PERFORMANCE MODELING OF GRAPH PROCESSING WORKLOADS

Merijn Verstraaten[1,2], Ana Lucia Varbanescu[1,3]

A.L.Varbanescu@utwente.nl

1 UNIVERSITY OF AMSTERDAM

2 netherlands eScience center

3 UNIVERSITY OF TWENTE.

Graph processing ...

… is / can be / will be everywhere![1,2]

- Bioinformatics
- Pandemic analysis[3]
- Social networks analysis
- Fraud detection
- Neural networks
- …

5+ sessions in this conference alone!

[1] Sherif Sakr et al.
"The Future Is Big Graphs: A Community View on Graph Processing Systems" – CACM Sept. 2021
[2] Tim Hegeman, Alexandru Iosup
"Survey of Grpah Analysis Applications" - arXiv:1807.00382
[3] https://neo4j.com/graphs4good/covid-19/

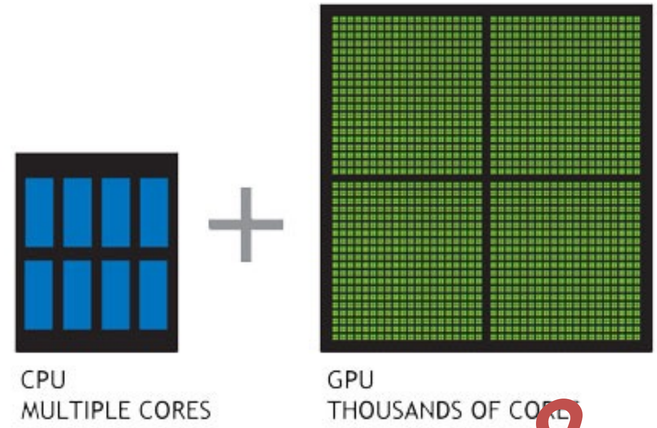
Large Scale Graph Processing

- Graph processing is (very) data-intensive
  - 10x larger graph => 100x or 1000x slower processing
- Graph processing becomes (more) compute-intensive
  - More complex queries => ?x slower processing
- Graph processing is (very) dataset-dependent
  - Unfriendly graphs => ?x slower processing

We use parallel processing & architectures to enable *more complex analytics* on *larger graphs.*

HPC Platforms

- Multi/many-core systems
  - Massive (data) parallelism
  - Built for high throughput processing
  - Penalties for branches
  - Penalties for load imbalance



(mis)match?

- Graph processing [4]

Parallelism <=> Increased performance

  - Poor data locality

More parallelism <=> Increased performance variability!

[4] Andrew Lumsdaine et al.
"Challenges in Parallel Graph Processing" – Parallel Processing Letters 2007

Today’s headlines

1. ~~Motivation~~
2. Variability analysis

   Case-studies: PageRank and BFS
3. Performance modeling

   Analytical modeling vs. Data-driven/ML modeling
4. Take home message

2. Variability analysis

All experiments …

- NVIDIA TitanX + CUDA 10.0
- Results presented on 9 graphs

Id	Graph	# Vertices	# Edges	Dataset
1	actor-collaboration	382,219	30,076,200	KONECT
2	amazon0601	403,394	3,387,390	KONECT
3	flixster	2,523,390	15,837,600	KONECT
4	jester1	73,512	8,272,720	KONECT
5	patentcite	3,774,770	16,518,900	KONECT
6	wikipedia_link_en	12,151,000	378,142,000	KONECT
7	wiki_talk_ru	457,017	919,790	KONECT
8	higgs-social_network	456,626	14,855,800	SNAP
9	sx-stackoverflow-c2q	1,655,350	11,226,800	SNAP

PageRank calculation

- Calculates the PR value for all vertices
  - Assign value to each vertex
  - Repeat until convergence
    - Collect PR for all incoming edges
    - Update vertex PR

We use 7 versions + 2 warp-parallelism parameterized ones

- Challenges
  - No computation
  - Load-balancing
  - Irregular memory accesses

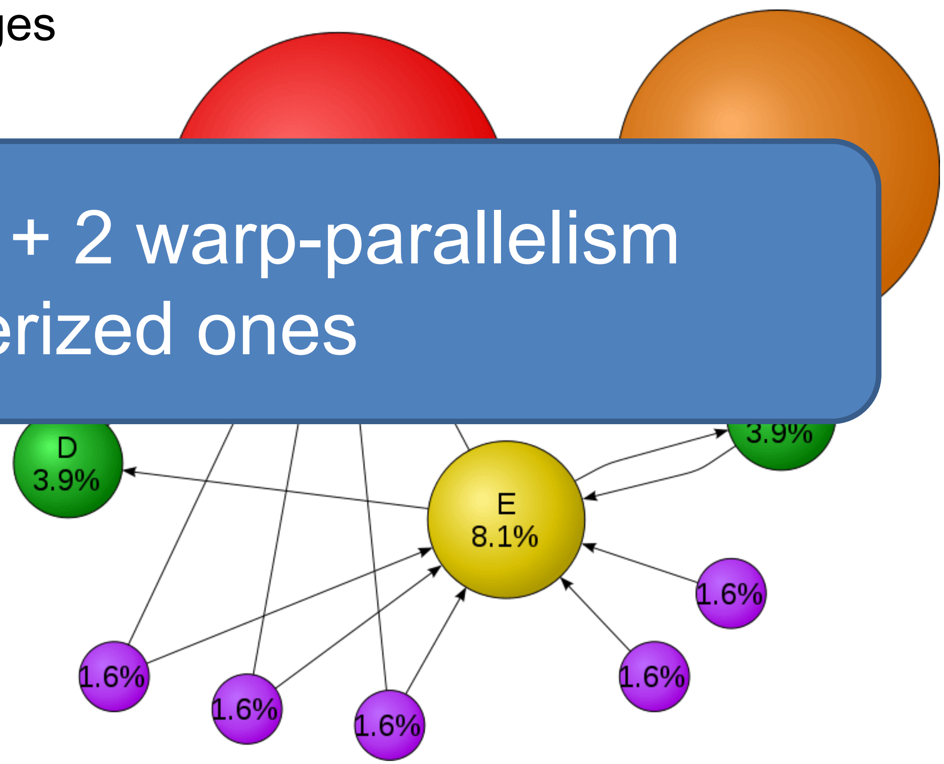
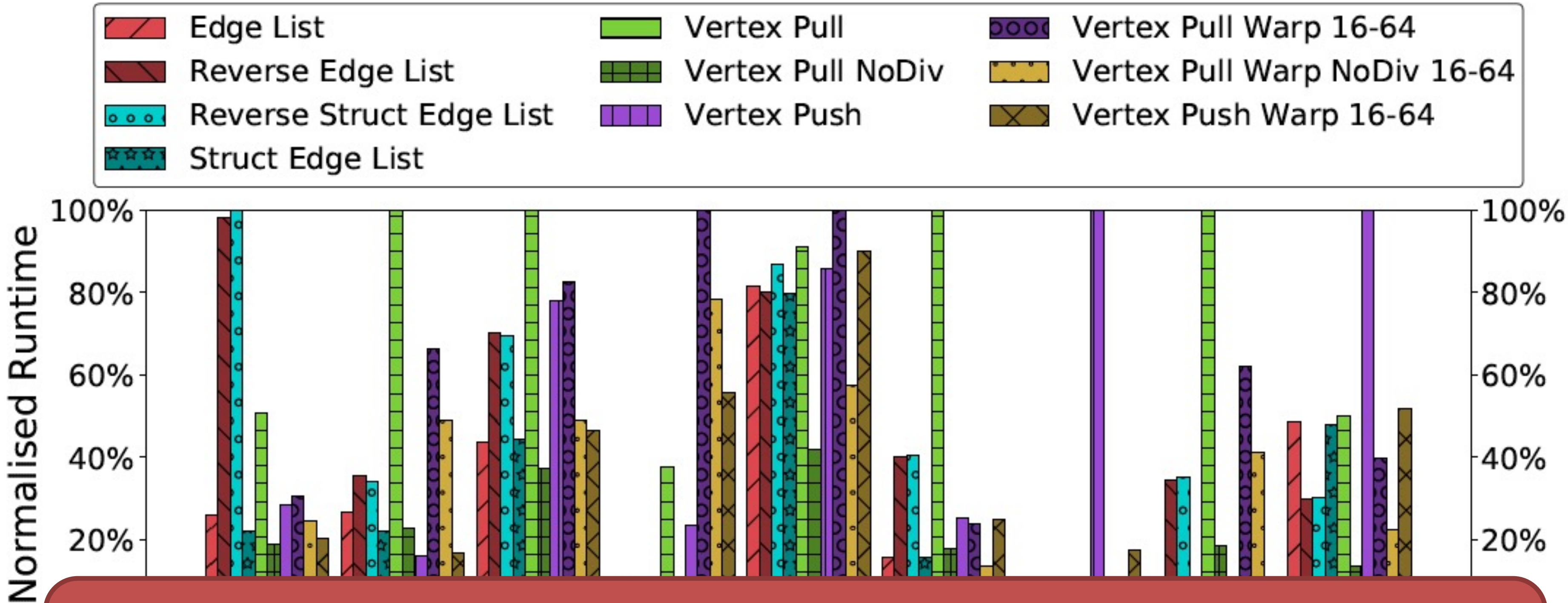


Image courtesy of: https://en.wikipedia.org/wiki/PageRank

PageRank: results



- Different algorithms behave best.
- Different algorithms behave worst.
- The gap in execution time can be up to 2 orders of magnitude.

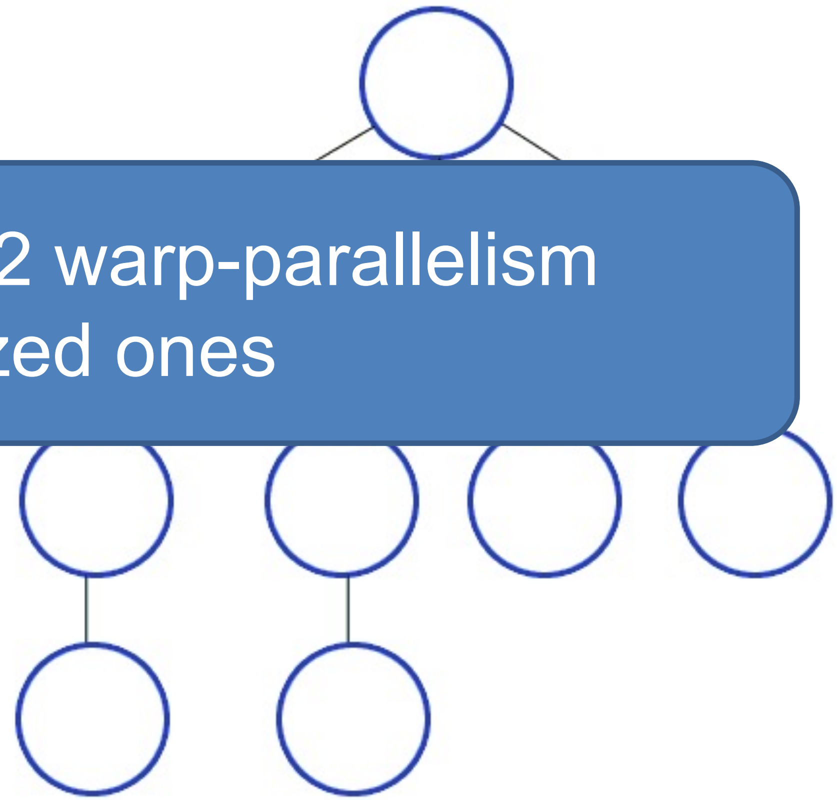
Choosing the wrong algorithm can really make a difference!

BFS traversal

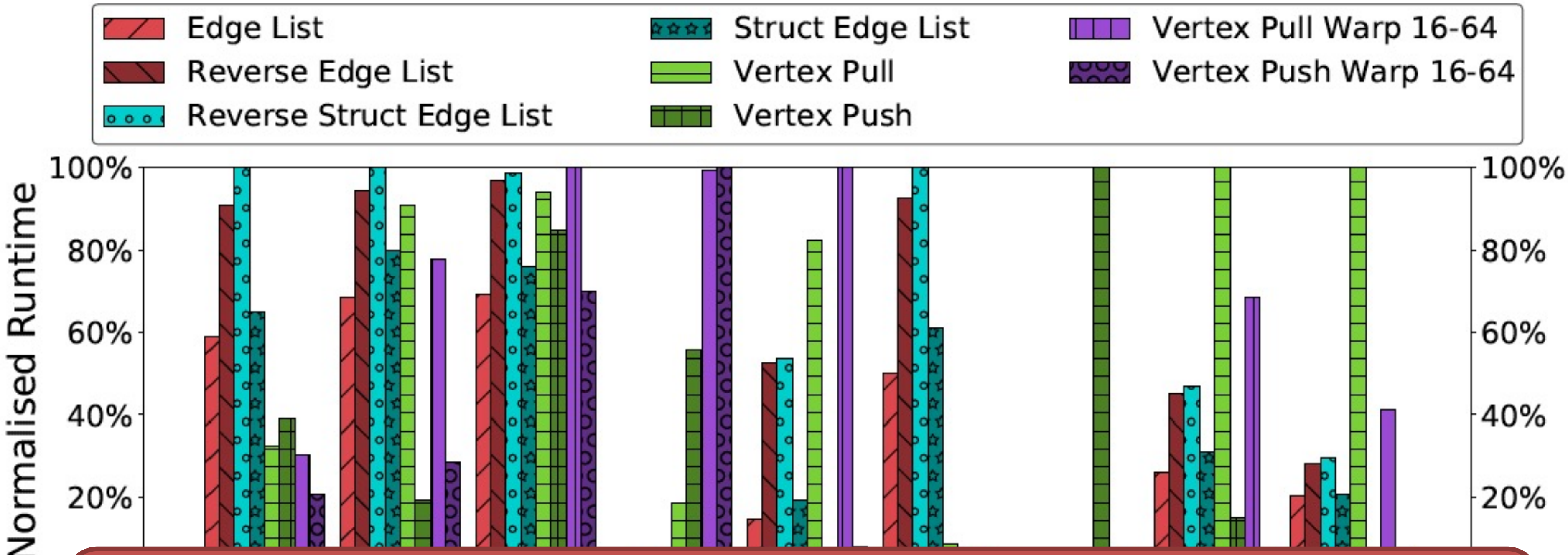
- Traverses the graph layer by layer
  - Starting from a given node
- Sensitive to …
  - High diameter
  - Graph density

We use 6 versions + 2 warp-parallelism parameterized ones

  - No computation
  - Load-balancing
  - Irregular memory accesses



BFS: results



- Different algorithms behave best.
- Different algorithms behave worst.
- The gap in execution time can be up to 2 orders of magnitude.

Choosing the right / wrong algorithm can really make a difference!

3. Modeling

Choose the best algorithm

- Model the algorithm
  - Basic analytical model (work & span)
- Calibrate to platform
  - GPU, CPU, …
- Model the dataset
  - Size, dimension, topology …

T = f(P, A, D)

- Predict performance
  - Plug the platform and graph parameters into algorithm model

- Rank solutions and pick best.

PageRank: Analytical models

- Different algorithms => different models

$$T_{\text{edge}} = (7 * |E| * T_{\text{read}} + |E| * T_{\text{atom}})$$

$$T_{\text{push}} = (6 * |V| * T_{\text{read}} + |E| * T_{\text{read}} + |E| * T_{\text{atom}})$$

$$T_{\text{pull}} = (5 * |V| * T_{\text{read}} + 3 * |E| * T_{\text{read}} + |V| * T_{\text{write}})$$

$$\begin{aligned} T_{\text{NoDiv}} &= (5 * |V| * T_{\text{read}} + 2 * |E| * T_{\text{read}} + |V| * T_{\text{write}}) \\ &\quad + (3 * |V| * T_{\text{read}} + 2 * |V| * T_{\text{write}} + \frac{|V|}{32} * T_{\text{atom}}) \\ &= (8 * |V| + 2 * |E|) * T_{\text{read}} + 3 * |V| * T_{\text{write}} + \frac{|V|}{32} * T_{\text{atom}} \end{aligned}$$

- Calibrate for the platform : T_{read}, T_{write}, T_{atom} …
- Use dataset features: |E| and |V| from the graph specs

PageRank: poor model accuracy!

- Work-models are correct
  - We capture correctly the number of operations
- Model calibration has failed
  - Workload imbalance between threads within a warp
  - Non-uniform memory access times due to coalescing, caching, and atomic contention.
- Can we do any better?
  - Tried modelling parallelism => too complex
  - Tried performance counters => still not “stable” enough

Choose the best algorithm

- Model the algorithm
  - Basic analytical model (work & span)
- Calibrate to platform
  - GPU, CPU, …
- Model the dataset
  - Size, dimension, topology …

T = f(P, A, D)

- Predict performance
  - Plug the platform and graph parameters into algorithm model

Only 50% accuracy ☹

- Rank solutions and pick best.

The models

Long list of trials … with various ratios failure/success

- Analytical model
  - Predict execution time
    - Able to predict work accurately
    - Unable to accurately calibrate it
  - Predict ranking
    - Use relative cost of operations
    - Still unable to accurately calibrate it
- Data-driven models (machine learning)
  - Predict execution time
    - Use random forest
      - Based on hardware counters (previous work)
      - Based on graph features
  - Predict ranking
    - Use decision trees
      - Based on graph features

Low accuracy.

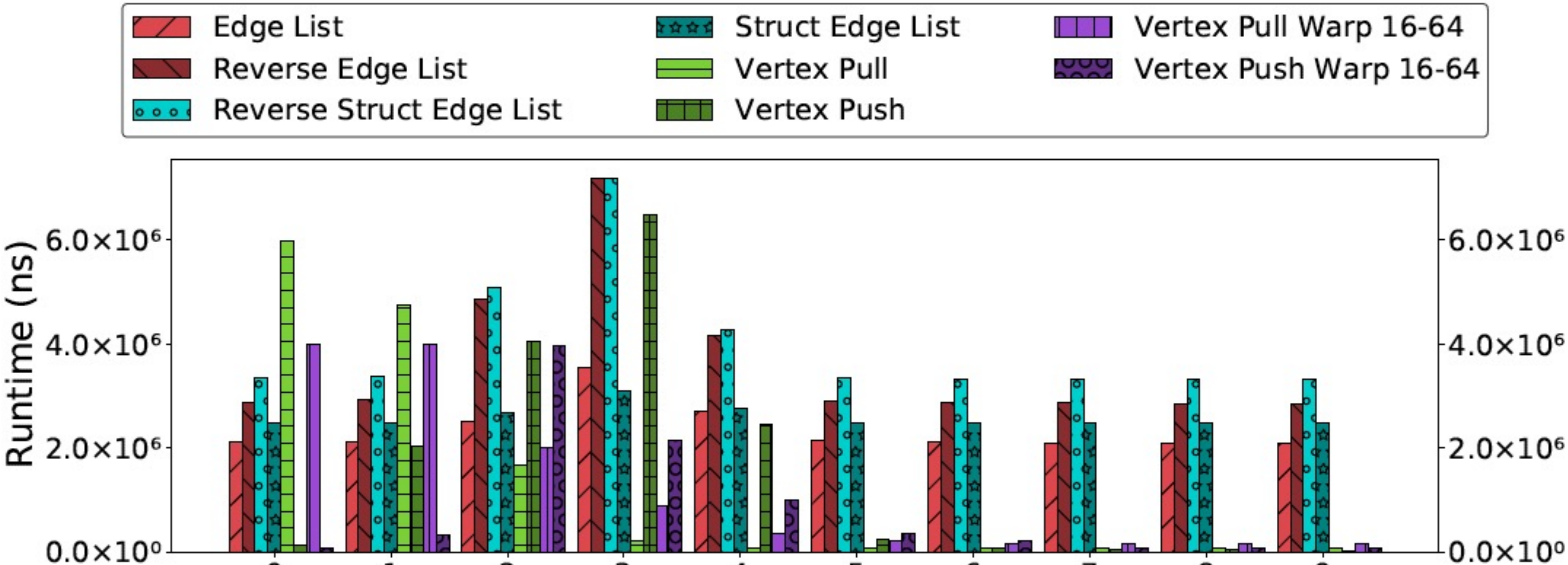
Still low accuracy

OK accuracy,
High prediction cost

High accuracy,
Low prediction cost

Still not working for BFS!!!

BFS: best algorithm changes!

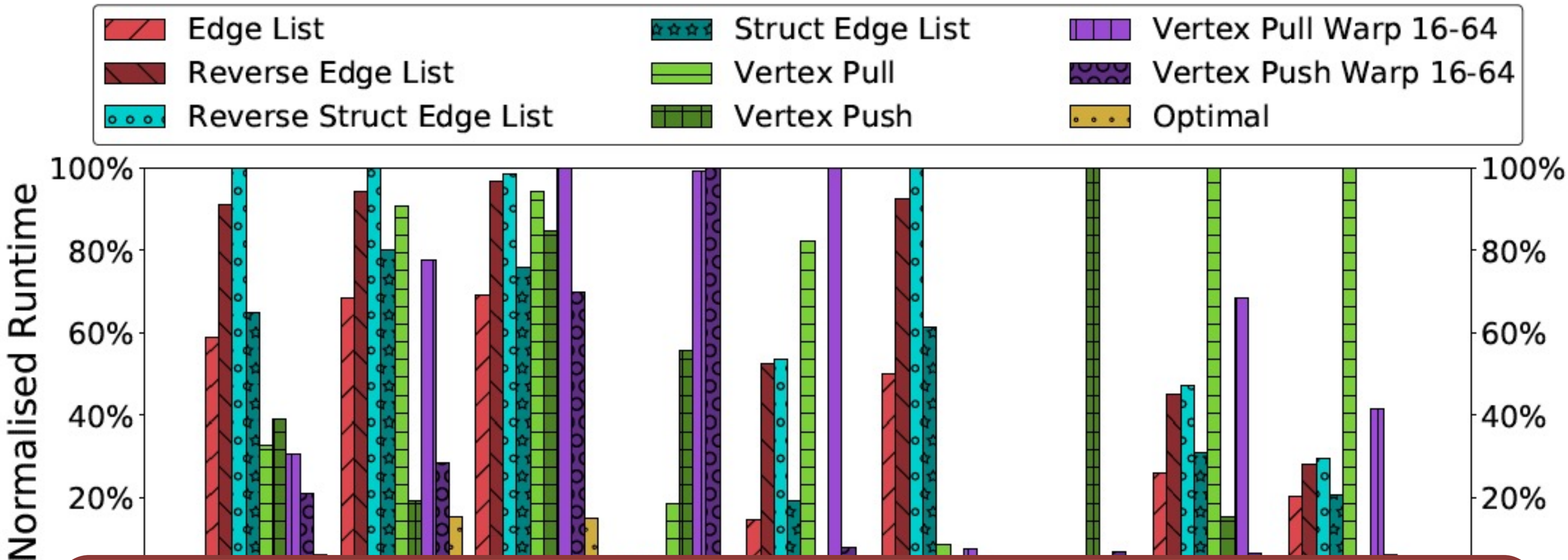


- Best algorithm changes per level
- Gaps are even larger than for the full scale
- We have more data for every level

We must predict at every level, NOT at the full graph level !

Results on the actor-collaborations graph - KONECT

BFS: *construct* the best algorithm!



- Optimal algorithm is the sum of the best per-level algorithms.
- Must switch implementations

If we predict best algorithm per level => we construct the best algorithm

BFS: *construct* the best algorithm!

- Predict ranking
  - Determine the **best algorithm per level**
  - **Still** depends on platform and dataset …

- **Construct** the best overall algorithm
  - Best algorithm per layer => best overall *by construction*
  - Switching between algorithms is a challenge
    - When?
    - How?

Mix-and-match: build the best algorithm at run-time by **switching to the best implementation** at every level*

*this is a generalization of the direction-switching BFS

Predicting ranking per level

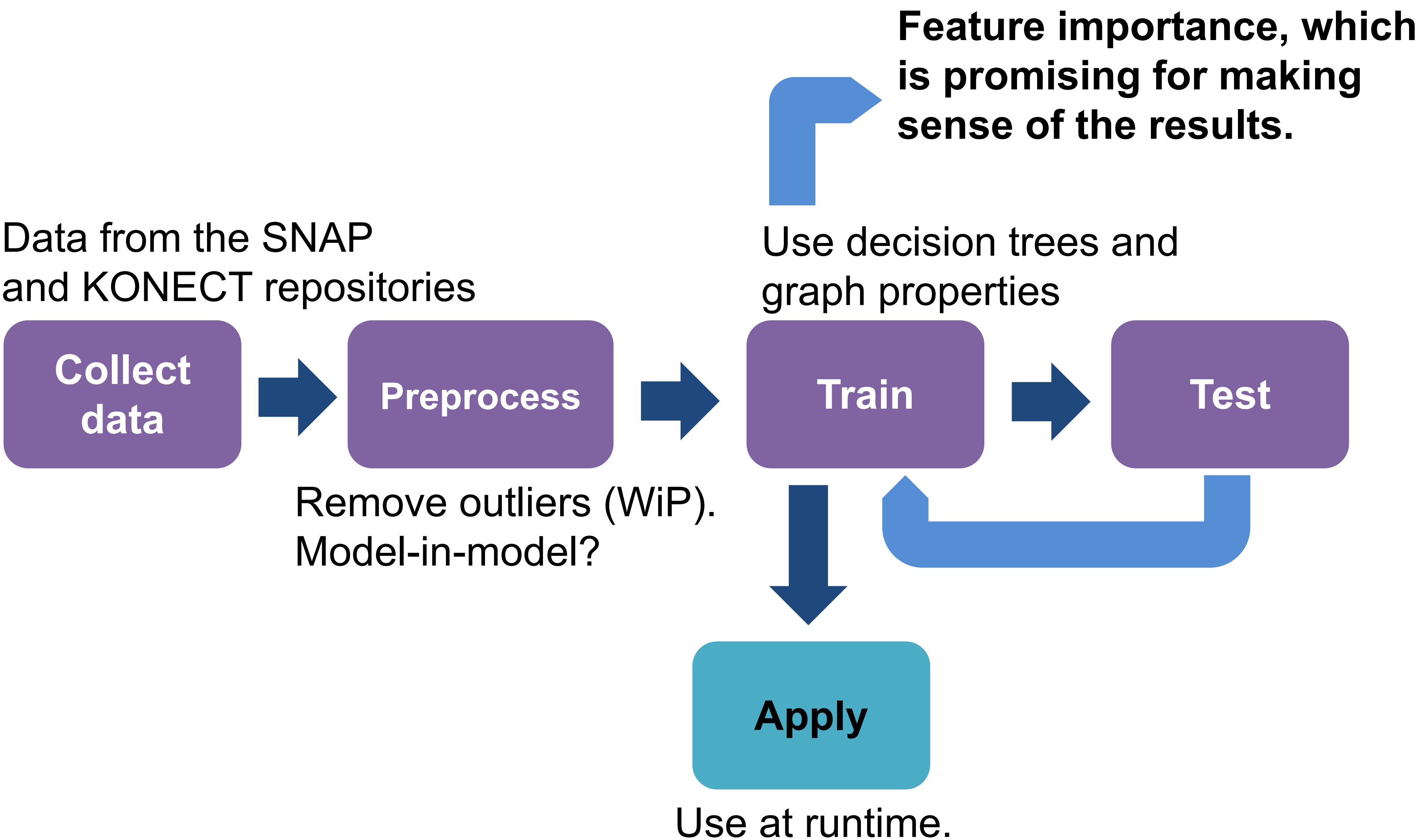
- Based on decision trees
  - Small number of samples
  - Fairly easy to train
  - Model is fast to use at runtime

Average prediction time: 144ns
Min BFS step: 20ms

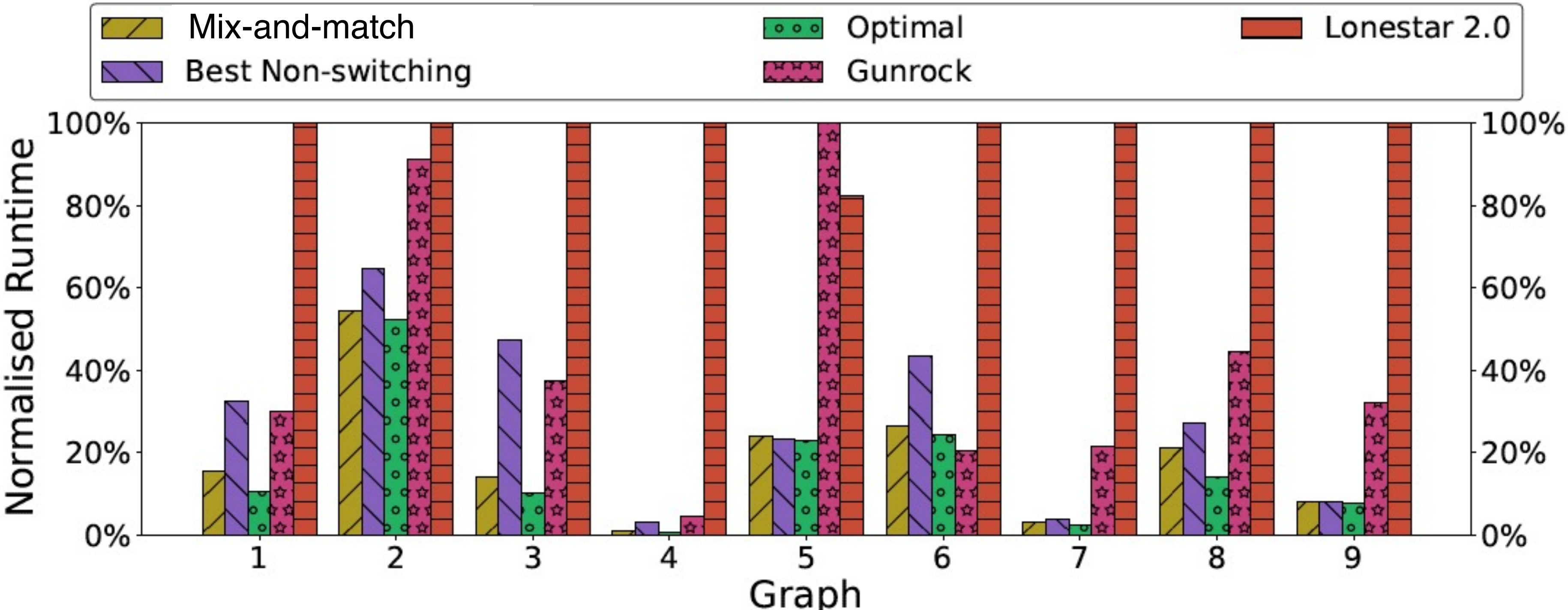
- Training parameters: **graph features** and **best algorithm**
  - Degree distribution (5 number summary and standard deviation)
  - Frontier size
  - Percentage discovered
  - Vertex count
  - Edge count
  - Ranking

Dataset: 248 graphs x ~11 root nodes
Accuracy: ~98%

Current workflow



Does it really work?

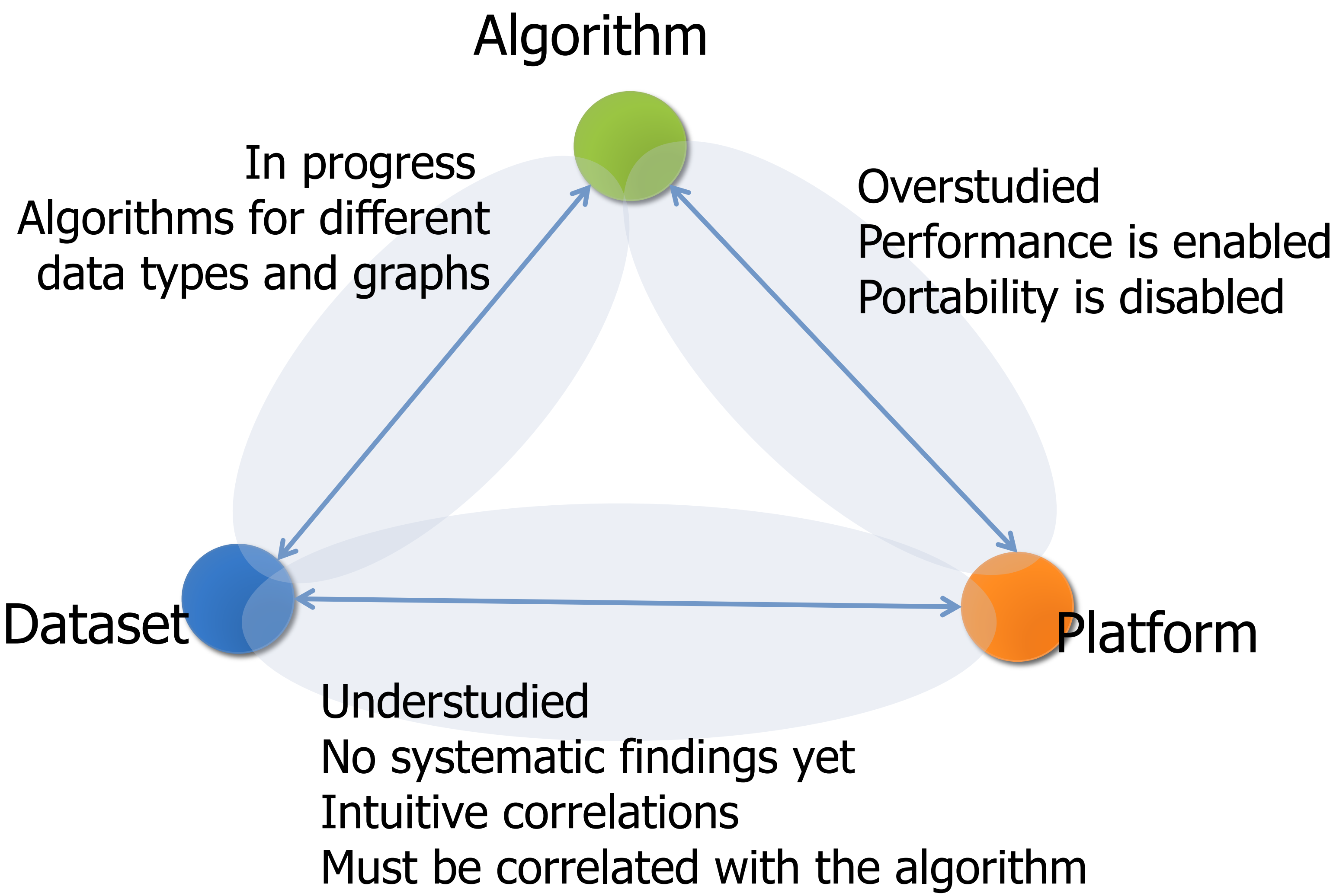


- Runtime switching is possible, (currently) with some memory overhead
- We are faster than the state-of-the art, on average, by 3x

Mix-and-match uses performance variability to build the best BFS per graph!

4. Take home message

P-A-D triangle



Take home message

- Main challenges in performance modeling
  - Performance depends on platform, algorithm, and dataset.
  - No modeling strategies exist for datasets
  - Analytical workload models are difficult to calibrate
  - Iterative algorithms (like BFS) require prediction per iteration
  - Statistical models require good features selection

- Current status
  - No accurate analytical performance models
  - Per-level , per-algorithm *ranking* prediction works well
  - Selection of "best-algorithm"
    - Possible for PageRank
    - Impossible for BFS => construct it with Mix-and-Match
      - Mix-and-Match outperforms state-of-the-art.

Take home message

- Mix-and-match enables dynamic, runtime switching among different versions of BFS
  - A generalization of the direction-optimized BFS
  - Machine learning model used to guide the switching
    - We use decision-trees as they offer a good accuracy-applicability trade-off

- More to follow
  - More algorithms & graphs & plaforms
  - Reason about features impact per algorithm

Authors:
Ana: A.L.Varbanescu@utwente.nl
Merijn: merijn@inconsistent.nl
Code: https://github.com/merijn/Belewitte