Speeding Up GPU Graph Processing Using Structural Graph Properties

Merijn Verstraaten, Ana Lucia Varbanescu & Cees de Laat

University of Amsterdam

The Problem: We want the fastest graph processing!

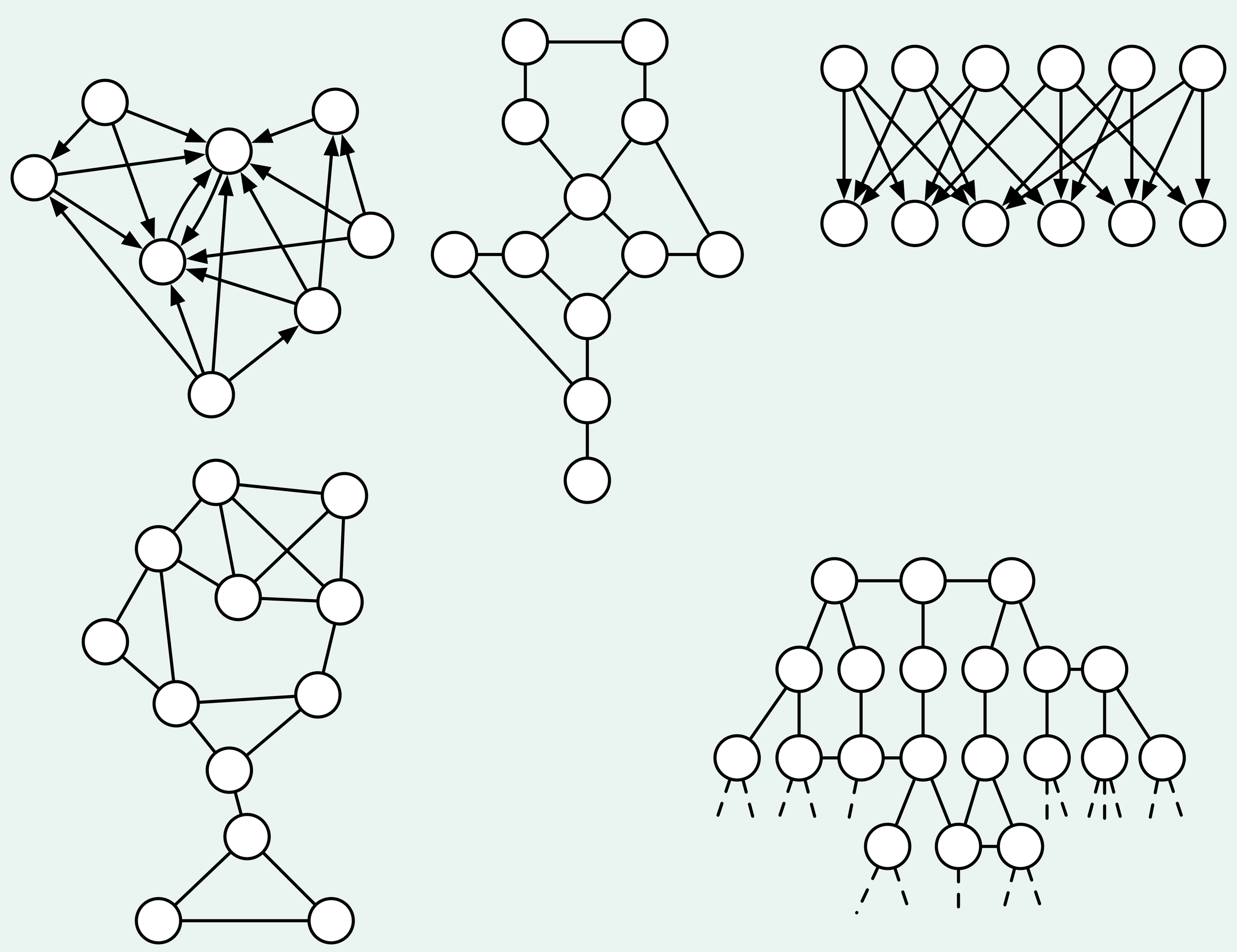
- High-performance graph processing is very interesting for data science
- High-performance computing is increasingly GPU/accelerator based
- Mapping irregular (graph) algorithms to GPU is hard
- Performance of irregular algorithms is data-dependent

Thesis Goals

- Quantify performance impact of data dependence
- Model how performance relates to structural properties of the input graph
- Predict best parallelisation strategy for a given graph and algorithm
- Create an automated pipeline to repeat this work for new algorithms and parallelisation strategies

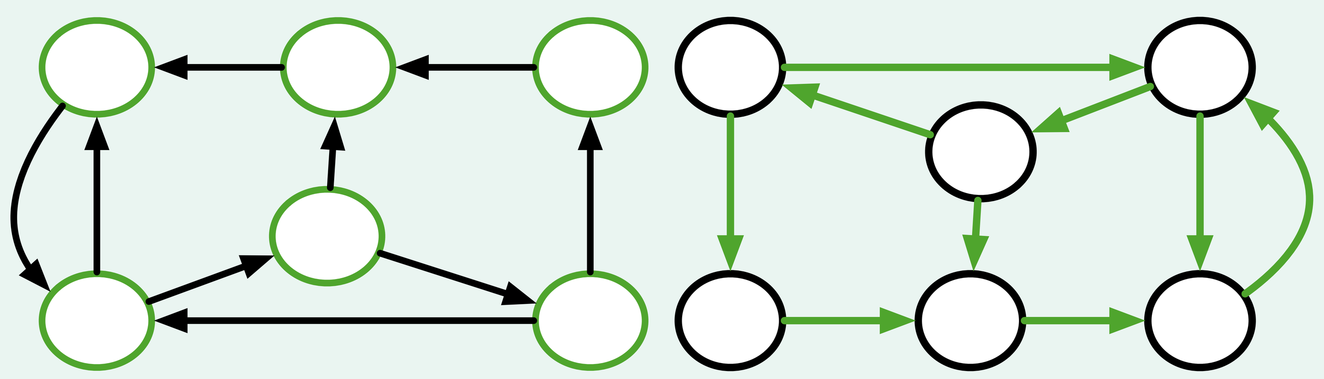
Structural Variation

We have graphs from social networks, road networks, biology. They are different in structure and properties.



Parallelisation Strategies

Vertex-centric push/pull, edge-centric, Gather-Apply-Scatter (GAS), virtual warps. Many possible variations of these, such as using warp and/or block reductions.



Vertex Push/Pull **Edge-centric**

parallel for v ∈ Vertices **do**
  f(v.neighbours)
endfor

parallel for e ∈ Edges **do**
  f(e.origin, e.destination)
endfor

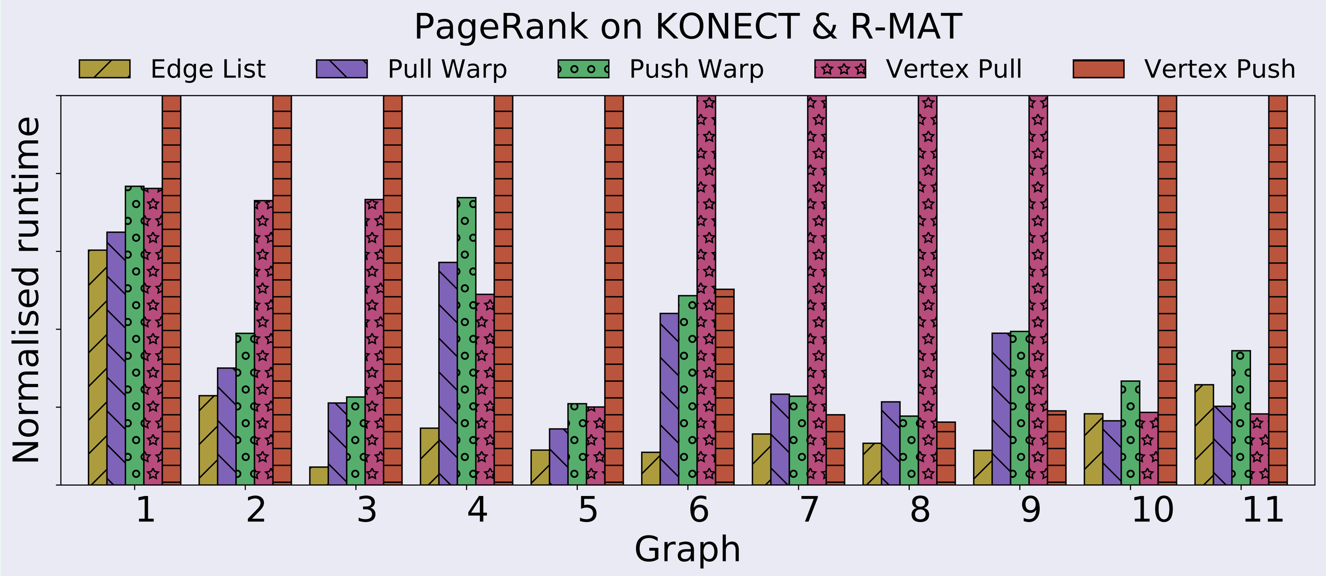
Bibliography

[1] J. Kunegis. Konect: The Koblenz Network Collection. In Proceedings of the 22nd International Conference on World Wide Web, WWW'13 Companion, pages 1343–1350, 2013.

[2] D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-MAT: A Recursive Model for Graph Mining. In SDM, volume 4, pages 442–446. SIAM, 2004.

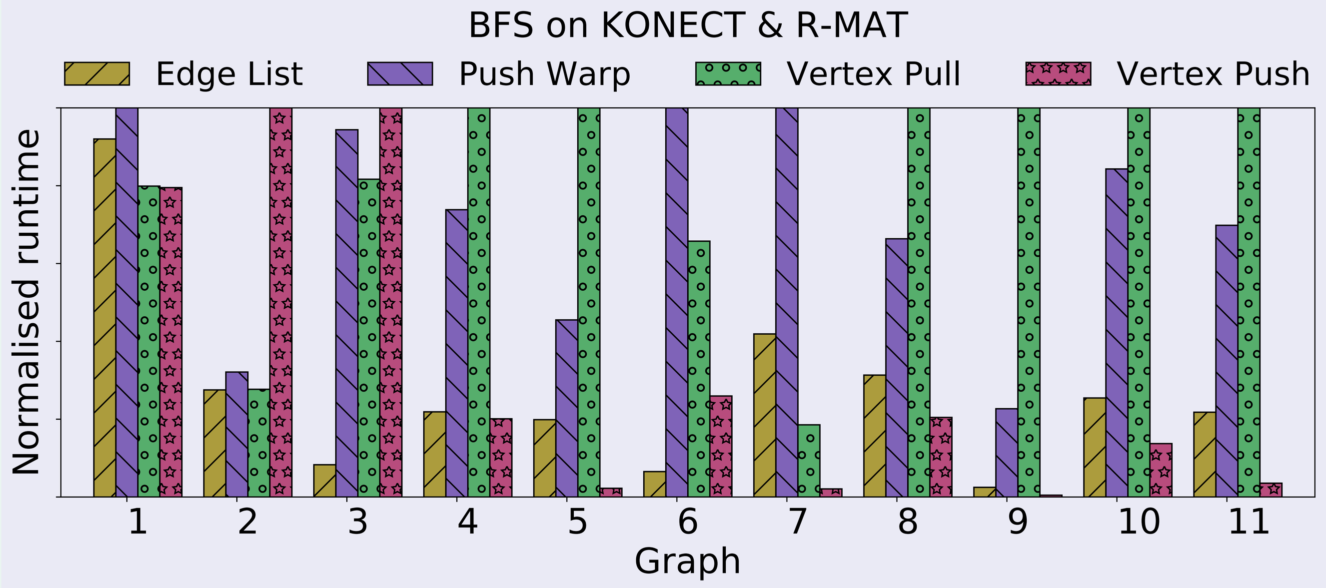
Performance Variation

The performance of different parallelisation strategies varies by an order of magnitude or more across graphs.



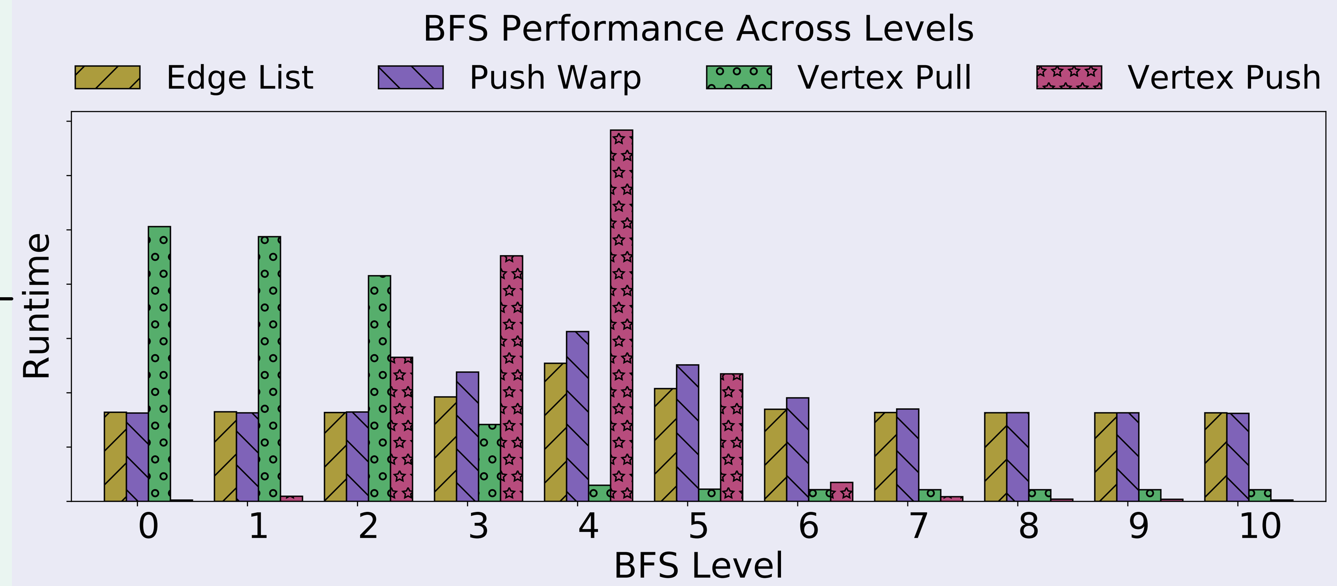
Dynamic Algorithms

For dynamic algorithms, where the relevant data changes over time, such as BFS, this effect is even stronger.



Variation Within a Single Run

For dynamic computations like BFS, we even see these huge performance differences between implementation across different steps computed on the same graph.



Graph Classification?

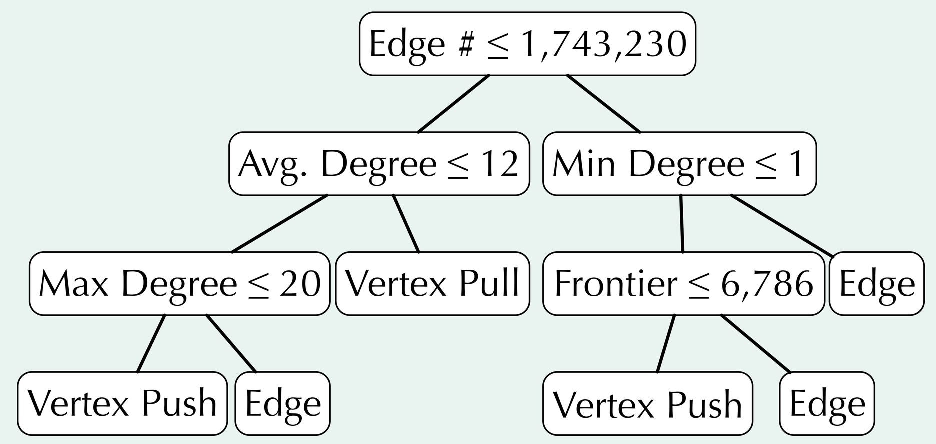
Graph structure affects performance for most algorithms, yet there is no consensus on any form of classification based on structural properties to aid implementation selection.

Performance Modelling

Can we learn to predict implementation performance from previously observed results?

	#V	#E	Avg. Degree	Max Degree	Standard Deviation Degree
1	382,219	31,076,166	79	3,956	163.3
2	28,093	6,296,894	224	4,909	315.1
3	2,025,594	10,604,552	5	93,257	113.4
4	1,899	20,296	21	339	35.6
5	89,269	3,330,225	75	6,515	139.4
6	325,729	1,497,134	9	10,721	48.4
7	12,150,976	378,142,420	62	963,032	606.4
8	3,023,165	102,382,410	68	337,969	556.9
9	1,984,484	14,869,484	15	61,572	137.2
10	8,870,942	260,379,520	59	406,416	631.3
11	17,062,472	523,602,831	61	639,143	740.3

Decision Trees



Prediction works!

Accuracy:	~98%
Avg. Evaluation:	144 ns (σ = 165 ns)
Min. BFS Step:	20 ms

Prediction Feasibility

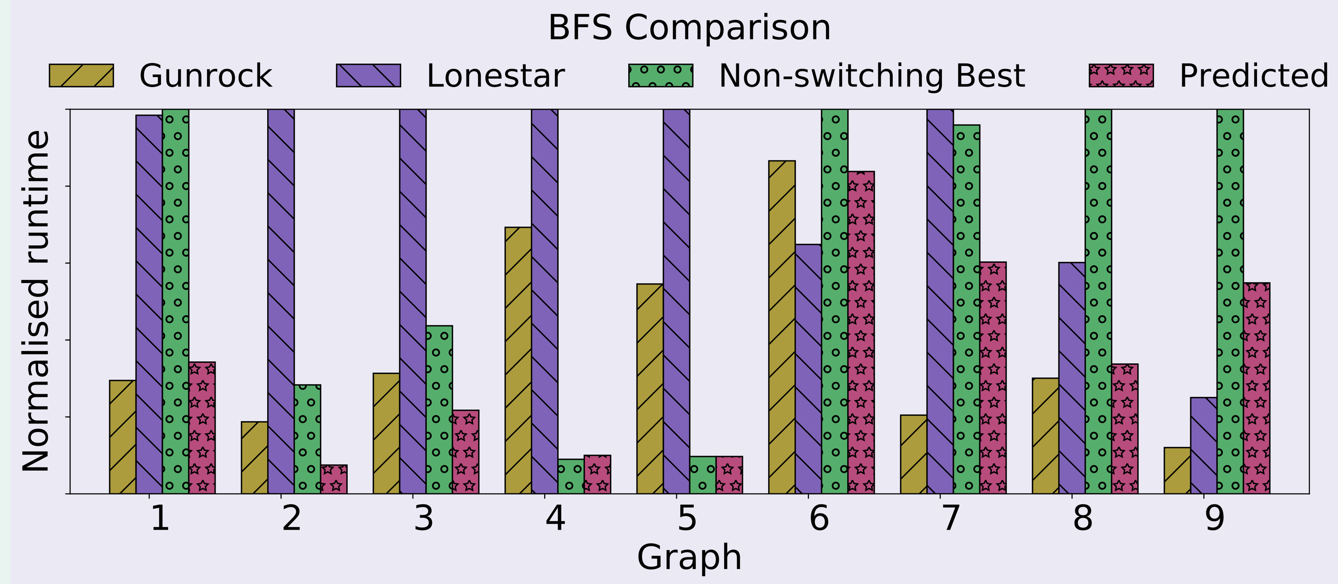
For simple algorithms we can use this model as an oracle to select the best performing implementation for a specific graph. For algorithms whose behaviour changes at runtime, like BFS, we can do better. We can keep multiple representations in memory and switch between implementations at runtime for a classic time-space trade-off.

BFS Prediction Results

Algorithm	Optimal	1–2×	>5×	Average	Worst
Predicted	56%	41%	1%	1.40×	236×
Oracle	23%	55%	2%	1.65×	9×
Edge list	10%	61%	7%	2.22×	38×
Vertex Pull	0%	15%	27%	38.62×	2,671×
Vertex Push	9%	15%	53%	39.66×	1,048×
Push Warp	0%	0%	3%	18.69×	97×

Results across all KONECT graphs.

The new BFS is fast!



In Summary

We show that using models trained on previously observed graph processing results lets us predict the best performing implementation of an algorithm for a given input graph.

We provide a framework for training such models and are investigating how much data is required to train an accurate and portable model for graph algorithms.

References

Varbanescu, A.L., Verstraaten, M., Penders, A., Sips, H., de Laat, C.: Can Portability Improve Performance? An Empirical Study of Parallel Graph Analytics. In: ICPE'15 (2015)

M. Verstraaten, A. L. Varbanescu, and C. de Laat. Quantifying the performance impact of graph structure on neighbour iteration strategies for pagerank. In Euro-Par 2015: Parallel Processing Workshops, pages 528–540. Springer, 2015.

M. Verstraaten, A. L. Varbanescu, and C. de Laat. Using Graph Properties to Speed-up GPU-based Graph Traversal: A Model-driven Approach. (Under Submission)

https://github.com/merijn/GPU-benchmarks