Massive inputs vs massive decentralization

Some algorithmic challenges in modern computing systems

Mattia D'Emidio[1]

Scan me

[1]Assistant Professor (RTD-B) @ **University of L'Aquila**
email: `mattia.demidio@gmail.com`
web: `www.mattiademidio.com`

December 2, 2021

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

1 **Research Themes**

2 **Scalable Graph Algorithms**
- On The Importance of Algorithms for Mining Graphs
- On the Importance of Scalability
- Scalable Mining of Distances

3 **Algorithms for Multi-Entity Computing Systems**
- Multi-Entity Computing Systems and Applications
- Computability and Algorithms for MCSs
- Programmable Matter

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

Outline

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

Areas of expertise

- **Algorithm engineering:** **theory and experimentation** in algorithmics
  - On the importance of combining the tools of the theoretician with careful **implementations, experimentation** and **data analysis**
- **Graph algorithms:** **design, analysis, efficient implementation of algorithms** for real-world applications that manage graphs
  - Focus on **dynamic graph algorithms:** processing graphs that **evolve over time**
- **Massive datasets:** challenges posed by **processing of massive datasets**
  - effective algorithmic frameworks, **massively parallel computing systems**
- **Distributed computing:** algorithms for **decentralized systems**
  - networks, swarms of robots, multi-agent systems, programmable matter

Teaching both doctoral and master's level courses on:

- **Design and Implementation of Algorithms**
- **Algorithm Engineering**
- **Big Data: Models and Algorithms**
- **Distributed Systems**

UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Research Activities

Currently two active lines:

1. **Algorithm engineering applied to (scalable) graph algorithms**
   - **Design, analysis, implementation, experimentation of algorithms** for graph problems that scale well with size
   - **Focus on real-world applications** that need to extract topological properties from **massive (possibly time-evolving) graph datasets graph with very low execution times** (e.g. social networks, web datasets, biological datasets, transport systems)
2. **Distributed algorithms for multi-entity computing systems**
   - Investigation on **computational properties** and on **design and analysis algorithms** for distributed systems of "mobile" autonomous entities
   - **Focus on emerging technologies** such as, e.g. swarm robotics, networks of software agents, systems of programmable particles

Main Objective of this presentation: high-level survey on **research activities** in these areas

- Few technical details, see references for more details
- We have **several active projects** related to **research themes:**
  - Possibility of **thesis** [a]

[a] www.mattiademidio.com

Some refs **TO KNOW MORE**

Gianlorenzo D'Angelo, Mattia D'Emidio, Shantanu Das, Alfredo Navarra, Giuseppe Prencipe: **Asynchronous Silent Programmable Matter Achieves Leader Election and Compaction.** IEEE Access 8: 207619-207634 (2020)

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

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

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

Related secondary/past topics I have been investigating:

- **Computational geometry** algorithms for CAD tools (schematization, decomposition, simplification problems)
- **Massively parallel computing systems** (MapReduce paradigm, Apache Spark)
- **Distributed routing algorithms** (algorithms for dynamic routing tables)

Some refs **to know more**

S. Cicerone, M. D'Emidio, D. Frigioni, F.T. Pascucci: **Combining Polygon Schematization and Decomposition Approaches for Solving the Cavity Decomposition Problem**. ACM Trans. Spatial Algorithms Syst. 7(4): 22:1-22:37 (2021)

S. Cicerone, M. D'Emidio, G. Di Stefano, A. Navarra: **On the effectiveness of the genetic paradigm for polygonization**. Inf. Process. Lett. 171: 106134 (2021)

G. D'Angelo, M. D'Emidio, D. Frigioni: **A loop-free shortest-path routing algorithm for dynamic networks.** Theor. Comput. Sci. 516: 1-19 (2014)

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

Outline

1 **Research Themes**

2 **Scalable Graph Algorithms**
- On The Importance of Algorithms for Mining Graphs
- On the Importance of Scalability
- Scalable Mining of Distances

3 **Algorithms for Multi-Entity Computing Systems**
- Multi-Entity Computing Systems and Applications
- Computability and Algorithms for MCSs
- Programmable Matter

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

On The Importance of Algorithms for Mining Graphs

GRAPH DATASETS are everywhere in (modern) computing/information systems

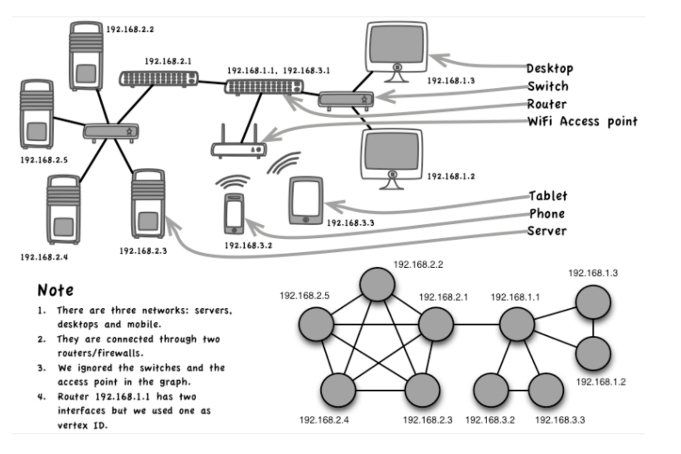
- model **binary relationships** between **individual entities**
- thus is an **extremely common data structure**
- essentially all modern applications exploit **graph modeling of data**
- essentially all modern applications exploit need graph algorithms for **effective processing**
  - for **OPTIMIZATION PURPOSES** (e.g. routing, network design, scheduling, transportation, logistics)
  - for **ANALYTICAL/INFORMATION DISCOVERY PURPOSES** (e.g. social network analysis, web indexing, bioinformatics)

Reason why **HUGE AMOUNT OF RESEARCH** is/has been devoted to such **structures**, their **properties** and to **designing suited algorithms**

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

On The Importance of Algorithms for Mining Graphs: Examples

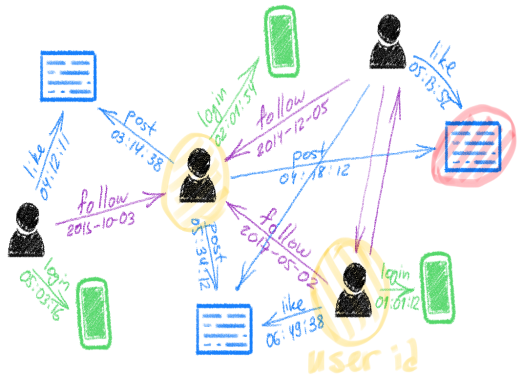
Routing in communication networks selection of small latency paths achieved via (various types of) shortest-path algorithms (vertices are network nodes, arcs are network links, weights are latencies)

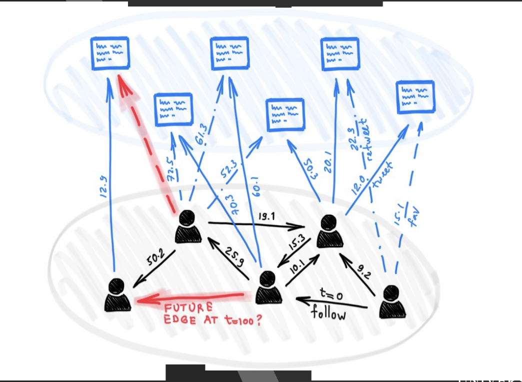


UNIVERSITÀ DEGLI STUDI DELL'AQUILA

On The Importance of Algorithms for Mining Graphs: Examples

Social network analysis: identification of communities, link prediction, detection of malicious behaviors by pattern detection algorithms or via extraction of various topological properties (vertices are entities of the social network, arcs are connections - e.g. friendship or "follows" - between entities)

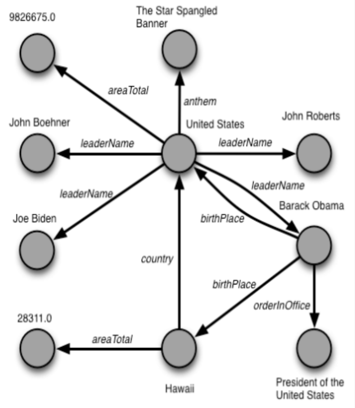




On The Importance of Algorithms for Mining Graphs: Examples

Similarly for **Semantic Networks as Graphs** (network properties, or useful patterns, via graph algorithms)

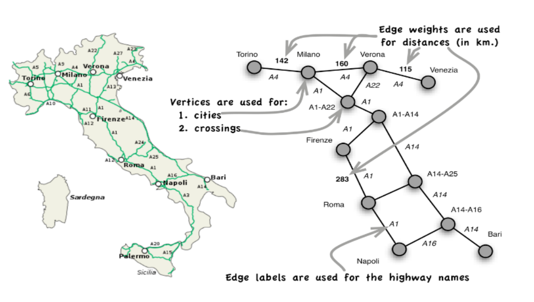
Subject	Predicate	Object
United States	areaTotal	9826675.0
United States	anthem	The Star Spangled Banner
United States	leaderName	Barack Obama
United States	leaderName	Joe Biden
United States	leaderName	John Boehner
United States	leaderName	John Roberts
Barack Obama	birthPlace	United States
Barack Obama	birthPlace	Hawaii
Barack Obama	orderInOffice	President of the United States
Hawaii	areaTotal	28311.0
Hawaii	country	United States

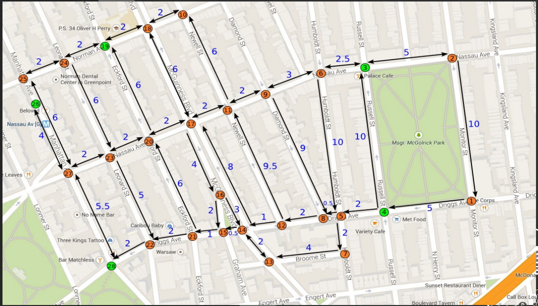


UNIVERSITÀ DEGLI STUDI DELL'AQUILA

On The Importance of Algorithms for Mining Graphs: Examples

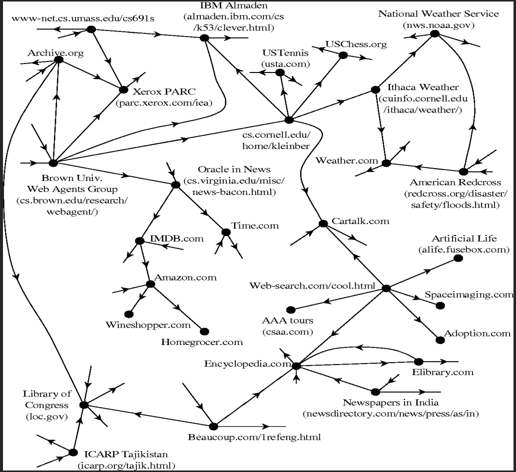
Optimization of Transport Networks selection of best paths (low travel time, low monetary cost, passing through some city or train station) via (various types of) shortest-path algorithms (vertices are crossings or locations, arcs are road segments or connections, weights are costs/times)

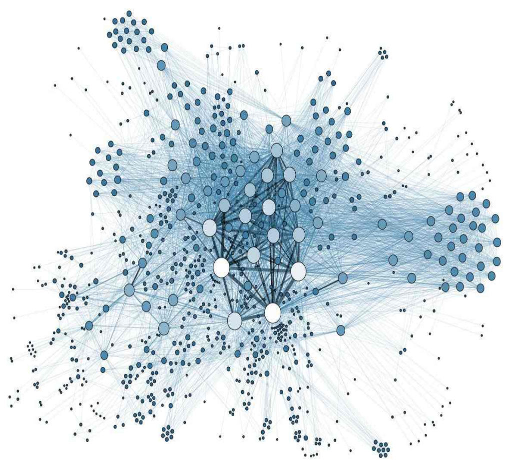




On The Importance of Algorithms for Mining Graphs: Examples

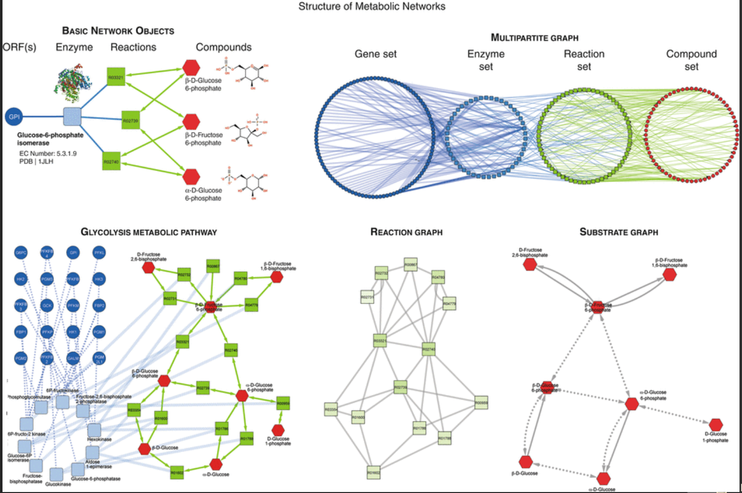
Web indexing, ranking, classification rank/cluster/index/query/similarity through various kinds of graph algorithms (vertices are pages, arcs are links, weight is probability of traversal)





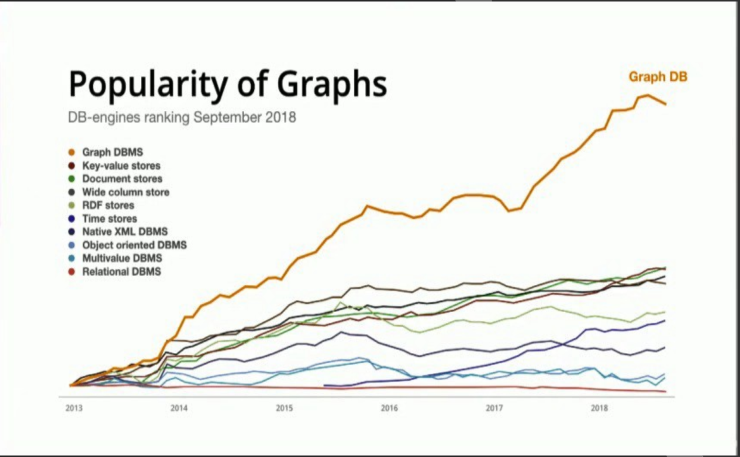
On The Importance of Algorithms for Mining Graphs: Examples

Metabolic Processes find/analyze interactions between compounds or expressions of genes via graph properties/subgraphs/frequent patterns/enumeration



On The Importance of Algorithms for Mining Graphs: Examples

MANY OTHERS: graph databases, network design, machine learning, scheduling, distributed systems ...



On the Importance of Scalability

For many graph-related problems

Huge amount of literature and results, many algorithms, studies on computational properties, lower/upper bounds, hardness or approximation, classification in classes of problems

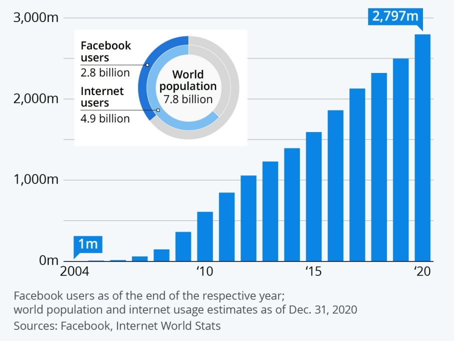
- Asymptotically optimal/near optimal solutions for most problems in class P
- Several good approximation algorithms for problems admitting bounded approximation

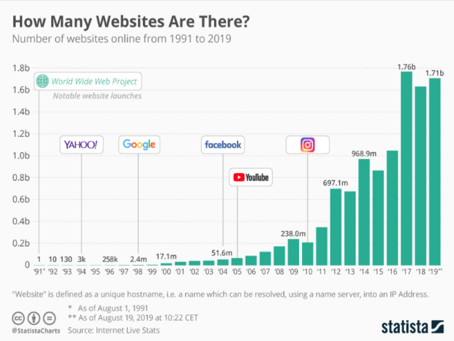
The curse of big data

- Several methods suffer of **scalability issues** against "modern inputs" (**billions vertices/arcs**, e.g. twitter, google maps, www)
  - **Massive graphs**
- Big datasets **challenge** the classical **notion of efficient algorithms**
- Algorithms that used to be considered **efficient**, according to **polynomial-time characterization**, may **no longer be adequate for solving today's problems**
- Do not **scale well** with respect to sizes or volumes

Motivation for Scalability

Trend in terms graph sizes





Trend in terms of volumes of executions:

- **Google** (web indexing and retrieval): estimated approximately avg 63 000 **search queries every second**, translating to 5.6 **billion searches per day** and roughly 2 **trillion per year**
- **Google Maps** (route planning): 50 **requests per second per user**

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

The Quest for Scalable Graph Algorithms

In several big-data applications not **just desirable** but **essential** to design **scalable algorithms**

- Their complexity should be **nearly linear/linear or sub-linear** wrt input size
- **Scalability**, in these cases, is elevated as the **central complexity notion** to characterize efficiency (not just polynomial-time computability)

Basic definition of scalability: algorithm A is **scalable** if there exists a constant $c > 0$ such that $\text{SCALABILITY}_A(n) = O(\log^c n)$ where $\text{SCALABILITY}_A(n) = \frac{T_A(n)}{n}$ and $T_A(n)$ is (worst-case) complexity of A on inputs of size n

- When $c = 0$, we say A is **linearly-scalable**
- Various other definitions for other values of c to better **capture** the differences in terms of efficiency (e.g. **parallel scalability or super scalability**)

Shang-Hua Teng: Scalable Algorithms for Data and Network Analysis. Found. Trends Theor. Comput. Sci. 12(1-2): 1-274 (2016)

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

On Achieving Scalability

Several problems/algorithms revisited in a **scalability-oriented perspective**

- **Very active research line** on designing scalable algorithms
- **Various techniques** besides restricting the focus on special input classes
  - **Approximation:** relaxing on optimality constraints for faster (though less accurate) results
  - **Sampling:** sample the input to compute solutions that have small (or no) error with some probability
  - **Parallelism:** faster executions via **parallel architectures** (mention **Apache Spark**)
  - **Preprocessing:** preprocess the input in an offline, una tantum step, exploit precomputed data to accelerate "online" executions

[D. Delling, A. V. Goldberg, T. Pajor, R. F. Werneck: **Robust Distance Queries on Massive Networks.** ESA 2014: 321-333]
[C. Schulz: **Scalable Graph Algorithms.** CoRR abs/1912.00245 (2019)]
[A. Conte, D. De Sensi, R. Grossi, A. Marino, L Versari: **Truly Scalable K-Truss and Max-Truss Algorithms for Community Detection in Graphs.** IEEE Access 8: 139096-139109 (2020)]

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

An example where **preprocessing** shown very effective:

Mining of distances/shortest paths

- **Given** (di)graph $G = (V, A)$, answer to **(distance) queries** $q(s,t)$ for **pairs of vertices** $s, t \in V$
- **Report distance** $d(s,t)$ **(weight of a shortest path** (or entire path) from s to t in G) as fast as possible

Widely studied problem tons of applications (routing, journey planning, recommendation systems, network analysis), **huge amount of research/literature**

Textbook/Standard Solutions

1. Solve **single source shortest paths problem** upon query (e.g. by **Dijkstra's**)
   - for an n-vertex, m-arc graph, $\mathcal{O}(m + n \log n)$ **time per query**
   - **no preprocessing, no extra space**
2. **Preprocess the graph** to solve **all pairs shortest paths problem** only once (e.g. via **Floyd-Warshall**), store results in **distance matrix**
   - $O(1)$ **time per query**
   - $\mathcal{O}(nm + n^2 \log n) \in \mathcal{O}(n^3)$ **preprocessing time**, $n \times n = \Theta(n^2)$ **extra space**

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

BIG GRAPHS, BIG PROBLEMS: both not suited from **MASSIVE GRAPHS**, do not scale well in terms of **time** (or **space**)

- **QUERY TIME** not suited for interactive applications (up to tens of **seconds per query**)
- Extra **SPACE OVERHEAD** impractical (thousands of GBs when $n \gg 10^6$)
  - Difficult/impossible to store on single machine
- **PREPROCESSING TIME** unacceptable (days when $n \gg 10^6$)

EFFORT to find scalable trade-offs

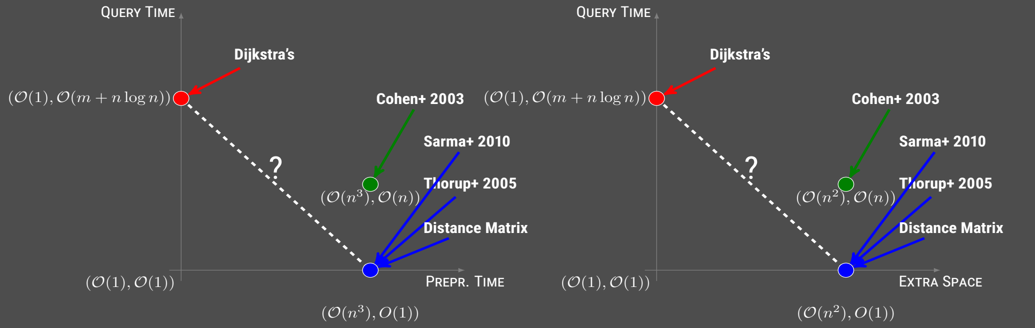
VERY ACTIVE RESEARCH LINE: some recent literature (non–exhaustive list):

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

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

Scalable Mining of Distances

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



AN UNEXPECTED BREAKTHROUGH: [Cohen+, 2003] not really **considered** initially **for usage in practice** since **WORST CASE (TIME,SPACE)** is **WORSE** than other approaches with respect to all criteria but then . . .

[T. Akiba, Y. Iwata, Y. Yoshida: **Fast exact shortest-path distance queries on large networks by pruned landmark labeling.** SIGMOD 2013: 349-360]

- Method has been improved to **be practical** by incorporating **SUITED HEURISTICS**
- **Tuning** and **experimental validation** to show it is **MOST EFFECTIVE SOLUTION IN PRACTICE**
- **THE KEY ROLE OF ALGORITHM ENGINEERING**
  - **MOST RECENT RESULTS** on *scalable graph algorithms* are of **experimental nature**
  - Effective **implementation** and **testing** to identify best solutions combined with theoretical efforts

[Angriman+: **Guidelines for Experimental Algorithmics: A Case Study in Network Analysis.** Algorithms 12(7): 127 (2019)] in Special Issue "Algorithm Engineering: Towards Practically Efficient Solutions to Combinatorial Problems".

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

2-HOP-COVER

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

- $n = |V|$ vertices, $m = |E|$ arcs, weight func. $w : A \to \mathbb{R}^+$
- 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 **endpoints** 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

[1] Special cases easy to derive

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

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 (e.g. FW or repeated Dijkstra's)
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** ($O(|H|)$)

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

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

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

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

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

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

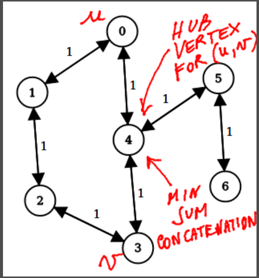
THREATS TO SCALABILITY

- **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 $\mathcal{O}(n^3)$

2-HOP-COVER Labeling of a Graph

COMPACT REPRESENTATION OF SHORTEST PATHS: precompute small set of **hub vertices**, assign **distance labels to vertices (from/to hubs)**, use labels to **retrieve distances/paths** by concatenations at hubs

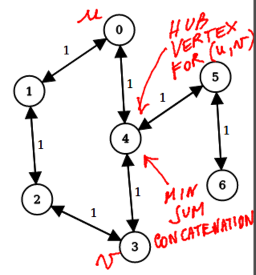
VERTEX	LABELS	
	$L_o(\cdot)$	$L_i(\cdot)$
0	$\{$**(4,1)**$, (0,0)\}$	$\{(4,1),(0,0)\}$
1	$\{(4,2),(0,1),(3,2),(1,0)\}$	$\{(4,2),(0,1),(3,2),(1,0)\}$
2	$\{(4,2),(0,2),(3,1),(1,1),(2,0)\}$	$\{(4,2),(0,2),(3,1),(1,1),(2,0)\}$
3	$\{$**(4,1)**$, (3,0)\}$	$\{(4,1),(3,0)\}$
4	$\{(4,0)\}$	$\{(4,0)\}$
5	$\{(4,1),(5,0)\}$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



UNIVERSITÀ DEGLI STUDI DELL'AQUILA

2-HOP-COVER Labeling of a Graph

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

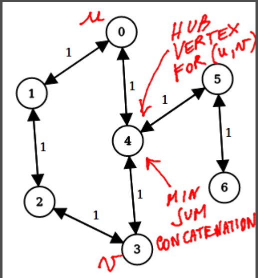


NEGATIVE FACTS:

- [✗] **Naive computation** yields $O(n^2)$ **space,** $O(n^3)$ **prepr. time,** $O(n)$ **query** (**NOT SCALABLE**)
- [✗] **NP-HARD** to build minimum-sized **2-HOP-COVER labeling**
- [✗] **LOWER BOUND** $\Omega(n^{4/3})$ on **size**
- [✗] $\mathcal{O}(\log n)$ **APPROXIMATION ALGORITHM** runs in $\mathcal{O}(mn^2\log(\frac{n^2}{m}))$ time (**NOT SCALABLE**)

2-HOP-COVER Labeling of a Graph

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

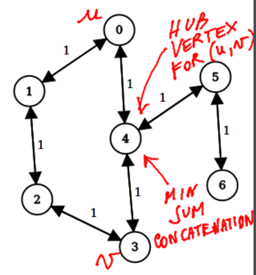


POSITIVE FACTS:

- [✓] **POLY-TIME HEURISTIC FOR PREPROCESSING** (PLL), no bound on approximation but shown experimentally **TO OUTPERFORM** all other approaches (relies on finding a **"good" vertex ordering** and a **minimal labeling**)
- [✓] Suited for **DISTRIBUTION** (query **accesses queried vertices only**)

2-HOP-COVER Labeling of a Graph

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



POSITIVE FACTS: [✅] further improved (**RXL**) in [D. Delling, A. V. Goldberg, T. Pajor, R. F. Werneck: **Robust Distance Queries on Massive Networks.** ESA 2014: 321-333]

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

INGREDIENTS of PLL/RXL

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

1. **FIX** a **vertex ordering** $\{v_1, v_2, \ldots, v_n\}$
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 $\Longrightarrow$ visit is **PRUNED** at u

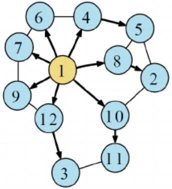
   3.4 IF NO $\Longrightarrow$ **ADD** (v_k, δ) to $L_i(u)$ (or $L_o(u)$) and **CONTINUE**

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

Basics of preprocessing

Greedy approach, progressively **shrink search space** by exploiting **partially precomputed** labeling



First BFS from vertex 1

$L'_1(1)$:

Vertex	1
Distance	0

$L'_1(2)$:

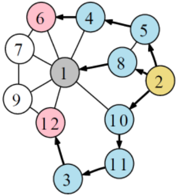
Vertex	1
Distance	2

⋮

$L'_1(6)$:

Vertex	1
Distance	1

⋮



Second BFS from vertex 2

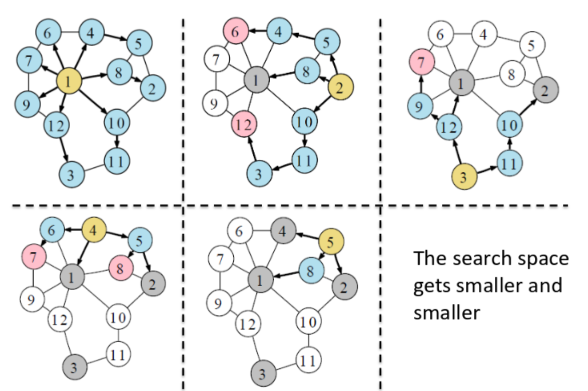
$\textsc{Query}(2, 6, L'_1) = 2 + 1 = 3 = d(2,6)$

→ Vertex 6 is pruned.

Vertex ordering $\{1, 2, 3, \ldots, 12\}$

Basics of preprocessing

Maintaining **cover property** across visits



Vertex ordering $\{1, 2, 3, \ldots, 12\}$

Performance

- 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
- **DISTRIBUTABLE**

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

	label size		preprocessing [s]				space [MiB]				query [µs]			
instance	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 wrt experimentation:

- 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:

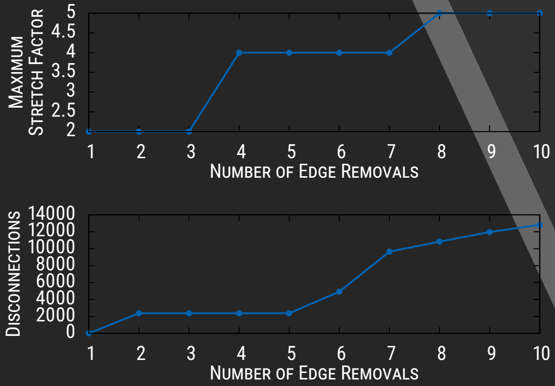
- Label entries can become **outdated** (i.e. hop contain obsolete distances)
- Large number even in presence of **a single arc update**
- Even a single update can lead to **large number of incorrect answers to queries**
  - $q_1(s_1, t_1), q_2(s_2, t_2), \ldots$ queries depends on status of graph G_i

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

FURTHER NON TRIVIAL POSITIVE FACT:

- [✔] 2-HOP-COVER can be adapted to work well in case **TIME-EVOLVING GRAPHS**, as **shown** in

  [T. Akiba, Y. Iwata, Y. Yoshida: **Dynamic and historical shortest-path distance queries on large evolving networks by pruned landmark labeling.** WWW 2014: 237-248]
  [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: **Faster Algorithms for Mining Shortest-Path Distances from Massive Time-Evolving Graphs.** Algorithms (Special Issue Algorithmic Aspects of Networks) 13(8): 191 (2020)]

TIME-EVOLVING GRAPHS: change over time, **most common case in practice** (e.g. social networks, or road networks)

- **PREPROCESSING** is affordable but still time-consuming, **CANNOT BE REPEATED EVERYTIME SOMETHING CHANGES**
- **DYNAMIC GRAPH ALGORITHMS:** update preprocessed data selectively, only the **PART OF THE DATA STRUCTURE** that is compromised by the change

DYNAMIC ALGORITHMS FOR 2HC LABELING: to save time, identify **parts of the labeling** that are not **compromised** by the changes, avoid unnecessary **exploration of (large) part of graph** (**avoid recomputations** that are not necessary)

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

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

Incremental Algorithm (RESUME−2HC)

[Akiba+ WWW 2014]

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

foreach $v_i \in L(u) \cup L(v)$ **do**

  Resume BFS/Dijkstra's rooted at v_i from vertices x and y;

  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
- $\Longrightarrow$ **Minimality not preserved**

Università degli Studi dell'Aquila

Incremental Algorithm (RESUME−2HC)

[Akiba+ WWW 2014]

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

IN PRACTICE

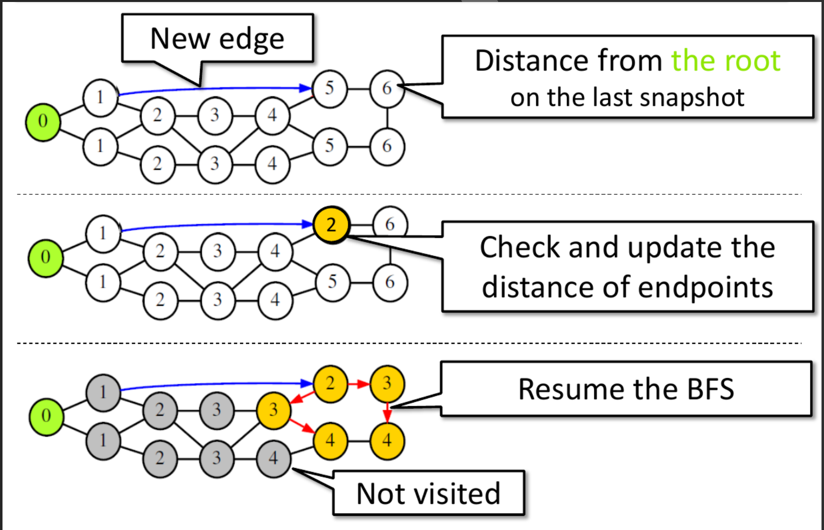
- **VERY EFFECTIVE**, on all tested inputs
- **MILLISECONDS** for updating **extremely large labelings**
- Whereas PLL takes **HOURS OF REPREPROCESSING**

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



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**

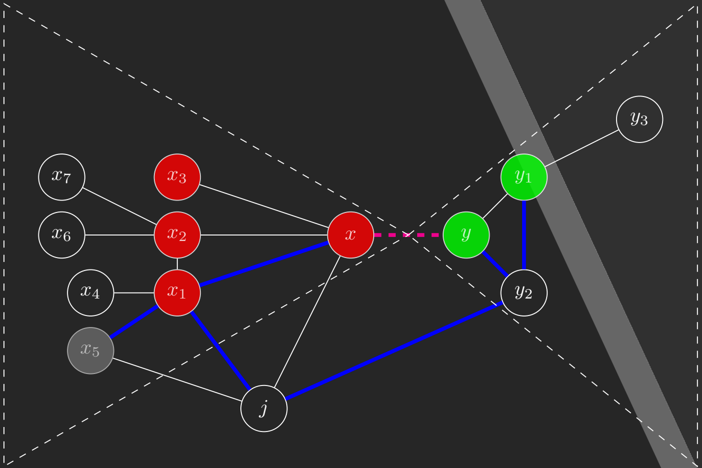
DECREMENTAL ALGO #1 (BIDIR−2HC) – [D'Angelo, D'Emidio, Frigioni, ACM JEA 2019]

THREE PHASES

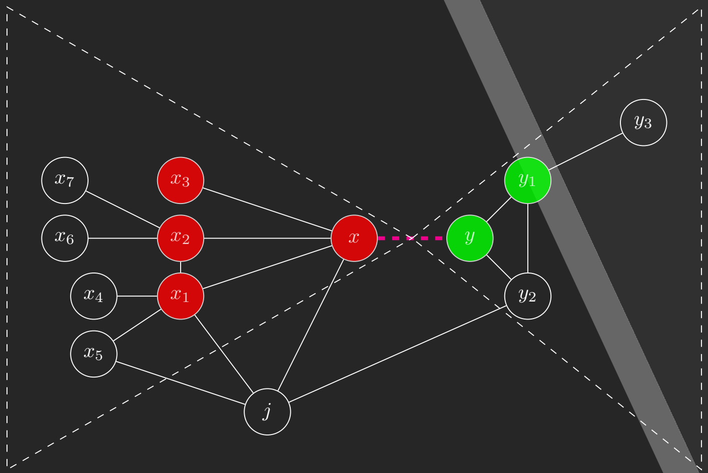
1. **IDENTIFICATION OF AFFECTED VERTICES** (potentially containing outdated entries)
   - use **induced paths**
2. **REMOVAL** of outdated (w/ binary search)
3. **RESTORE OF COVER PROPERTY** by **suited SP visits** (in order) rooted at each affected vertex

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

IDENTIFICATION: red/green vs gray vertices connected by paths **containing/not containing modified arc** (check via content of label sets)

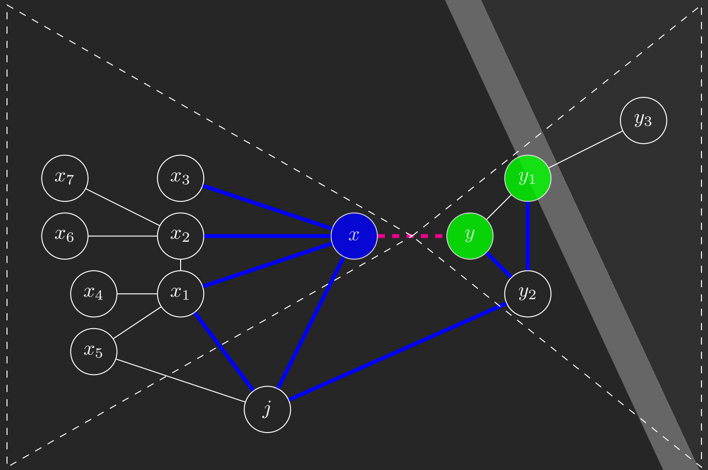


REMOVAL of green entries from red outgoing labels and red entries from green incoming labels (linear scan)



UNIVERSITÀ DEGLI STUDI DELL'AQUILA

RESTORE one **forward** visit (of G) per red vertex and 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 REPREPROCESSING**

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

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

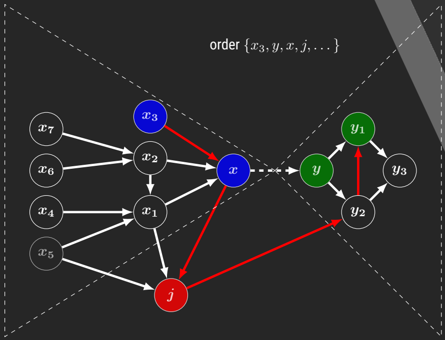
UNIVERSITÀ DEGLI STUDI DELL'AQUILA

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

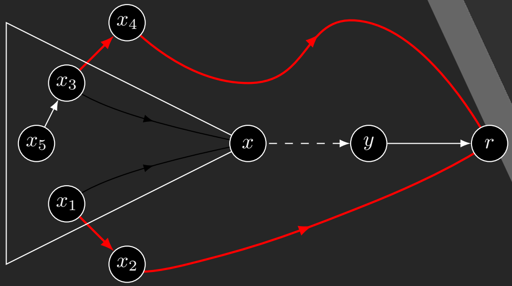
- Visits **traverse non–affected vertices**
- Pruning can stop visit only for **pairs of affected vertices**
- **VISIT** from x to y **CANNOT STOP** j (although x and j **are covered**)



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



How Much Time Do We Save via Scalable Algorithms

Dataset	Pruned Landmark Labeling				Hierarchical Hub Labeling [2]				Tree Decomposition [4]			BFS
	IT	IS	QT	LN	IT	IS	QT	LN	IT	IS	QT	
Gnutella	54 s	209 MB	5.2 μs	644+16	245 s	380 MB	11 μs	1,275	209 s	68 MB	19 μs	3.2 ms
Epinions	1.7 s	32 MB	0.5 μs	33+16	495 s	93 MB	2.2 μs	256	128 s	42 MB	11 μs	7.4 ms
Slashdot	6.0 s	48 MB	0.8 μs	68+16	670 s	182 MB	3.9 μs	464	343 s	83 MB	12 μs	12 ms
NotreDame	4.5 s	138 MB	0.5 μs	34+16	10,256 s	64 MB	0.4 μs	41	243 s	120 MB	39 μs	17 ms
WikiTalk	61 s	1.0 GB	0.6 μs	34+16	DNF	-	-	-	2,459 s	416 MB	1.8 μs	197 ms
Skitter	359 s	2.7 GB	2.3 μs	123+64	DNF	-	-	-	DNF	-	-	190 ms
Indo	173 s	2.3 GB	1.6 μs	133+64	DNF	-	-	-	DNF	-	-	150 ms
MetroSec	108 s	2.5 GB	0.7 μs	19+64	DNF	-	-	-	DNF	-	-	150 ms
Flickr	866 s	4.0 GB	2.6 μs	247+64	DNF	-	-	-	DNF	-	-	361 ms
Hollywood	15,164 s	12 GB	15.6 μs	2,098+64	DNF	-	-	-	DNF	-	-	1.2 s
Indochina	6,068 s	22 GB	4.1 μs	415+64	DNF	-	-	-	DNF	-	-	1.5 s

	label size		preprocessing [s]				space [MiB]				query [µs]			
instance	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

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

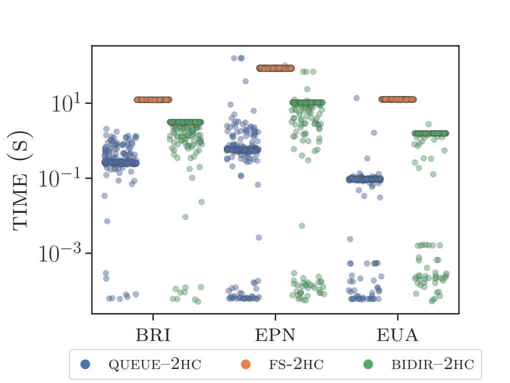
How Much Time Do We Save via Dynamic Algorithms

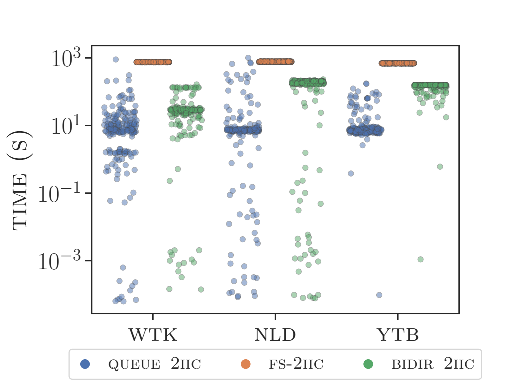
Dataset	Network Type	|V|	|E|	avg deg	S	D	W
Caida (cai)	Ethernet	3.20e+04	4.01e+04	2.51	●	●	○
Luxembourg (lux)	Road	3.06e+04	7.55e+04	4.11	●	○	○
wgtGnutella (gnu)	Peer2Peer	6.26e+04	1.48e+05	4.73	●	○	○
Brightkite (bkt)	Location-based	5.82e+04	2.14e+05	7.35	●	●	●
Efz (Efz)	Railway	1.25e+05	4.02e+05	6.43	●	○	○
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

How Much Time Do We Save via Dynamic Algorithms

SOME RESULTS: various networks

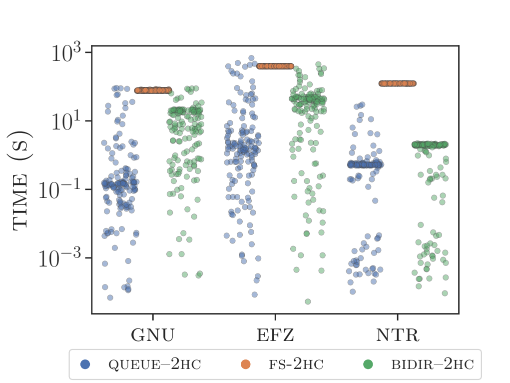


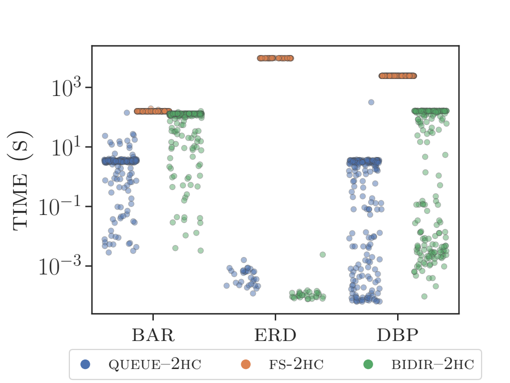


UNIVERSITÀ DEGLI STUDI DELL'AQUILA

How Much Time Do We Save via Dynamic Algorithms

SOME RESULTS: various networks





Future Work

FUTURE/ONGOING WORK:

- improve **known dynamic frameworks**
- design of **scalable graph algorithm** for other prominent graph mining problems
  - **TOP-K LOOPLESS SHORTEST PATHS PROBLEM**
  - **VERTEX SIMILARITY**
  - **ENUMERATION PROBLEMS**
- build/validate frameworks for **VERTEX/GRAPH SIMILARITY/CLASSIFICATION/RANKING** based on centrality measures and/or distances

[Abraham+: **Hub Labeling for Shortest Path Counting.** Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, 1813-1828]
[Akiba+: **Efficient top-k shortest-path distance queries on large networks by pruned landmark labeling.** Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 29. No. 1. 2015.]
[Al Zoobi+: **Finding the k Shortest Simple Paths: Time and Space trade-offs**. SEA 2020]

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

Outline

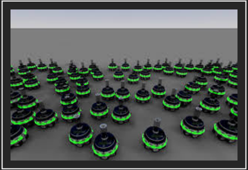
UNIVERSITÀ DEGLI STUDI DELL'AQUILA

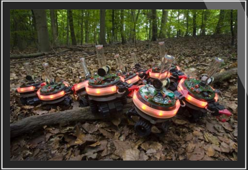
MULTI-ENTITY COMPUTING SYSTEM (MCS) distributed system of **AUTONOMOUS AND POSSIBLY MOBILE ENTITIES**

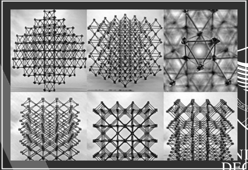
- equipped with **(few) computational resources**, some **perception**
- motion capabilities (physical or virtual), operating independently
- shared environment (e.g., Euclidean space or graphs)
- to accomplish some global (computational) task(s) (**actuators**)

SEVERAL REAL-WORLD TECHNOLOGIES are modeled as **MCSs**

- robotic swarms, flocks of UAVs, metamorphic robotic systems
- networks of software agents, web crawlers, viruses
- fleets of drones, programmable matter







UNIVERSITÀ DEGLI STUDI DELL'AQUILA

Active research field:

- In the last few years **considerably large amount of research** in the area of **distributed computing** devoted to study of **computational properties** and **algorithms** for this kind of systems
- **Reason:** high practical impact, interest driven by **real-world applications**
  - Exploration of Unknown/Dangerous Areas, Emergency Management, Search&Rescue
  - Process Automation, Monitoring/Surveillance
- **Combining** **computation** and **motion** introduces **several challenges** from computational perspective

Main objective of investigation:

- **Determine:** what computational tasks can be performed by the entities, under what conditions, and at what cost
- **Design algorithms** for the weakest possible entities to build **reliable**, **fault-tolerant**, **resistant to malicious behaviors** systems
- **Identify relationships** that, computationally speaking, exist among **different types of systems of mobile entities**

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

Why Massive Decentralized Systems of Weak Entities

MASSIVE DECENTRALIZATION MAIN ADVANTAGE cooperative behavior

- Tasks that require **many multiple entities made possible**
- Removing single point of failure, **no central control necessary**
- Cheap entities can be **replaced easily** without **breaking system**
- Moreover employing **CHEAP, WEAK ENTITIES** can **increase tolerant to disruptions/malicious behaviors**
- E.g. using entities **not requiring communication to achieve some goal** implies **SYSTEM ROBUST TO ANY MALICIOUS ATTACK ON COMMUNICATION CHANNELS**
- E.g. using entities **not requiring synchronization** means implies **SYSTEM ROBUST TO ANY MALFUNCTIONING IN SYNCHRONIZATION PROCESS**

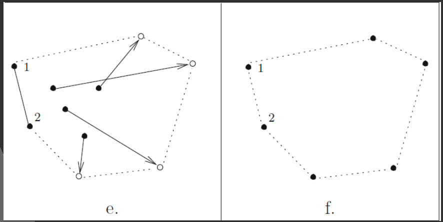
MASSIVE DECENTRALIZATION MAIN DISADVANTAGE: complex **algorithm design/analysis**

- Coordination **difficult to achieve**, under very weak entities it is **DIFFICULT TO DESIGN DISTRIBUTED ALGO-RITHMS** even for solving elementary tasks
- Convergence/Computation on **global properties/knowledge** might be **very slow or even infeasible**

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

Example of Task and Results: Pattern Formation

ARBITRARY PATTERN FORMATION (APF)	
Input:	A set of entities endowed with **multiplicity detection**, each one initially placed on a different vertex/point of an input **graph/environment**
Solution:	Find a distributed algorithm that ensures entities form any arbitrary pattern they are given in input, starting from any arbitrary initial configuration where entities occupy distinct location



Theorem. There exists a **deterministic transition-safe algorithm** that solves if and only the Leader Election problem can be solved in the initial configuration R, that is, R is a leader configuration[2].

S. Cicerone, G. Di Stefano, A. Navarra: **Asynchronous Arbitrary Pattern Formation: the effects of a rigorous approach.** Distributed Computing 32, 91–132 (2019)

[2]Defined wrt geometric properties

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

Example of Task and Results: Gathering

GATHERING	
Input:	A set of entities endowed with **multiplicity detection**, each one initially placed on a different vertex/point of an input **graph/environment**
Solution:	Find a distributed algorithm that ensures all entities to **REACH THE SAME VERTEX/POINT** from where they do not move anymore

Theorem. In absence of multiplicity detection and of any agreement on the coordinate systems, Gathering is deterministically unsolvable under semi-synchronization for anonymous uniform entities.

Prencipe, G.: **Impossibility of gathering by a set of autonomous mobile robots.** Theor. Comput. Sci. 384(2–3), 222–231 (2007)

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

In General: Characterization Results are Desirable

All such results led to **WIDER INVESTIGATION** to provide **GENERAL CHARACTERIZATIONS OF COMPUTATIONAL POWER** for multi-entity computing systems **sharing some set of features/capabilities**, under different assumptions:

- E.g. **VARIETY OF COMBINATIONS OF CAPABILITIES** (visibility, synchronicity, uniformity, being anonymous, communication, etc)
- E.g. moving on **graphs** rather than **Euclidean plane** or **3D environments**
- What **a system can and cannot do**, if it is made of entities that have certain characteristics

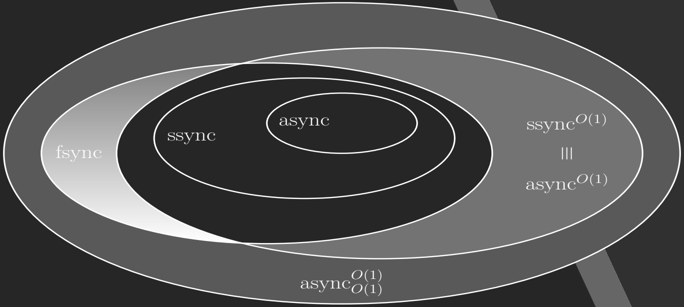
[M. D'Emidio, G. Di Stefano, D. Frigioni, A. Navarra. **Characterizing the computational power of mobile robots on graphs and implications for the Euclidean plane.** Information & Computation 263: 57-74 (2018)]

[K. Buchin, P. Flocchini, I. Kostitsyna, T. Peters, N. Santoro, K. Wada: **Autonomous Mobile Robots: Refining the Computational Landscape.** IPDPS Workshops 2021: 576-585]

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

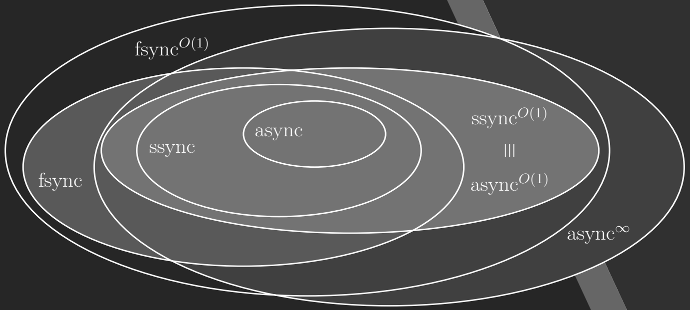
Example of Characterization of Computational Power

EXAMPLES OF RELATIONS BETWEEN COMPUTATIONAL MODELS: sets are **computable tasks**, entities are displaced on **EUCLIDEAN PLANE**: systems of entities that enjoys **full synchronicity** are **MORE POWERFUL** (can solve successfully more tasks) than those who do not (easy to show, other relations less trivial).



[M. D'Emidio, G. Di Stefano, D. Frigioni, A. Navarra. **Characterizing the computational power of mobile robots on graphs and implications for the Euclidean plane.** Inform. & Computation 263: 57-74 (2018)]

EXAMPLES OF RELATIONS BETWEEN COMPUTATIONAL MODELS: sets are **computable tasks**, entities are displaced on **GRAPHS (DISCRETE) ENVIRONMENTS**: systems where entities have **few bits of visible communication and act asynchronously** are **INCOMPARABLE** wrt fully synchronous systems (some tasks cannot be solved in one case, some others in the other)



[M. D'Emidio, G. Di Stefano, D. Frigioni, A. Navarra. **Characterizing the computational power of mobile robots on graphs and implications for the Euclidean plane.** Inform. & Computation 263: 57-74 (2018)]

Programmable Matter

Programmable Matter (PM)

- **MATTER** with the ability to **change its physical properties** in a **programmable fashion**
- **PROPERTIES** such as *shape*, *orientation*, *optical/electrical characteristics*
- An example of MCS (a system made of **WEAK NANO-SCALE SELF-ORGANIZING COMPUTATIONAL ENTITIES** called **PARTICLES**
  - Particles can be **PROGRAMMED** and some **form of actuators** to interact with environment/other particles to **COLLECTIVELY** achieve global **tasks**

Common tasks:

- coating, shape formation, compaction
- reconfigurable, smart materials, self-repairing structures, minimally invasive surgery

Originally introduced in

T. Toffoli and N. Margolus. 1991. Programmable matter: Concepts and realization. Physica D: Nonlinear Phenomena 47, 1, 263-272. 1991.

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

Now several Research&Development projects

- e.g. `https://www.programmable-matter.com/` (several prominent partners)

Several research activities toward **effective modeling and algorithms**, necessary for **deployment of PM systems** in real-world scenarios

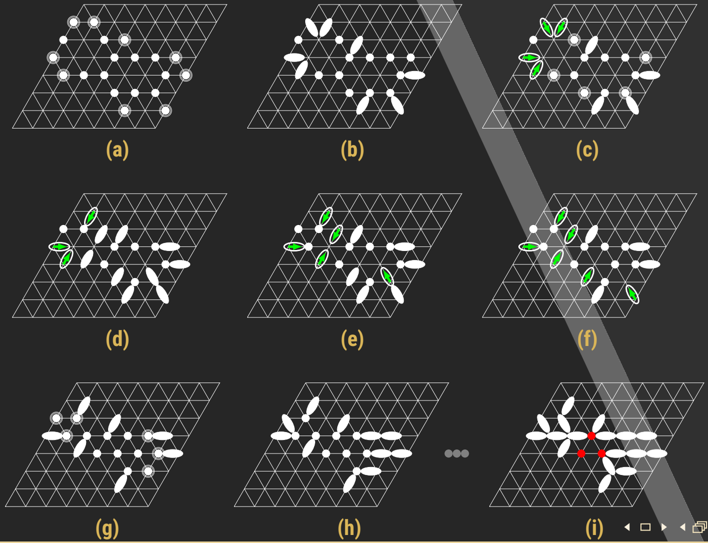
Some recent results

- various **distributed algorithms** for shape formation, coating, leader election with explicit communication
- algorithm for **particle leader election (PLE)** problems without use of inter-particle communication
- **impossibility** of achieving PLE cannot be solved without disconnecting the set of particles. Result holds even if the particles are endowed with unlimited memory and chirality
- **distributed algorithms** for **compaction** when particles can detect interiors of initial displacement (through sensors)

[G. D'Angelo, M. D'Emidio, S. Das, A. Navarra, G. Prencipe. **Leader Election and Compaction for Asynchronous Silent Programmable Matter.** Proceedings of 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '20), 276-284 (2020)]
[G.A. Di Luna, P. Flocchini, N. Santoro, G. Viglietta, Y. Yamauchi: Shape formation by programmable particles. Distributed Comput. 33(1): 69-101 (2020)]

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

HOW TO REMOVE EXPLICIT COMMUNICATION: algorithms can use **PARTICLES' STATES** (**expanded or contracted**) to encode information for **PROBLEM-SOLVING PURPOSES** (e.g. synchronization)



UNIVERSITÀ DEGLI STUDI DELL'AQUILA

Future Work

PM Systems

- **MORE REALISTIC MODELS**, according to technological advancements (**COMMUNICATION** vs **PERCEPTION**)
- algorithms for **MORE COMPLEX TASKS** (e.g. **SILENT SHAPE FORMATION**)
- investigating which tasks, besides leader election, **can** be successfully **performed** under no-communication and **which cannot**
- in case of impossibility results, determine capabilities one must add to PM systems to achieve **feasibility**
- on field experimentation/dedicated discrete events simulations environments

MCSs

- Dealing with **UNCERTAINTY/DYNAMIC ENVIRONMENTS**
- Solutions for **METAMORPHIC ROBOTIC SYSTEMS**

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

Thanks for your attention

Q&A

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

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

Future Work for Other Projects

- Design, Implementation, Testing of Technologies and Algorithms for Processing Massive Scientific Datasets via Cloud Computing Services
- Engineering, Implementation and Experimental Evaluation of Solvers for Computational Geometry problems
- Implementation and Experimental Evaluation of Algorithms for various vehicle-routing problems via Google OR-tools (OR-Tools is fast and portable software for combinatorial optimization)

UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA