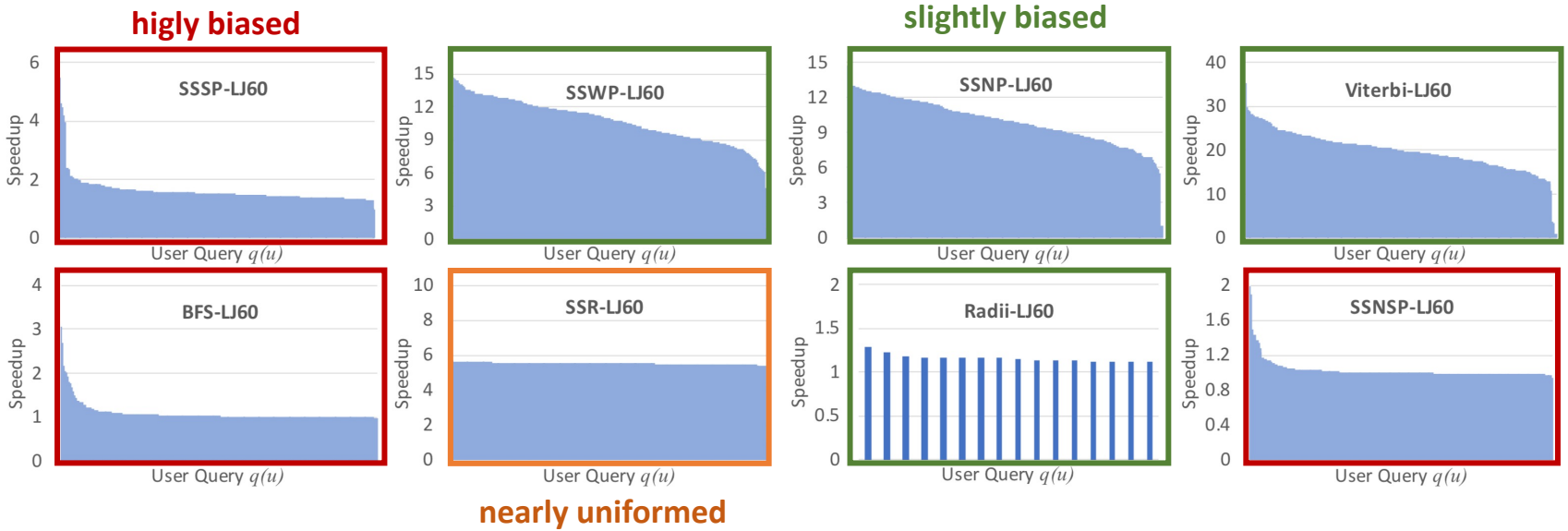


Figure: Speedups of 256 Queries (16 for Radii) **on LiveJournal**

Experimental Results

$$r^* = \arg\min_{r \in Standing_K} property(u, r)$$

- Confirmation of standing query selection heuristics

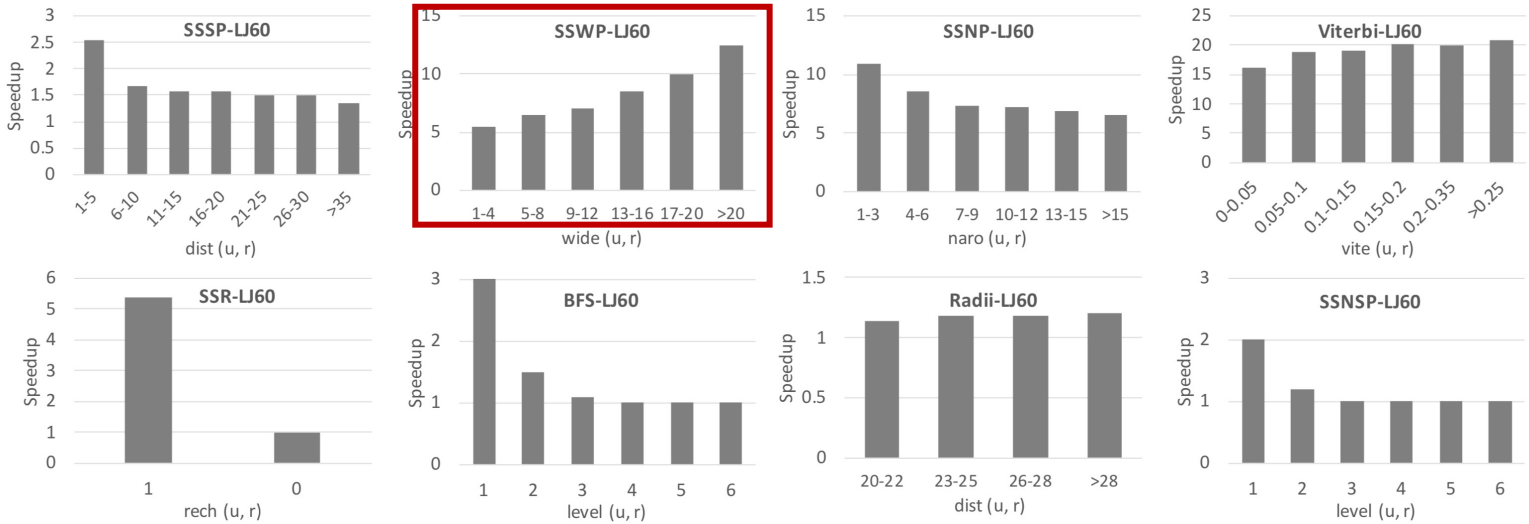


Figure: Correlations between Speedups and *property(u,r)*

Experimental Results

- Integeration into Differential Dataflow (with Shared Arrangements) [VLDB'20]

Table: DD with Triangle Inequality Optimization

Graph	Method	BFS	SSSP	SSWP
LJ-100	DD-SA	1.10s	8.41s	4.63s
	DD-SA-Tri	1.11s	3.24s	0.52s
	Speedup	[0.99x]	[2.60x]	[8.90x]
TW-100	DD-SA	10.69s	58.63s	32.68s
	DD-SA-Tri	10.71s	14.72s	7.74s
	Speedup	[1.00x]	[3.98x]	[4.22x]

Table: Reduction of `Reduce` Operations

Graph	Method	BFS	SSSP	SSWP
LJ-100	DD-SA	9156594	30418846	20622003
	DD-SA-Tri	8956638	17570555	6292821
	Reduction	[1.02x]	[1.73x]	[3.28x]

DD-SA: DD with shared arrangements enabled (baseline)
DD-SA-Tri: DD with triangle inequality opt.

UNIVERSITY OF CALIFORNIA UC RIVERSIDE

Conclusion

- **Graph triangle inequality** is common for vertex-specific graph queries
- It is key to enable **generalized incremental evaluation**
- Prototyped **Tripoline** system
- Observed significant speedups for multiple types of graph queries

Thank You!