

On Mining Distances out of Massive Time-Evolving Graphs

Applications, Algorithms, Experiments, Open Problems



Mattia D'Emidio¹



¹ Assistant Professor @ **UNIVERSITY OF L'AQUILA**
Lecturer/Scientific Collaborator @ **GRAN SASSO SCIENCE INSTITUTE**
email: mattia.demidio@univaq.gssi.it
web: www.mattiademidio.com

NetworkKit Day 2020
October 15, 2020



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA



- 1 Mining Distances: Problem
- 2 Mining Distances vs Modern Applications
- 3 Scalable Mining of Distances
- 4 2-HOP-COVER
- 5 2-HOP-COVER and Time-Evolving Graphs



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Outline



- 1 Mining Distances: Problem
- 2 Mining Distances vs Modern Applications
- 3 Scalable Mining of Distances
- 4 2-HOP-COVER
- 5 2-HOP-COVER and Time-Evolving Graphs



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Mining Distances: Problem



MINING DISTANCES FROM (DI)GRAPHS

- Given **(DI)GRAPH** $G = (V, A)$
- Given sequence of **(DISTANCE) QUERIES** $\{q(s_1, t_1), q(s_2, t_2), \dots\}$ for pairs of vertices $s_i, t_i \in V$
- Report **DISTANCE** $d(s_i, t_i)$ as fast as possible
 - $d(s_i, t_i)$: distance = **weight of a shortest path** from s_i to t_i in G

RELATED VARIANTS

- **REACHABILITY QUERIES**: report **yes** if there exist a path from s_i and t_i in G , **no** otherwise
- **PATH-REPORTING QUERIES**: report **whole shortest path** (sequence of vertices/arcs) if any, **empty set** otherwise



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

ONE OF MOST STUDIED PROBLEMS in Computer Science/Engineering

SEVERAL HIGHLY IMPACTING APPLICATIONS in real-world

- *Routing* in communication networks
- *Route/Journey planning* road/transport networks
- *Context Aware Search*, web indexing
- *Data mining* for linked data, *Graph Databases*
- *Social Networks* analysis, *Social Engineering*
- *Bioinformatics*, *Top-k Nearest Keyword Search*

HUGE AMOUNT OF RESEARCH/LITERATURE

- Algorithms, data structures, bounds and complexity results

TEXTBOOK/STANDARD SOLUTION

- **SOLVE SSSP** (upon query), e.g. **Dijkstra's** algorithm
- **COST PER QUERY:** (for an n -vertex, m -arc graph)
 - $\mathcal{O}(m + n \log n)$ time, $\mathcal{O}(n)$ space
- **BEST KNOWN METHOD** for generally positively weighted digraphs



Problem: **SCALABILITY ISSUES** against "modern inputs"

Outline



- 1 Mining Distances: Problem
- 2 Mining Distances vs Modern Applications
- 3 Scalable Mining of Distances
- 4 2-HOP-COVER
- 5 2-HOP-COVER and Time-Evolving Graphs



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Mining Distances vs Modern Applications



BIG GRAPHS, BIG PROBLEMS

- Polynomial/linear **not good enough**
- $\mathcal{O}(m + n \log n)$ query algorithm leads to (tens of) **SECONDS PER QUERY** when graphs are **MASSIVE IN SIZE** ($\geq 10^5$ vertices/arcs)
- **UNSUSTAINABLE TIME OVERHEAD**, especially for interactive applications executing **thousands/millions queries a day**

GRAPHS EMERGING FROM MODERN APPLICATIONS are indeed **MASSIVE**

- **BILLIONS** of **VERTICES/ARCS**
 - e.g. Twitter, Facebook, Google Maps, WWW, Internet
- Moreover they are "**COMPLEX**"
- **NO TOPOLOGICAL FEATURES TO EXPLOIT** for accelerating queries
 - e.g. *regularity, planarity, power-law degree distributions* (communication networks, Internet, Web)



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Mining Distances vs Modern Applications



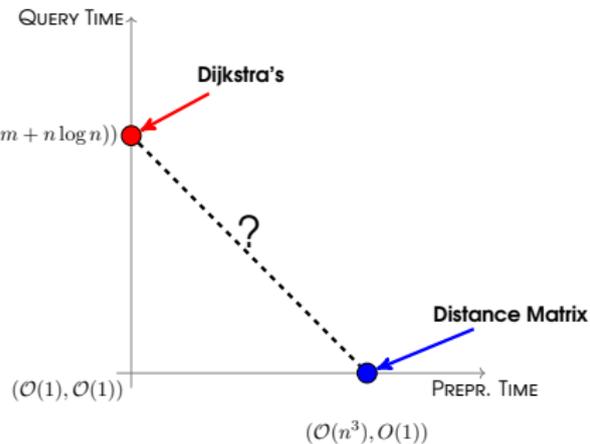
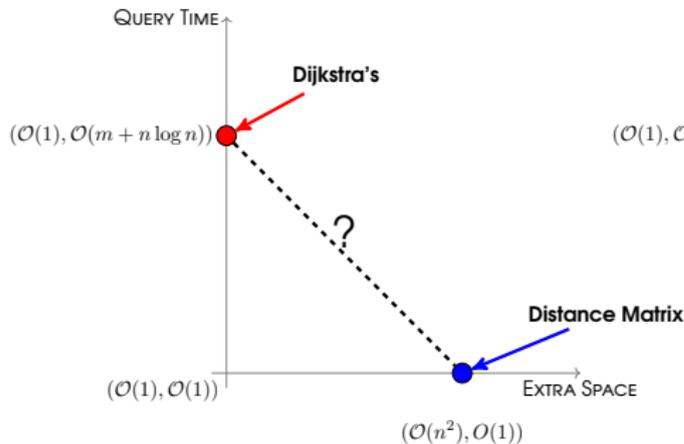
ALTERNATIVE TO STANDARD: use of **PREPROCESSING**

- Quite **well established strategy** to handle large inputs
 1. **SOLVE APSP** once, e.g. **Floyd-Warshall** algorithm or **repeated Dijkstra's**
 2. **STORE** outputs in a **DISTANCE MATRIX**
 3. **RETRIEVE** distances upon query by accessing right DM location
- **COST PER QUERY:** $\mathcal{O}(1)$ time/space to access (optimal)
- **PREPROCESSING COSTS**
 - $\mathcal{O}(nm + n^2 \log n) \in \mathcal{O}(n^3)$ **time**
 - $n \times n = \Theta(n^2)$ **extra space**

Problem: **SCALABILITY ISSUES** remain

- $\Theta(n^2)$ **space** is impractical (Thousands of GBs when $n \geq 10^6$)
- $\mathcal{O}(n^3)$ **time** is unacceptable (Tens of days when $n \geq 10^6$)

Query Time vs Extra Space and Prepr. Time



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Outline



- 1 Mining Distances: Problem
- 2 Mining Distances vs Modern Applications
- 3 Scalable Mining of Distances
- 4 2-HOP-COVER
- 5 2-HOP-COVER and Time-Evolving Graphs



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Scalable Mining of Distances



VERY ACTIVE RESEARCH FIELD: various techniques to find trade-offs

1. **APPROXIMATION**
2. **SAMPLING**
3. **PARALLELISM**
4. **RESTRICTION TO SPECIAL CLASSES OF GRAPHS**

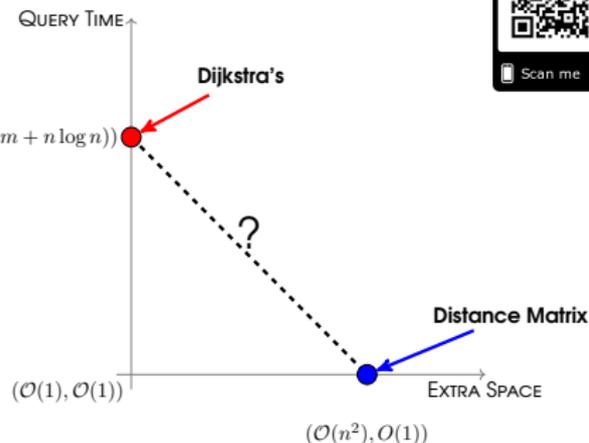
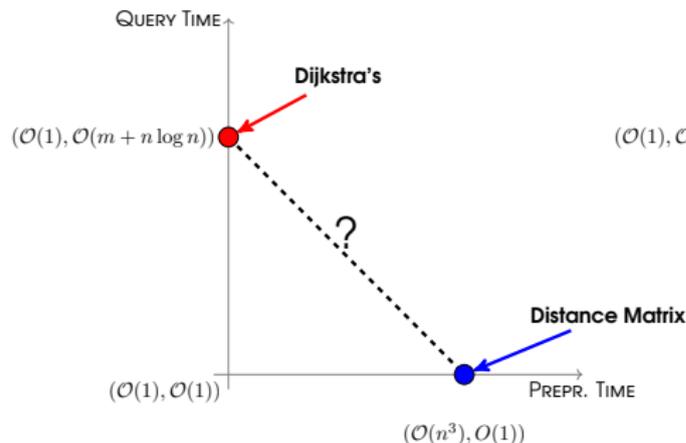
SOME LITERATURE (non-exhaustive list):

- [Potamias+ CIKM 2009][Elkin+ SODA 2015]
- [Thorup+ JACM 2015][Alstrup+, SODA 2016]
- **[Cohen+, SODA 2002, SIAM J. Comp. 2003]**
- [Thorup+ JACM 2005][Sarma+ WSDM 2010]
- [Abraham+ ESA 2012][Akiba+ SIGMOD 2013][Delling+ ESA 2014]



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

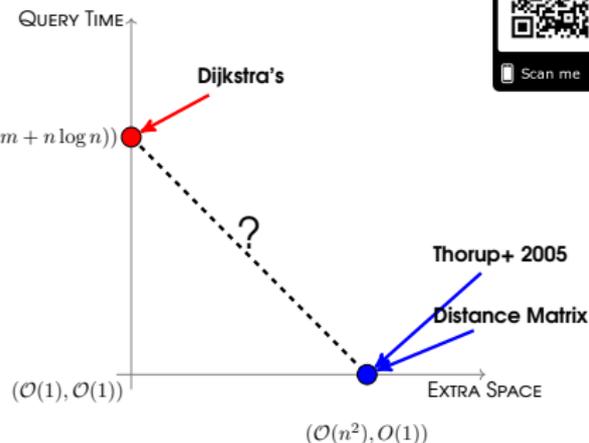
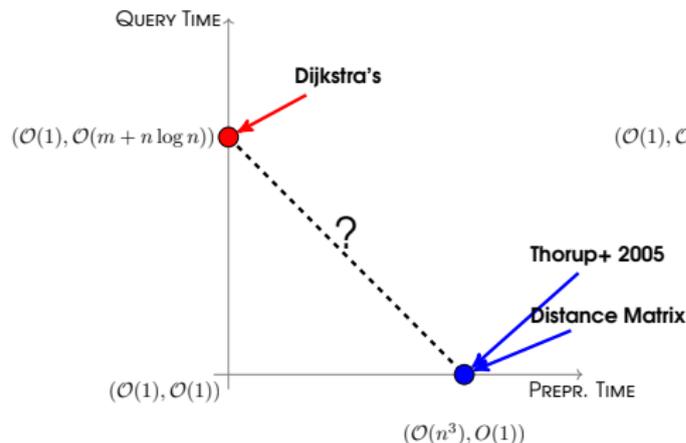
Examples of Trade-Off



M. Thorup, U. Zwick: *Approximate distance oracles*. J. ACM 2005

- Based on building **GRAPH BALLS** of exp. increasing **diameters**
- **PREPROCESSING** in $\mathcal{O}(kmn^{\frac{1}{k}})$ expected time
- **EXTRA SPACE** $\mathcal{O}(kn^{1+\frac{1}{k}})$ – **QUERY** in $\mathcal{O}(k)$ time for any $k \geq 1$
- Distances have **STRETCH** $\leq 2k - 1$ (i.e. approximated for $k > 1$)
- **EQUIVALENT TO DISTANCE MATRIX** for **exact distances** $k = 1$

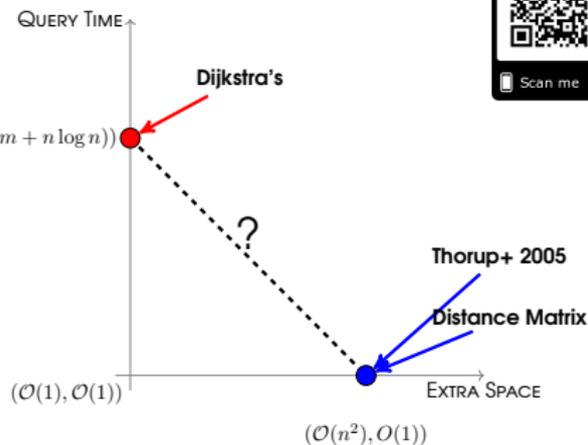
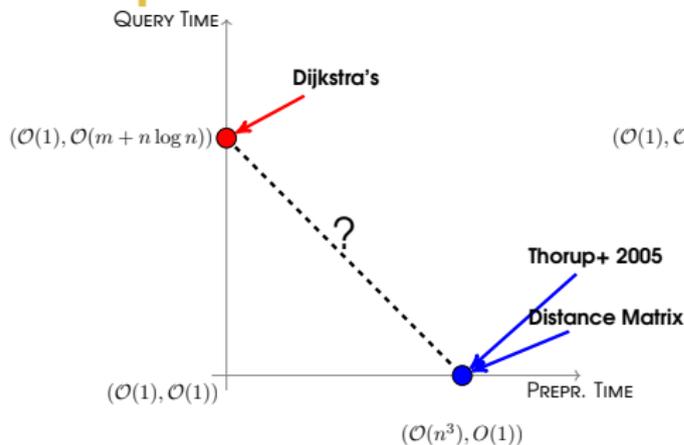
Examples of Trade-Off



M. Thorup, U. Zwick: *Approximate distance oracles*. J. ACM 2005

- Based on building **GRAPH BALLS** of exp. increasing **diameters**
- **PREPROCESSING** in $\mathcal{O}(kmn^{\frac{1}{k}})$ expected time
- **EXTRA SPACE** $\mathcal{O}(kn^{1+\frac{1}{k}})$ – **QUERY** in $\mathcal{O}(k)$ time for any $k \geq 1$
- Distances have **STRETCH** $\leq 2k - 1$ (i.e. approximated for $k > 1$)
- **EQUIVALENT TO DISTANCE MATRIX** for **exact distances** $k = 1$

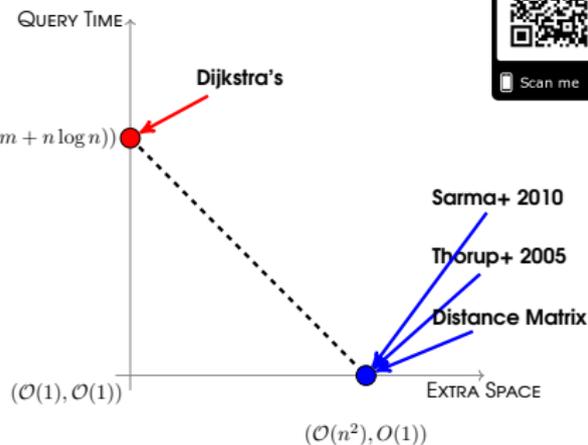
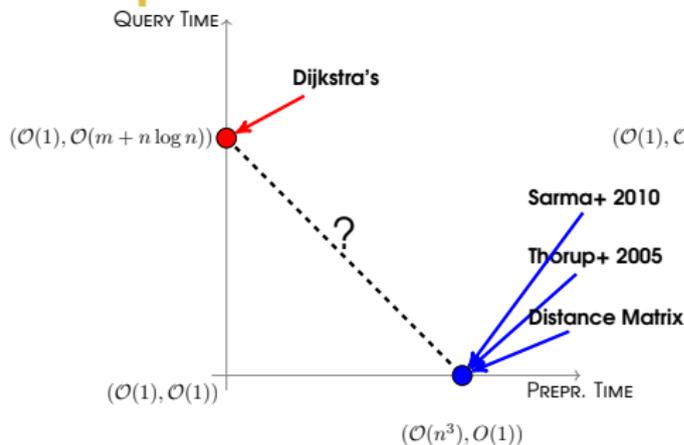
Examples of Trade-Off



A. D. Sarma, S Gollapudi, M. Najork, R. Panigrahy: *A sketch-based distance oracle for web-scale graphs*. WSDM 2010: 401-410

- Simplification of method of Thorup and Zwick via **SEED NODES**
- Same theoretical guarantees on **preprocessing** and **space**
- **QUERY** in $\tilde{\mathcal{O}}(n^{\frac{1}{c}})$ time for any $c \geq 1$, **STRETCH** $2c - 1$
- Better **behavior experimentally**, again **EQUIVALENT TO DISTANCE MATRIX** for $c = 1$

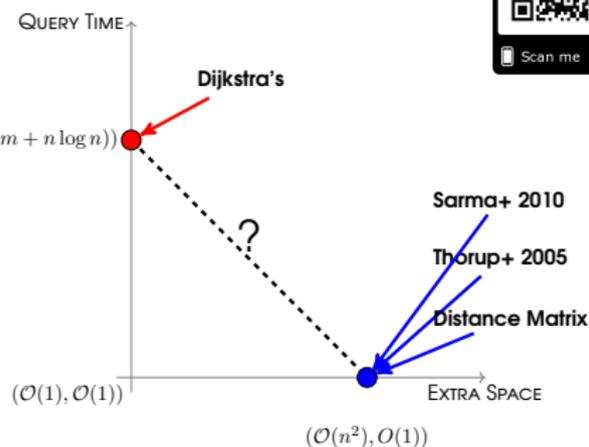
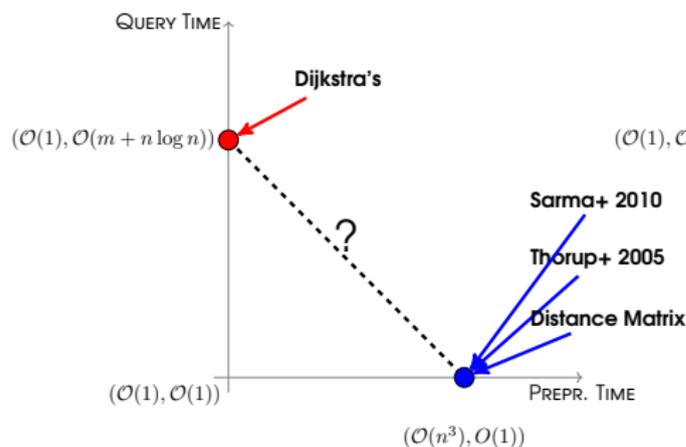
Examples of Trade-Off



A. D. Sarma, S Gollapudi, M. Najork, R. Panigrahy: *A sketch-based distance oracle for web-scale graphs*. WSDM 2010: 401-410

- Simplification of method of Thorup and Zwick via **SEED NODES**
- Same theoretical guarantees on **preprocessing** and **space**
- **QUERY** in $\tilde{\mathcal{O}}(n^{\frac{1}{c}})$ time for any $c \geq 1$, **STRETCH** $2c - 1$
- Better **behavior experimentally**, again **EQUIVALENT TO DISTANCE MATRIX** for $c = 1$

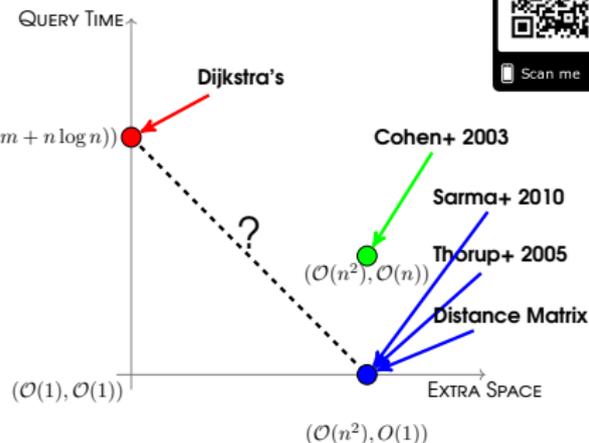
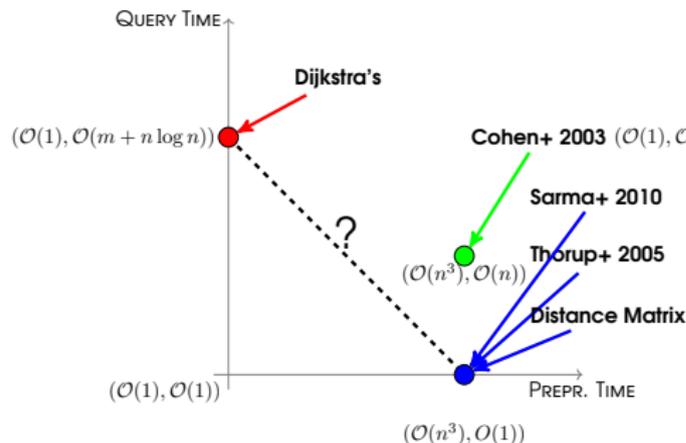
A Notable Trade-Off



E. Cohen, E. Halperin, H. Kaplan, U. Zwick: *Reachability and Distance Queries via 2-Hop Labels*. SIAM J. Comput. 32(5): 1338-1355 (2003)

- Based on the notion of **2-HOP-COVER** ("compact" representation of transitive closure)
- Several works followed on the same ideas [Abraham+ ESA 2012] [Akiba+ EDBT 2012, SIGMOD 2013] [Delling+ ESA 2014]

A Notable Trade-Off



E. Cohen, E. Halperin, H. Kaplan, U. Zwick: *Reachability and Distance Queries via 2-Hop Labels*. SIAM J. Comput. 32(5): 1338-1355 (2003)

- Based on the notion of **2-HOP-COVER** ("compact" representation of transitive closure)
- Several works followed on the same ideas [Abraham+ ESA 2012] [Akiba+ EDBT 2012, SIGMOD 2013] [Delling+ ESA 2014]
- **WORSE** worst-case but **EFFECTIVE IN PRACTICE WITH SUITED HEURISTICS**



ON THE IMPORTANCE OF EXPERIMENTATION and TOOLS FOR (MASSIVE) GRAPH PROCESSING

- **MOST RESULTS** on *shortest-path / distance queries in complex networks* are of **experimental nature**
- **FOR GENERAL GRAPHS** no known exact approach **PROVABLY BETTER** than Dijkstra's and APSP+Distance Matrix in terms of the **three criteria** (query time, extra space, preprocessing time)
- **EXPERIMENTAL EFFORTS** to determine best solutions
- **OF PARAMOUNT IMPORTANCE** having **EFFECTIVE, EASY TO USE TOOLKITS** for
 - manipulation, generation, analysis of **large-scale complex networks**
 - efficient **implementation of graph algorithms**
 - many thanks to **NETWORKIT COMMUNITY** for their effort (more details later)

Outline



- 1 Mining Distances: Problem
- 2 Mining Distances vs Modern Applications
- 3 Scalable Mining of Distances
- 4 2-HOP-COVER
- 5 2-HOP-COVER and Time-Evolving Graphs



2-HOP-COVER



GIVEN DIRECTED WEIGHTED GRAPHS $G = (V, A, w)$ ¹

- $n = |V|$ vertices, $m = |E|$ arcs, weight func. $w : A \rightarrow \mathbb{R}^+$
- Let P_{uv} be **COLLECTION OF SHORTEST PATHS FOR PAIR** $u, v \in V$ in G
- Let $P = \bigcup_{u,v \in V} P_{uv}$ be **COLLECTION OF ALL SHORTEST PATHS** of G

HOP: a triple (h, u, v) where h is a (simple) **path** and u, v are **end-points** of such path

A **SET OF HOPS** H is a **2-HOP-COVER OF** G if and only if:

- For any $s, t \in V$ such that $P_{st} \neq \emptyset$ (pair of connected vertices)
- There exists a **(SHORTEST) PATH** $p \in P_{st}$ and **TWO HOPS** (h_1, s, h) , $(h_2, h, t) \in H$ such that

$$p = h_1 \oplus_h h_2$$

- i.e. p can be reconstructed as **CONCATENATION AT HUB VERTEX** h



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

¹Special cases easy to derive

2-HOP-COVER



IN OTHER WORDS

- A 2-HOP-COVER hop set H allows to **RECONSTRUCT** (the weight of) one shortest path by **CONCATENATING TWO (SHORTEST) PATHS** emanating from s and t at a suited **HUB VERTEX**
- H is said to **COVER** G (or to satisfy **COVER PROPERTY**)
- $|H|$ is the **SIZE** of the 2-HOP-COVER

NAIVE BUILDING of a 2-HOP-COVER

1. Start with $H = \emptyset$
2. **Solve** APSP once
3. For any found shortest path p from s to t
 - $H = H \cup \{(\emptyset, s, s), (p, s, t)\}$
 - Or $H = H \cup \{(h_1, s, h), (h_2, h, t)\}$ **Where** h_1 and h_2 are any two disjoint subpaths of p emanating from a common vertex h

RESULT: H has size $\mathcal{O}(n^2)$ (# triples)

- Moreover **RETRIEVAL** of shortest paths from H requires **SEARCHING**



2-HOP-COVER



MORE EFFICIENT RETRIEVAL

- **CONVERT** into 2-HOP-COVER **distance labeling** data structure
- Well known from distributed computing
- **STORES** data at each vertex in **label form**
- **ALLOWS** retrieval of distances/paths by **accessing only labels of involved vertices**

Populating **2-HOP-COVER DISTANCE LABELING** from 2-HOP-COVER hop set H :

- For any $(h_1, s, h), (h_2, h, t) \in H$
 - **ADD** entry $(h, w(h_1))$ to $L_o(s)$ (**outgoing label** of s) with $w(h_1) = d(s, h)$
 - **ADD** entry $(h, w(h_2))$ to $L_i(t)$ (**incoming label** of t) with $w(h_2) = d(h, t)$

DISTANCE (2-HOP-COVER) LABELING is

$$L = \{ \{L_o(v)\}_{v \in V}, \{L_i(v)\}_{v \in V} \}$$



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

QUERY ALGORITHM for 2-HOP-COVER distance labeling

$$Q(s, t, L) = \begin{cases} \min_{v \in V} \{\delta_{sv} + \delta_{vt} \mid (v, \delta_{sv}) \in L_o(s) \wedge (v, \delta_{vt}) \in L_i(t)\} & \text{if } L_o(s) \cap L_i(t) \neq \emptyset \\ \infty & \text{otherwise} \end{cases}$$

- $L_o(s) \cap L_i(t) \neq \emptyset$ denotes the two label sets share a **common hub vertex**
- If **labels sorted by vertex**, query algo takes

$$\mathcal{O}\left(\max_{s, t \in V, s \neq t} \{\max\{|L_i(s)|, |L_o(t)|\}\}\right)$$

- $\Theta(n)$ with **NAIVE 2-HOP-COVER** computation, on top of $\mathcal{O}(n^2)$ extra space
- **MORE COMPACT HOP SETS/LABELS** necessary for practical usage

THREATS TO SCALABILITY

- **large** label sets: worst case $\mathcal{O}(n)$ per vertex
- **unsustainable** space requirements: worst case $\mathcal{O}(n^2)$
- **impractical** query times: worst case $\mathcal{O}(n)$
- **infeasible** preprocessing: worst case $\mathcal{O}(n^3)$

Scalable 2-HOP-COVER Distance Labelings



(NEGATIVE) FACTS

- **NP-HARD** to compute minimum-sized **2-HOP-COVER hop set** (from min set cover)
- **NP-HARD** to find a corresponding minimum-sized **distance labeling**
- $\Omega(n^{4/3})$ **LOWER BOUND** on the **size of any labeling scheme** (sum of sizes of all label sets)
- A $\mathcal{O}(\log n)$ -factor **APPROXIMATION ALGORITHM** running in $\mathcal{O}(mn^2 \log(\frac{n^2}{m}))$ time is known (again **useless at large scale**)



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Scalable 2-HOP-COVER Distance Labelings



(POSITIVE) FACTS

- Akiba et al. – *Fast exact shortest-path distance queries on large networks by pruned landmark labeling*. SIGMOD 2013: 349-360
- Simple **POLY-TIME HEURISTIC FOR PREPROCESSING** (PLL) that computes **MINIMAL LABELINGS** (ML)
 - Any labeling such that **any removal of any label entry** breaks **COVER PROPERTY**
 - Certain types of ML (**WELL-ORDERED** ones) perform very well in practice
- Variant of PLL, named **RXL**, given in [Delling+ ESA 2014]

INGREDIENTS of PLL/RXL

- **VERTEX ORDERING** (according to some "importance criterion")
- **SHORTEST PATH** (Dijkstra's like) **visits**
- **PRUNING** mechanism



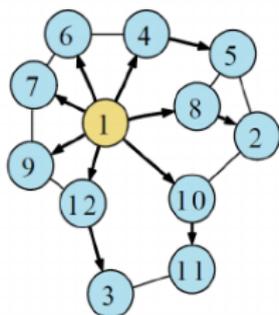
1. **FIX** a **vertex ordering** $\{v_1, v_2, \dots, v_n\}$
2. **PERFORM** $2n$ (n forward, n backward) Dijkstra's-like visits, each rooted at a vertex $v_i \in V$
3. **INCREMENTALLY ENRICH LABELING** L as follows:
 - L^{k-1} status of labeling after execution of SP visits rooted at v_{k-1}
 - Initially $L_i(v)^0 = L_o(v)^0 = \emptyset$
 - 3.1 **DURING** visit **rooted** at v_k on G (or G^T) if **vertex** u **settled** with **distance** δ
 - 3.2 **CHECK** whether $Q(v_k, u, L^{k-1}) \leq \delta$ (or $Q(u, v_k, L^{k-1}) \leq \delta$)
 - 3.3 IF YES \implies visit is **PRUNED** at u
 - 3.4 IF NO \implies **ADD** (v_k, δ) to $L_i(u)$ (or $L_o(u)$) and **CONTINUE**

PRUNING STEP: means L^{k-1} **already covers** pair (v_k, u) (or (u, v_k))

- **Holds** for all pairs (v_k, x) (or (x, v_k)) such that a shortest path from v_k to x (for rom x to v_k) **passes through** u

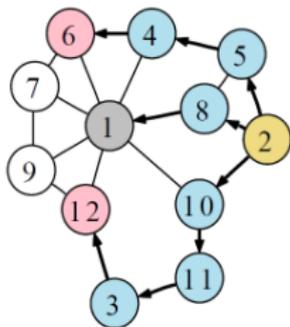
Pruned Landmark Labeling

Preprocessing



First BFS from vertex 1

$L'_1(1):$	Vertex	1
	Distance	0
$L'_1(2):$	Vertex	1
	Distance	2
	⋮	⋮
$L'_1(6):$	Vertex	1
	Distance	1
	⋮	⋮



Second BFS from vertex 2

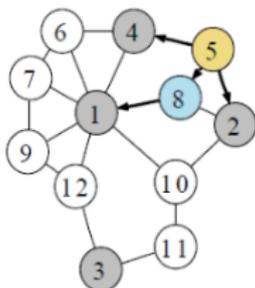
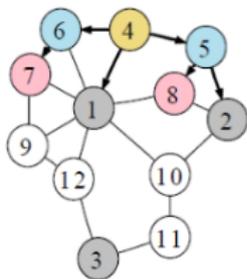
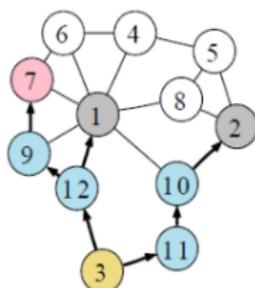
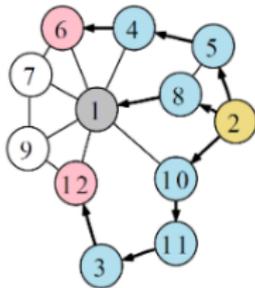
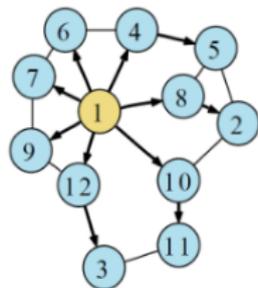
$\text{QUERY}(2, 6, L'_1) = 2 + 1 = 3 = d(2, 6)$
→ Vertex 6 is pruned.



UNIVERSITÀ
GLI STUDI
DELL'AQUILA

Pruned Landmark Labeling

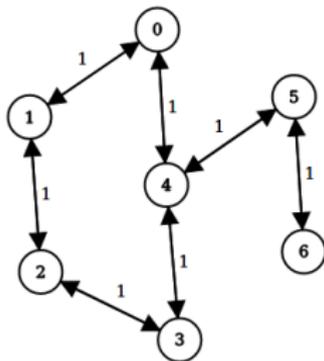
Preprocessing



The search space gets smaller and smaller



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA



VERTEX	$L_o(\cdot)$	$L_i(\cdot)$
0	$\{(4, 1), (0, 0)\}$	$\{(4, 1), (0, 0)\}$
1	$\{(4, 2), (0, 1), (3, 2), (1, 0)\}$	$\{(4, 2), (0, 1), (3, 2), (1, 0)\}$
2	$\{(4, 2), (0, 2), (3, 1), (1, 1), (2, 0)\}$	$\{(4, 2), (0, 2), (3, 1), (1, 1), (2, 0)\}$
3	$\{(4, 1), (3, 0)\}$	$\{(4, 1), (3, 0)\}$
4	$\{(4, 0)\}$	$\{(4, 0)\}$
5	$\{(4, 1), (5, 0)\}$	$\{(4, 1), (5, 0)\}$
6	$\{(4, 2), (5, 1), (6, 0)\}$	$\{(4, 2), (5, 1), (6, 0)\}$

Sample graph and a corresponding 2-HOP-COVER distance labeling w/ **vertex ordering** $\{4, 0, 3, 1, 2, 5, 6\}$

- **WELL ORDERED** property (nice property to exploit)



- Easy to see **LABELING SIZE**, **PREPROCESSING TIME** and **QUERY TIME** depend on **CHOSEN ORDERING** (correctness does not)
- **WORST CASES** (again):
 - preprocessing time $n \times \mathcal{O}(\text{Dijkstra's})$ i.e. $\mathcal{O}(n^3)$
 - extra space $\mathcal{O}(n^2)$
 - query time $\mathcal{O}(n)$
- Clearly **NP-HARD** to find an ordering **yielding optimum**

VERY GOOD EXPERIMENTAL BEHAVIOR when ordering found via **fast-to-compute centrality measures**

- degree, approx betweenness, number of covered pairs (greedy)

GOOD BEHAVIOR means, even on billion-vertex networks:

- **PREPROCESSING** \approx hours
- **SPACE OCCUPANCY** \approx tens of GBs
- **QUERY TIME** \approx milliseconds



instance	label size		preprocessing [s]				space [MiB]				query [μ s]			
	PLL	RXL	PLL	Tree	RXL	CRXL	PLL	Tree	RXL	CRXL	PLL	Tree	RXL	CRXL
Gnutella*	644×16	791	54	209	307	451	209	68	95.7	49.1	5.2	19.0	7.1	45.9
Epinions*	33×16	118	2	128	31	39	32	42	19.1	7.7	0.5	11.0	1.1	4.1
Slashdot*	68×16	219	6	343	85	110	48	83	37.4	17.8	0.8	12.0	1.7	8.0
NotreDame*	34×16	25	5	243	18	22	138	120	22.9	11.9	0.5	39.0	0.2	1.0
WikiTalk*	34×16	113	61	2459	1076	1278	1000	416	560.8	86.5	0.6	1.8	1.0	3.4
Skitter	123×64	273	359	–	2862	3511	2700	–	1074.6	316.7	2.3	–	2.3	20.6
Indo*	133×64	43	173	–	173	201	2300	–	158.6	90.2	1.6	–	0.5	1.8
MetroSec	19×64	116	108	–	2300	2573	2500	–	592.8	207.7	0.7	–	0.8	3.6
Flickr*	247×64	360	866	–	5888	7110	4000	–	1794.6	345.9	2.6	–	2.8	19.9
Hollywood	2098×64	2114	15164	–	61736	75539	12000	–	5934.3	2050.0	15.6	–	13.9	204.0
Indochina*	415×64	91	6068	–	8390	8973	22000	–	1978.8	876.8	4.1	–	0.9	3.9

TODO:

- Evaluate **RXL** on weighted (sparse) digraphs
- Evaluate **CRXL**: compressed version compromising on query time to save space
- Evaluate **APPROXIMATION ALGO**

Limits of Preprocessing in Modern Networks



"Problem": **REAL-WORLD NETWORKS ARE TIME-EVOLVING** (aka *dynamic*)

- Topology and arc weights **likely to change over time**

EXAMPLES:

- **SOCIAL NETWORKS:** new friends, removed friends/pages
- **WEB GRAPHS:** new pages/links, broken links, removed pages
- **BLOGGING:** new replies/posts, removed users/posts/replies
- **COLLABORATION NETWORKS:** new/withdrawn papers
- **INFRASTRUCTURES:** disruptions, new roads, cancelled flights
- **GRAPH DATABASES:** updated/outdated entries



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Limits of Preprocessing



ALL PREPROCESSING-BASED TECHNIQUES suffer of the following issues:

- **PRECOMPUTED DATA** can become **OUTDATED/INCORRECT** due to updates to the graph
- **PRECOMPUTED DATA** require time-consuming preprocessing
- **RE-PROCESSING** after any update: impractical in terms of time overhead
- **ENRICHING** data structure to tolerate updates to graph: infeasible due to huge space overheads

FOR 2-HOP-COVER LABELINGS:

- Label entries can become **outdated** (i.e. hop contain obsolete distances)
- Large number even in presence of **A SINGLE ARC UPDATE**
- Even a single update can lead to **LARGE NUMBER OF INCORRECT ANSWERS TO QUERIES**
 - $q_1(s_1, t_1), q_2(s_2, t_2), \dots$ queries depends on status of graph G_i



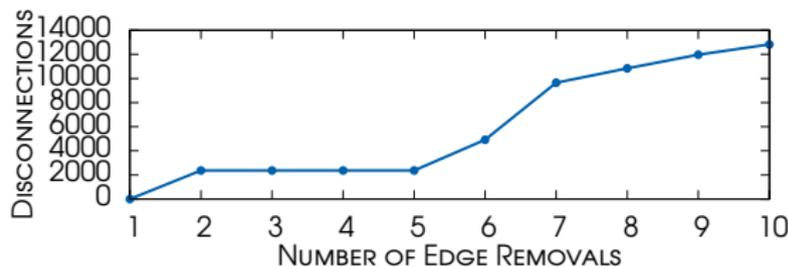
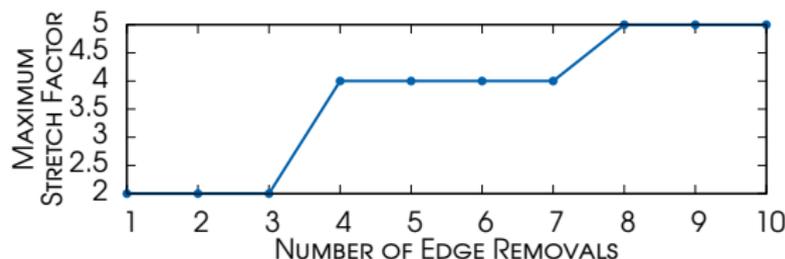
UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Limits of Preprocessing



EFFECTIVE DYNAMIC ALGORITHMS are necessary

- Algorithms able to update only the **part of the data structure** that is compromised by the change
- EFFECTIVE** typically means faster (enough) wrt scratch recomputation



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA



DYNAMIC ALGORITHM: updates existing labeling when **GRAPH UNDERGOES CHANGES**

- Exploit current data structure to identify **compromised entries**
- In this specific case: we want **COVER PROPERTY** to remain true after each update

$$\begin{array}{ccccccccc} G_0 & \rightarrow & G_1 & \rightarrow & \dots & \rightarrow & G_{k-1} & \rightarrow & G_k \\ \downarrow & & \downarrow & & \dots & & \downarrow & & \downarrow \\ L_0 & \rightarrow & L_1 & \rightarrow & \dots & \rightarrow & L_{k-1} & \rightarrow & L_k \end{array}$$



Outline



- 1 Mining Distances: Problem
- 2 Mining Distances vs Modern Applications
- 3 Scalable Mining of Distances
- 4 2-HOP-COVER
- 5 2-HOP-COVER and Time-Evolving Graphs



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

2-HOP-COVER and Time-Evolving Graphs



DYNAMIC PROBLEM, two flavors:

- **INCREMENTAL:** vertex/arc insertions, arc weight decreases
 - Usually **EASIER TO HANDLE**
- **DECREMENTAL:** vertex/arc deletions, arc weight increases
 - Typically **MORE COMPUTATIONALLY CHALLENGING**

DYNAMIC ALGORITHMS FOR 2-HOP-COVER LABELINGS

- **INCREMENTAL ALGORITHM** [Akiba+, WWW 2014]
- **DECREMENTAL ALGORITHM(S)** [D'Angelo, D'Emidio, Frigioni, ACM JEA 2019] [D'Emidio, MDPI Algorithms 2020]



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Incremental Algorithm (RESUME-2HC)

[Akiba+ WWW 2014]



Input: Arc (x, y) undergoes **incremental update**

- 1 **foreach** $v_i \in L(u) \cup L(v)$ **do**
- 2 **RESUME** BFS/Dijkstra's rooted at v_i from vertices x and y ;
- 3 **ADD** new pairs if **pruning test** passed;

MAIN FEATURES:

- **LAZY ALGORITHM:** outdated entries **NOT REMOVED**
- RESUME-2HC only **ADDS SHORTER DISTANCES** induced by incremental updates
 - **REMOVING** non-shortest-path distances is computationally expensive
- **CORRECTNESS** holds since query algo **searches for minimum**
- **LABELING SIZE** inevitably grows with number of updates
- \implies **MINIMALITY NOT PRESERVED**



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Incremental Algorithm (RESUME-2HC)

[Akiba+ WWW 2014]



WORST CASE RUNNING TIME: $\mathcal{O}(n \times \text{Dijkstra's})$

IN PRACTICE

- **VERY EFFECTIVE**, on all tested inputs
- **MILLISECONDS** for updating **extremely large labelings**
- Whereas PLL takes **HOURS OF REPROCESSING**

OPEN PROBLEM: design algorithm that **does not break minimality**

- **PERIODICAL REPROCESSING** necessary if labeling size "grows too much" (performance **degrades over time**)

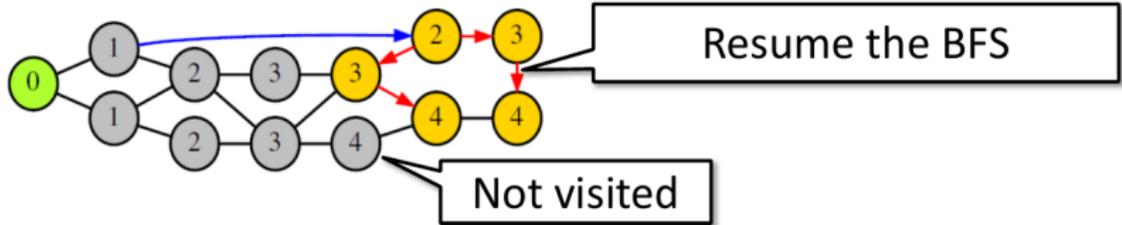
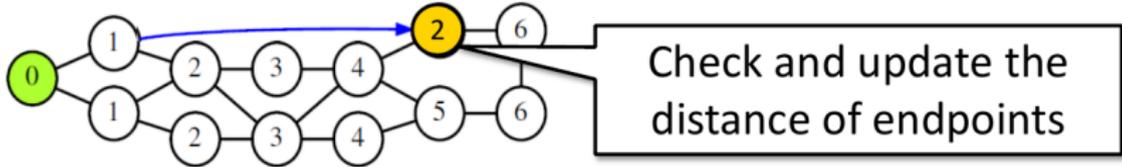
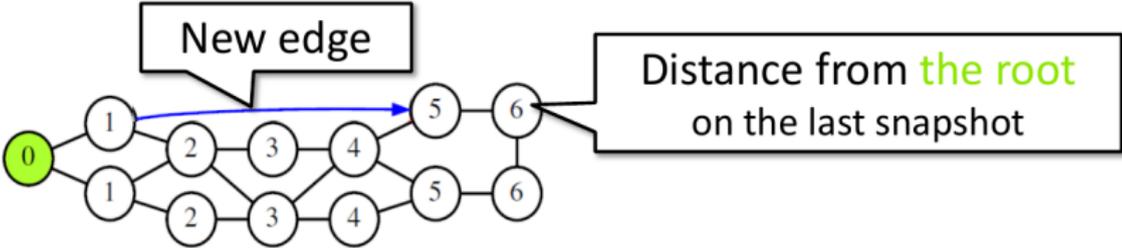


UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Example of RESUME-2HC execution



scan me



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Decremental Algorithm(s)

[D'Angelo, D'Emidio, Frigioni, ACM JEA 2019][D'Emidio, MDPI Algorithm 2020]



DECREMENTAL OPERATIONS more difficult to handle: **OUTDATED ENTRIES MUST BE REMOVED**

- otherwise **correctness not guaranteed**

DECREMENTAL ALGO #1 (BIDIR-2HC) – [D'Angelo, D'Emidio, Frigioni, ACM JEA 2019]

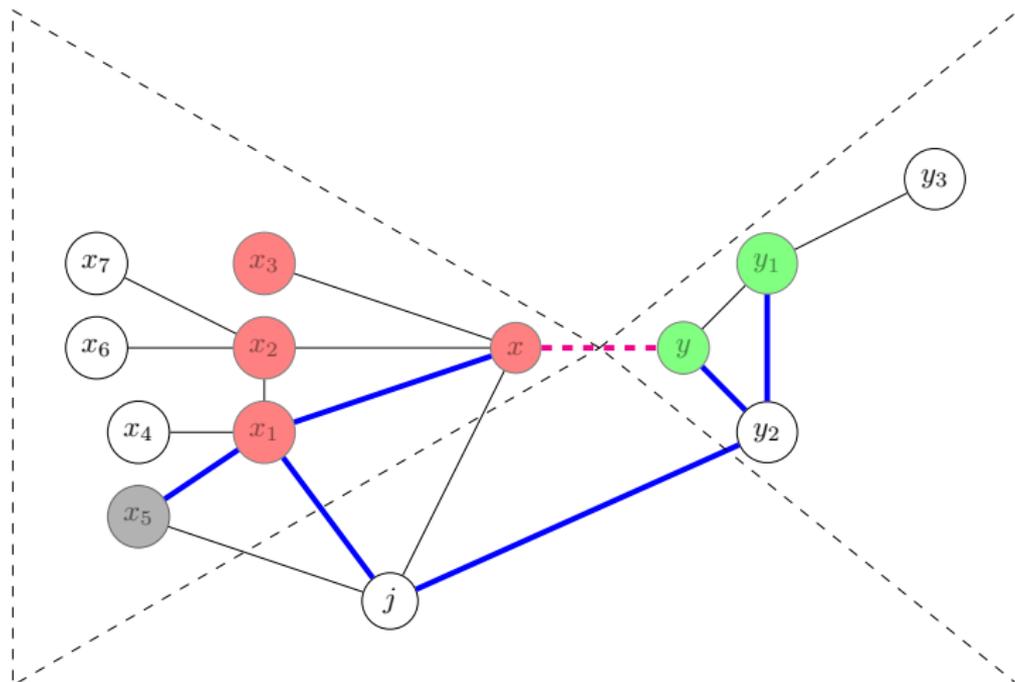
THREE PHASES

1. **IDENTIFICATION OF AFFECTED VERTICES** (potentially containing outdated entries)
 - use **induced paths**
2. **REMOVAL** of outdated (w/ binary search)
3. **RESTORE OF COVER PROPERTY** by **suited SP visits** (in order) rooted at each affected vertex



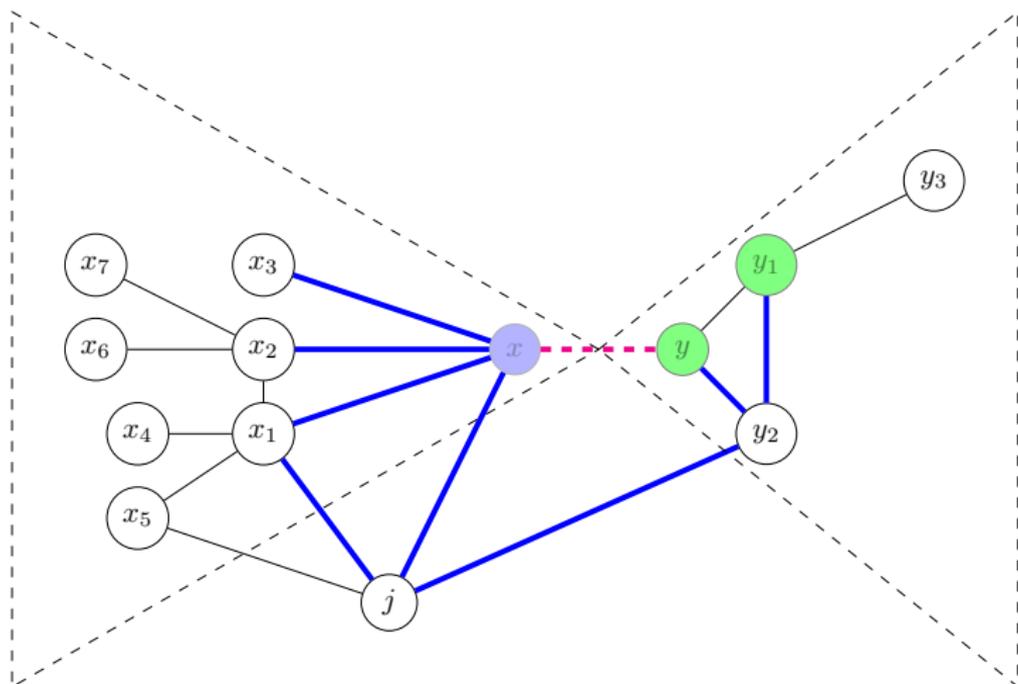
UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

IDENTIFICATION: red/green vs gray vertices
connected by s-paths **containing/not containing modified arc**
(can be checked via content of label sets)



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

RESTORE one **forward** visit (of G) per **red** vertex
one **backward** visit (of G^T) per **green** vertex
(to **re-cover** pairs)





WORST CASE RUNNING TIME: $\mathcal{O}(nm \log n + n^3)$

- Looks bad but in practice **RATHER EFFECTIVE IN ALL INSTANCES**
- At most, on average, **TENS OF SECONDS** for updating **extremely large labelings**
- Where PLL takes **HOURS FOR REPROCESSING**

PROBLEM: slow on some **SPARSE, WEIGHTED DIGRAPHS**

- Not so rare cases **slower than from scratch** (even if **better on average**)



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

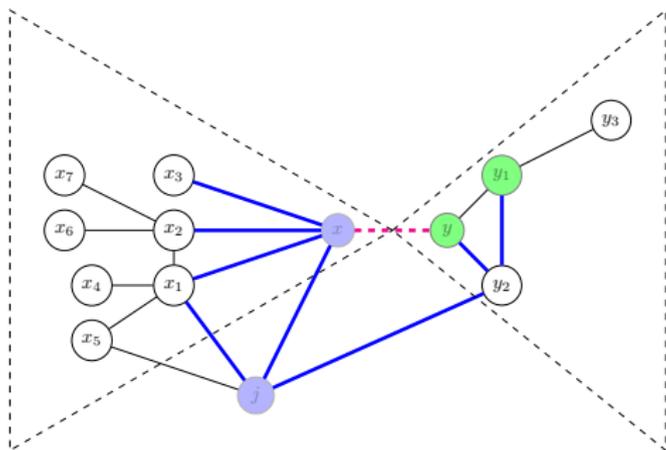
REASON: less effective pruning mechanism

- Leads to unnecessary exploration of parts of the graph
- Large fractions execution time **spent on this step** (**PROFILING**)



LESS EFFECTIVE PRUNING

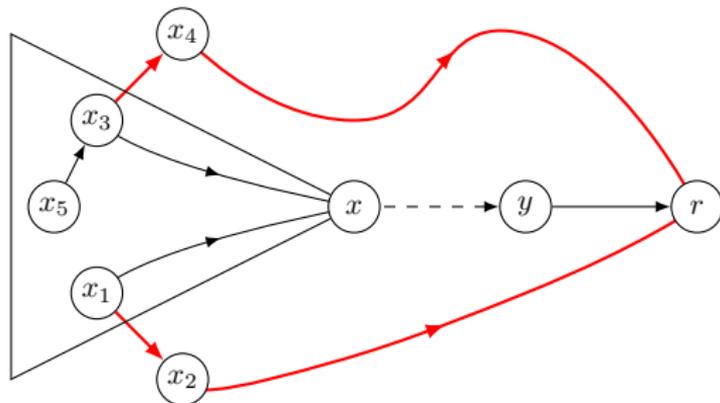
- Visits **traverse non-affected vertices**
- Pruning can stop visit only for **pairs of affected vertices**
- VISIT** from x to y **CANNOT STOP** j (although x and j **are covered**)



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

MAIN DIFFERENCES:

- **IDENTIFICATION** and **REMOVAL** combined in single step (use induced trees)
- **RESTORING DOES NOT TRAVERSE** unchanged vertices
- Exploits **label entries** of unchanged vertices to **AVOID UNNECESSARY EXPLORATIONS** (such entries encode **shortest paths in new graph**)
- Can be used to **re-cover pairs**
- **EVALUATES** them via **PRIORITY QUEUE**, in order



Some Experimentation



Dataset	Network Type	V	E	avg deg	S	D	W
CAIDA (CAI)	ETHERNET	3.20e+04	4.01e+04	2.51	○	○	●
LUXEMBOURG (LUX)	ROAD	3.06e+04	7.55e+04	4.11	○	●	●
WGTGNUTELLA (GNU)	PEER2PEER	6.26e+04	1.48e+05	4.73	○	●	●
BRIGHTKITE (BKT)	LOCATION-BASED	5.82e+04	2.14e+05	7.35	○	○	○
EFZ (EFZ)	RAILWAY	1.25e+05	4.02e+05	6.43	○	●	●
EU-ALL (EUA)	EMAIL	2.65e+05	4.19e+05	2.77	○	●	○
EPINIONS (EPN)	SOCIAL	1.32e+05	8.41e+05	12.76	○	●	○
BARABÁSI-A. (BAA)	SYNTHETIC (Power-Law)	6.32e+05	1.00e+06	3.17	●	○	●
WEB-NOTREDAME (NTR)	HYPERLINKS	3.26e+05	1.09e+06	6.69	○	○	○
NETHERLANDS (NLD)	ROAD	8.92e+05	2.28e+06	5.11	○	●	●
YOUTUBE (YTB)	SOCIAL	1.13e+06	2.99e+06	5.26	○	○	○
WIKITALK (WTK)	COMMUNICATION	2.39e+06	5.02e+06	4.19	○	●	○
HUMAN-GENOME (BIO)	BIOLOGICAL	1.43e+04	9.03e+06	1262.94	○	○	●
AS-SKITTER (SKI)	COMPUTER	1.70e+06	1.11e+07	13.08	○	○	○
DBPEDIA (DBP)	KNOWLEDGE	3.97e+06	1.29e+07	6.97	○	●	○
ERDŐS-RÉNYI (ERD)	SYNTHETIC (Uniform)	1.00e+04	2.50e+07	2499.11	●	○	○

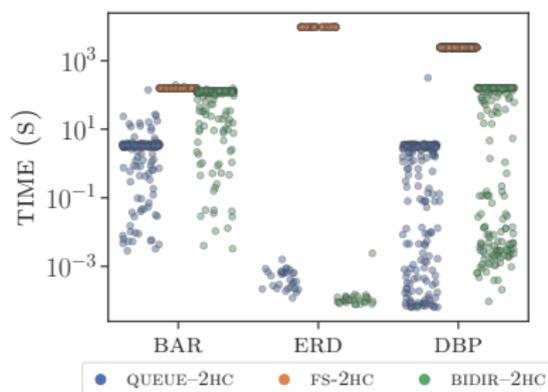
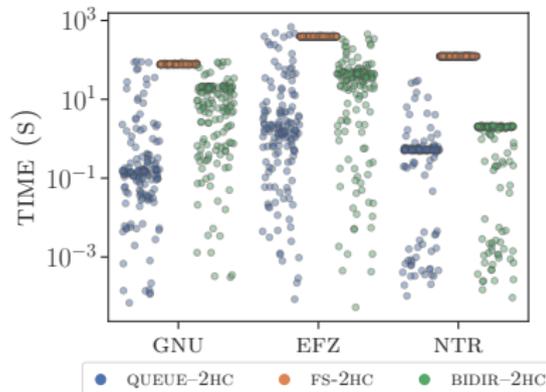
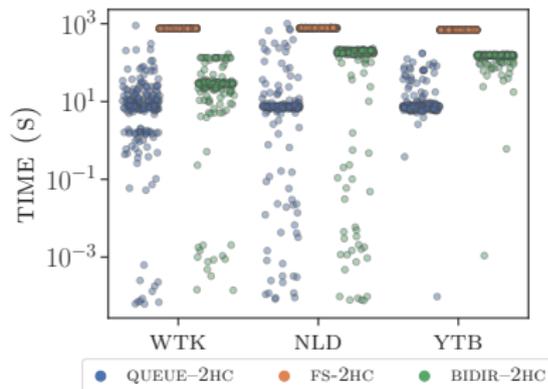
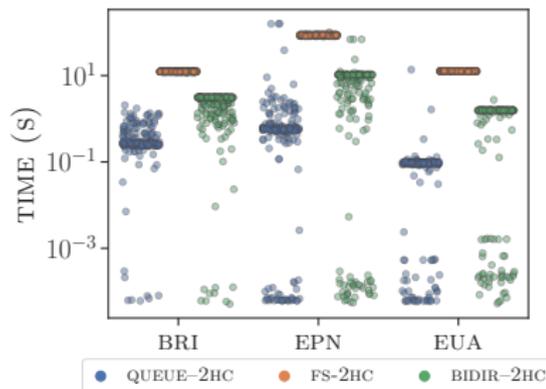


UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Some Experimentation



an me



UNIVERSITÀ
DI STUDI
AQUILA



OPEN (ON-GOING) WORK

- **MINIMALITY PRESERVING** incremental algorithm
- **IMPROVE** further decremental algorithm, or find some lower bound
- **REFINE** theoretical analysis (output bounded sense?)
 - Theoretical foundations to **explain inaccuracy of worst case**
- **BATCH** algorithms
- **ATTACK** other big time-evolving graph mining problems via similar techniques (or adapt dynamic algo to relevant special cases – e.g. **TIMETABLE QUERIES**)
- **DISTRIBUTED** preprocessing/dynamic algorithms
- **STRENGTHEN** experimental results
 - More inputs
 - Better evaluation of RXL/CRXL (adaptation of dyn algo?)
 - Evaluation of apx algo



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

NetworkKit: key features in this context

GRAPHS

- **Easy, effective** graph manipulation/processing tools
- All basic graph algorithms, also **EASILY CUSTOMIZABLE**
- Support for **GRAPH UPDATE OPERATIONS**

NETWORK ANALYSIS

- **PERFORMANCE-CRITICAL ALGORITHMS** implemented in C++/OpenMP
- **CENTRALITY MEASURES** (degree distrib. in $\mathcal{O}(n)$, easily parallelizable)
- **IMPLEMENTATION OF VERY RECENTLY INTRODUCED ALGORITHMS:** parallel implementations of two approx algorithms for Betweenness centrality

GRAPH GENERATORS/INPUT INTERFACES

- Erdős-Renyi Model, Barabasi-Albert **MODELS FOR RANDOM GRAPHS**
- **READERS FOR DATASETS** from known repositories (SNAP, Konect)

INTEGRATION WITH PYTHON FOR DATA ANALYSIS AND INTEROPERABILITY

- pandas, numpy, scipy, networkx

FOR EDUCATIONAL PURPOSES (courses: big data processing, algorithm engineering)

- **interactive workflow** and seamless **Python integration**



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

(Some) Publications supported by NetworkKit



M. D'Emidio, I. Khan, D. Frigioni: Journey Planning Algorithms for Massive Delay-Prone Transit Networks. *Algorithms* 13(1): 2 (2020)

M. D'Emidio: Faster Algorithms for Mining Shortest-Path Distances from Massive Time-Evolving Graphs. *Algorithms* 13(8): 191 (2020)

G. D'Angelo, M. D'Emidio, D. Frigioni: Fully Dynamic 2-Hop Cover Labeling. *ACM J. Exp. Algorithmics* 24(1): 1.6:1-1.6:36 (2019)

M. D'Emidio, I. Khan: Dynamic Public Transit Labeling. *ICCSA (1) 2019*: 103-117



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA



Thanks for your attention

Q&A

`mattia.demidio@{univaq,gssi}.it`
`www.mattiademidio.com`



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA