Which Graph Representation to Select for Static Graph-Algorithms on a CUDA-capable GPU

Thorsten Blaß and Michael Philippsen

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

COMPUTER SCIENCE DEPARTMENT
PROGRAMMING SYSTEMS GROUP
ERLANGEN • GERMANY

Motivation (I)

Your task: write static graph algorithm on the GPU.

Preparation: check literature

- several data structures to represent graphs.
- all papers pick CRS data structure
  and focus on optimizing algorithmic aspects.

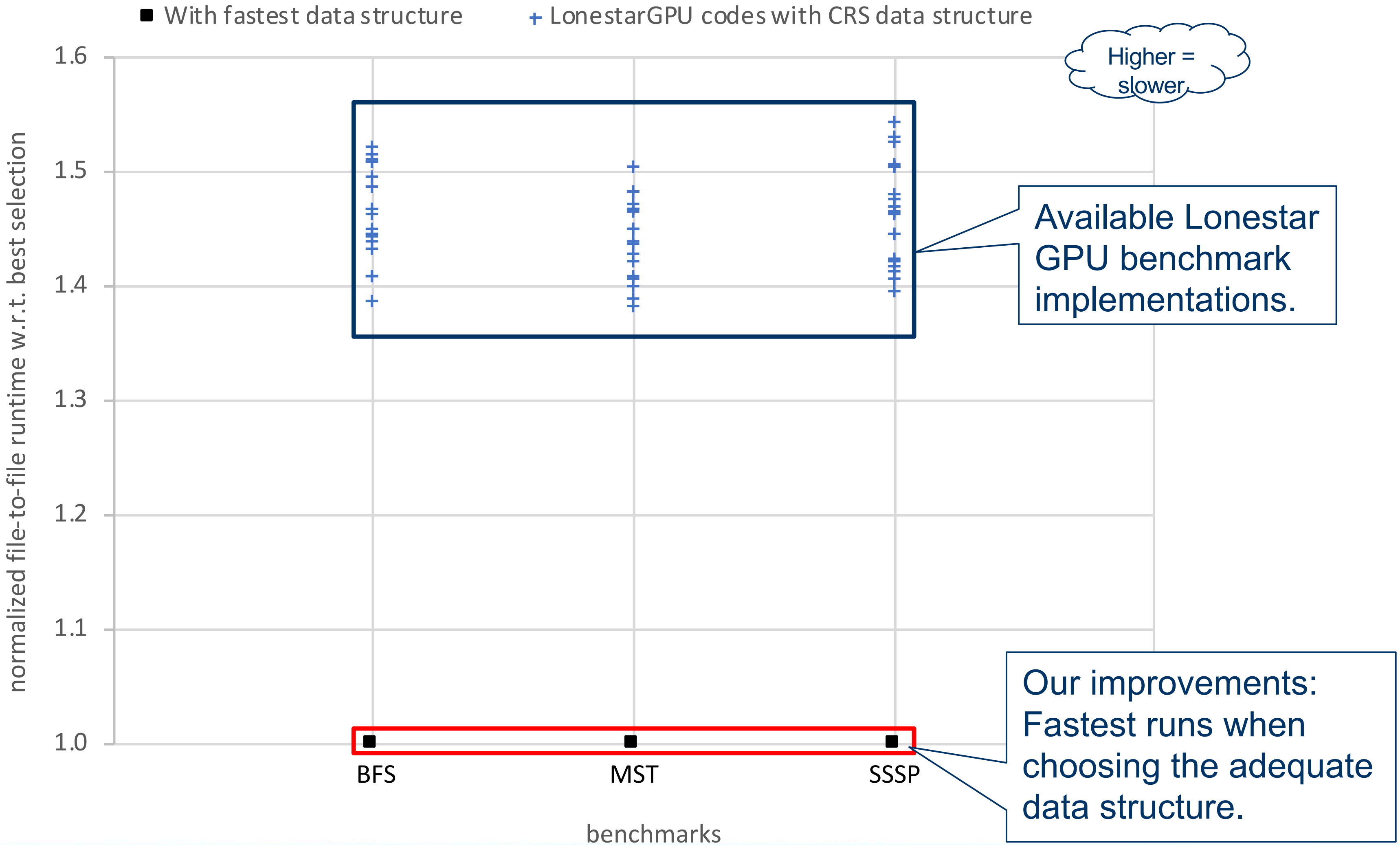
You wonder:

- Is data structure irrelevant for performance?
- Is it right to follow the crowed?

This paper: Graph representation matters!

Motivation (II)

- *Adequate* data structure speeds up graph algorithms.



Study Setup

- What is the *adequate* data structure?

- A total of 754,000 measurements:
  - 10 state-of-the-art static graph algorithms (do not modify the graph).
  - 19 input graphs.
  - 4 architecturally different Nvidia GPUs.
  - 10 graph data structures.
  - 3 widely used graph exchange formats.

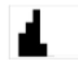
Study Setup – Benchmark Graph Algorithms

- 10 state-of-the-art implementations from recent research and benchmark suites with different characteristics.

	Thread per...	Granu-larity	Access Pattern	Termination	*weighted*	*directed*
APSP	edge	all	edge	worklist	✓	✓
BFS	node	subset	successors	recursion		
MIS	node	subset	random nodes successors	worklist		
MST	node	all	successors	recursion	✓	
PR-n	node	all	predecessors	fixpoint		✓
PR-e	edge	all	edge	fixpoint		✓
SpMV	node	subset	successors	single pass	✓	
SSSP	node	subset	successors	worklist	✓	✓
WCC-n	node	subset	successors	fixpoint	✓	
WCC-e	edge	all	edge	fixpoint	✓	

Study Setup – Input Graphs

- 19 input graphs with different characteristics.

Graph (original name)		avail. Format	Nodes Edges	Degrees avg. / max. excess (%)	Spyplot
USA-road.d-USA	network	*GR*	23.9M 116.6M	2.46 / 9 1.79	
USA-road-d.-CAL	network	*GR*	1.8M 9.3M	2.46 / 8 1.83	
USA-road-d.-CRT	network	*GR*	14.0M 68.5M	2.48 / 9 1.88	
USA-road-d.-BAY	network	*GR*	0.3M 1.6M	2.49 / 7 1.90	
USA-road-d.-FLA	network	*GR*	1.0M 5.4M	2.53 / 9 1.91	
USA-road-d.-COL	network	*GR*	0.4M 2.1M	2.42 / 8 1.93	
pwtk	mesh	*MTX*, *RB*	0.2M 5.9M	55.39 / 181 2.20	
rgg-n-2-24-s0	mesh	*MTX*, *RB*	16.7M 265.1M	15.97 / 40 51.35	
msdoor	mesh	*MTX*, *RB*	0.4M 19.1M	49.69 / 78 56.03	

Graph (original name)		avail. Format	Nodes Edges	Degrees avg. / max. excess (%)	Spyplot
it-2004	scale-free graph	*MTX*, *RB*	41.2M 1.150.7M	25.31 / 1, 326k 6.99	
indochina-2004	scale-free graph	*MTX*, *RB*	7.4M 194.1M	25.48 / 256.4k 7.16	
kron_g500-logn21	scale-free graph	*MTX*, *RB*	2.1M 182.0M	86.44 / 213.9k 8.06	
amazon-2008	scale-free graph	*MTX*, *RB*	0.7M 5.1M	14.01/1k 11.57	
twitter-retweet	scale-free graph	*MTX*	1.1M 4.5M	4.05/4.3k 11.87	
hollywood-2009	scale-free graph	*MTX*, *RB*	1.1M 113.8M	100.83 / 11.5k 21.54	
ljournal-2008	scale-free graph	*MTX*, *RB*	5.3M 79.0M	14.73 / 19.3k 22.91	
coAuthorsDBLP	scale-free graph	*MTX*, *RB*	0.3M 1.9M	6.53 / 336 26.98	
soc-orkut	scale-free graph	*MTX*	3M 106.3M	70.32 / 27.3k 30.67	
dblp-2010	scale-free graph	*MTX*, *RB*	0.3M 1.6M	4.95/238 30.76	

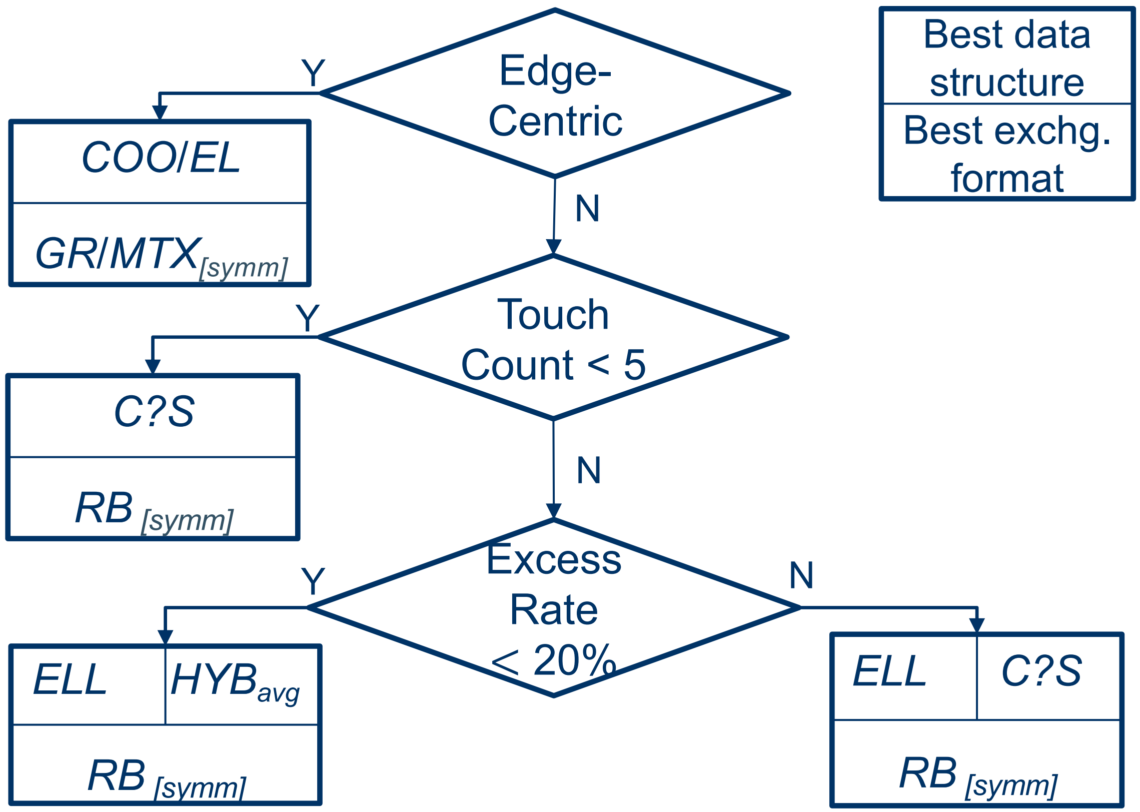
Study Setup – GPUs

- 4 architecturally different Nvidia GPUs
  - Titan XP, 12.2 GB, Pascal
  - Geforce GTX 980, 4.0 GB, Maxwell
  - Geforce GTX 680, 2.0 GB, Kepler
  - Geforce GTX 580, 1.0 GB, Fermi

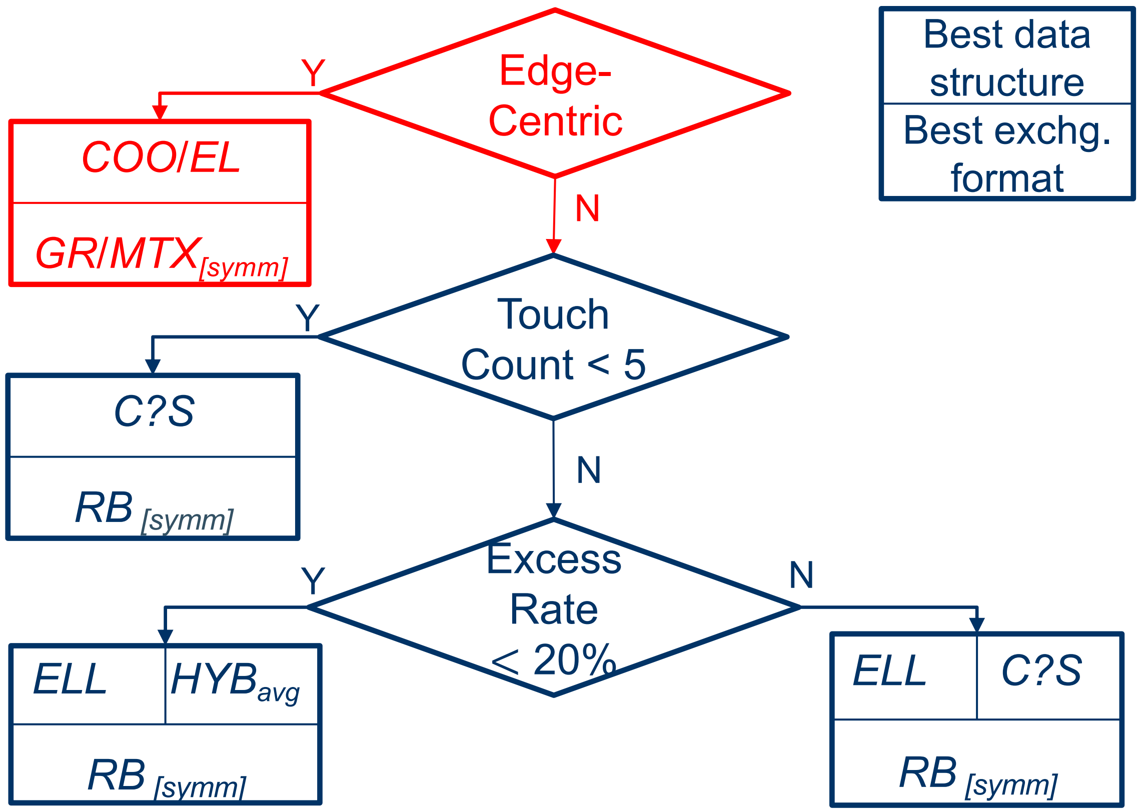
- Only Titan XP can run all measurement configurations.

- All measurements scale w.r.t. nominal GPU performance.
  → Suffices to discuss the Titan XP measurements.

Decision Tree (file-to-file)

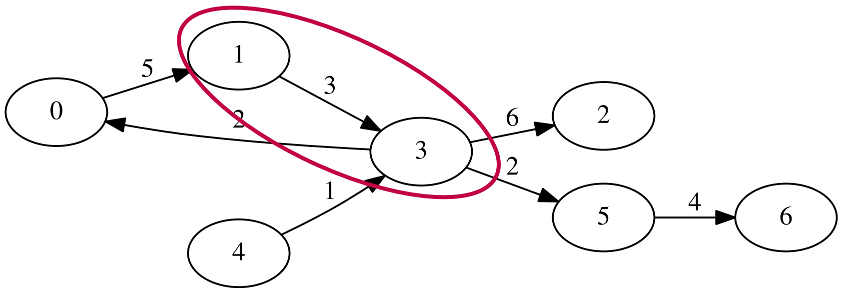


Decision Tree – First Layer (I)



Decision Tree – First Layer (II)

- First decision depends on whether the algorithm is edge-centric or node-centric.
  - Splits data structures into two groups.
- For edge-centric algorithms *COO* and *EL* perform best.
  - The other formats are 25% slower in our measurements.



- **COO**rdinate format
  - Stores source, target, and weight of an edge in different arrays.

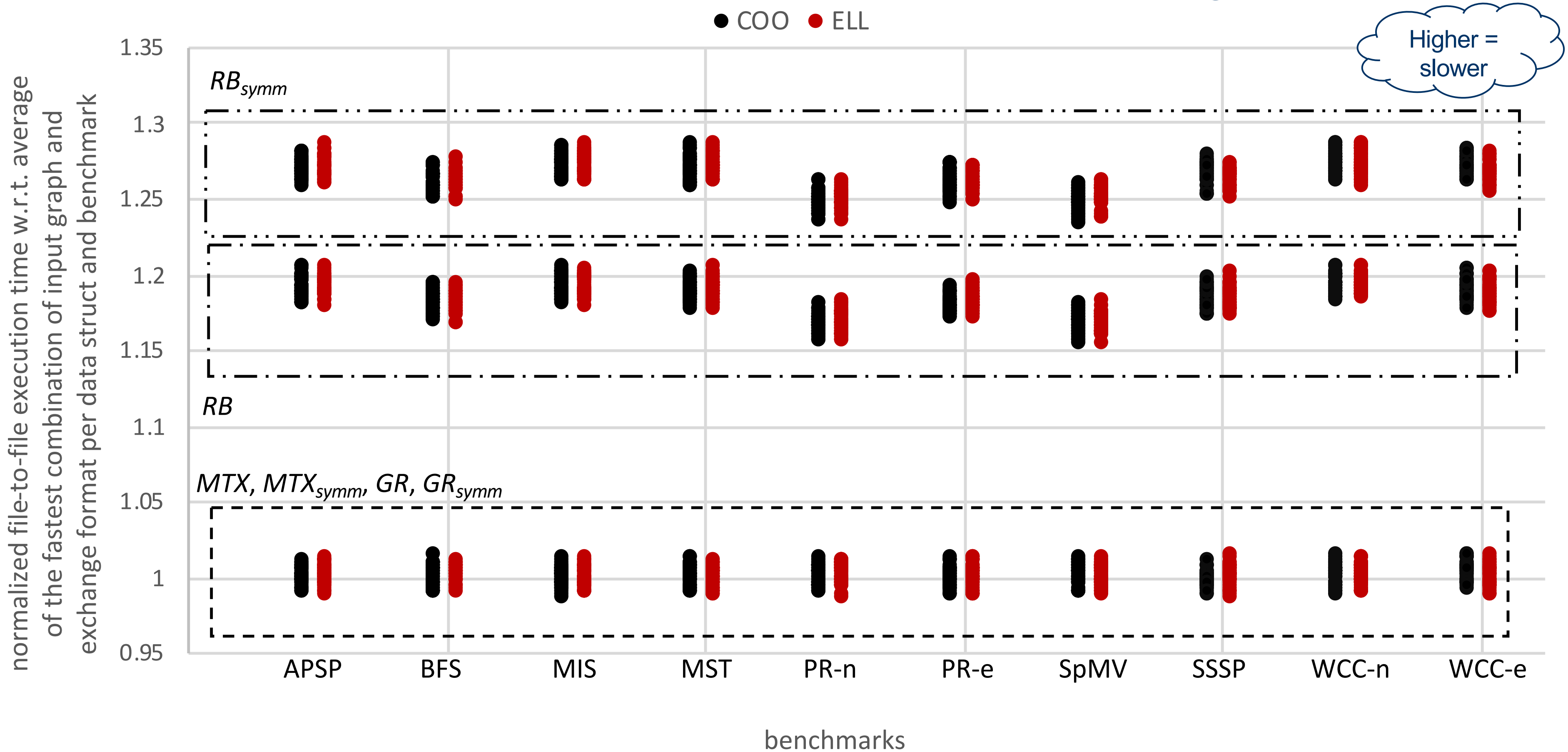
$$
\begin{aligned}
src &= [0 \quad 1 \quad 3 \quad 3 \quad 3 \quad 4 \quad 5] \\
trgt &= [1 \quad 3 \quad 0 \quad 2 \quad 5 \quad 3 \quad 6] \\
weights &= [5 \quad 3 \quad 2 \quad 6 \quad 2 \quad 1 \quad 4]
\end{aligned}
$$

- **E**dge **L**ist format
  - Stores an edge as a struct in an array.

$$
edges = \left[\begin{Bmatrix} 0 \\ 1 \\ 5 \end{Bmatrix}, \begin{Bmatrix} 1 \\ 3 \\ 3 \end{Bmatrix}, \begin{Bmatrix} 3 \\ 0 \\ 2 \end{Bmatrix}, \cdots \right]
$$

Decision Tree – First Layer (III)

- *COO* and *EL* show the same runtime behavior.
  - Either *COO* or *EL* can be picked for edge-centric algorithms.
  - Runtime can be minimized with MTX/GR exchange format.



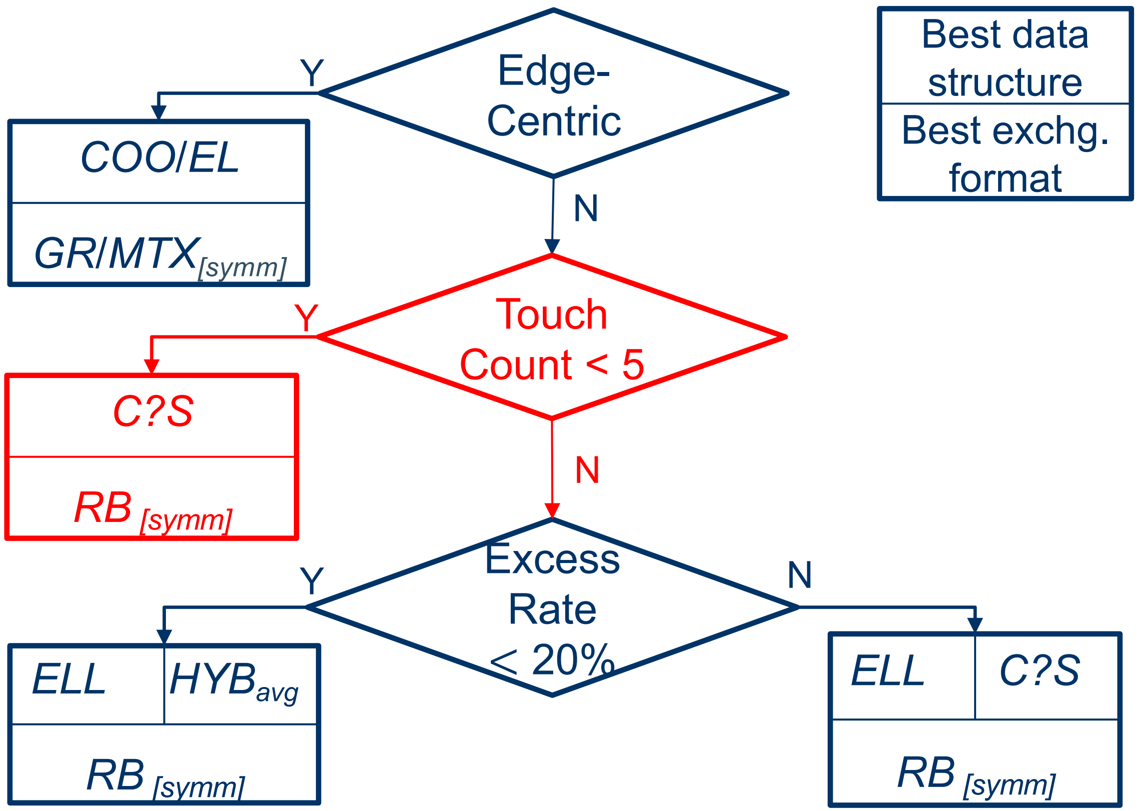
Decision Tree – First Layer (IV)

- The *MTX* and *GR* input file formats are dumps of the *COO* and *EL* formats.
- No costly conversion from file to representation in CPU memory.

	$< MTX\ header >$	$< GR\ header >$
Size of the array(s)	$7\ 7\ 7$	$p\ sp\ 7\ 7\ 7$
Encoded edges	$0\quad 1\quad 5$	$a\quad 0\quad 1\quad 5$
	$1\quad 3\quad 3$	$a\quad 1\quad 3\quad 3$
	$3\quad 0\quad 2$	$a\quad 3\quad 0\quad 2$
	$3\quad 2\quad 6$	$a\quad 3\quad 2\quad 6$
	$3\quad 5\quad 2$	$a\quad 3\quad 5\quad 2$
	$4\quad 3\quad 1$	$a\quad 4\quad 3\quad 1$
	$5\quad 6\quad 4$	$a\quad 5\quad 6\quad 4$

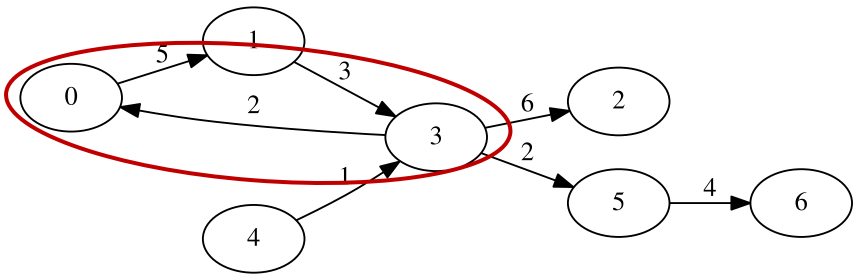
- The RB format is 20% slower in our measurements.

Decision Tree (file-to-file)



Decision Tree – Second Layer (I)

- Node-centric algorithms need a deeper analysis.



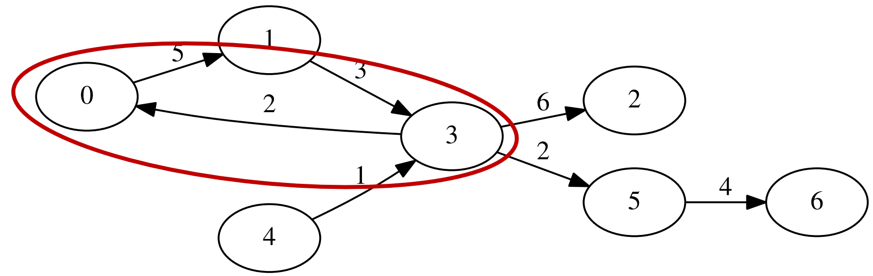
- **C**ompressed **R**ow **S**torage format
  - Most memory efficient format.
  - Indirect addressing step necessary.

$$src_{ptr} = [0 \quad 1 \quad 2 \quad 2 \quad 5 \quad 6 \quad 7 \quad 7]$$
$$\boldsymbol{succ} = [1 \quad 3 \quad 0 \quad 2 \quad 5 \quad 3 \quad 6]$$
$$weights = [5 \quad 3 \quad 2 \quad 6 \quad 2 \quad 1 \quad 4]$$

- **C**ompressed **C**olumn **S**torage format
  - Same as CRS but stores the predecessors of a node.

$$src_{ptr} = [0 \quad 1 \quad 2 \quad 3 \quad 5 \quad 5 \quad 7 \quad 8]$$
$$\boldsymbol{pred} = [3 \quad 0 \quad 3 \quad 1 \quad 4 \quad 3 \quad 5]$$
$$weight = [2 \quad 5 \quad 6 \quad 3 \quad 1 \quad 2 \quad 4]$$

Decision Tree – Second Layer (II)



- **ELL** format
  - Fixed row length reserved for successors.
  - Well suited for vector processors.
  - Memory efficiency depends on degree distribution of a graph.

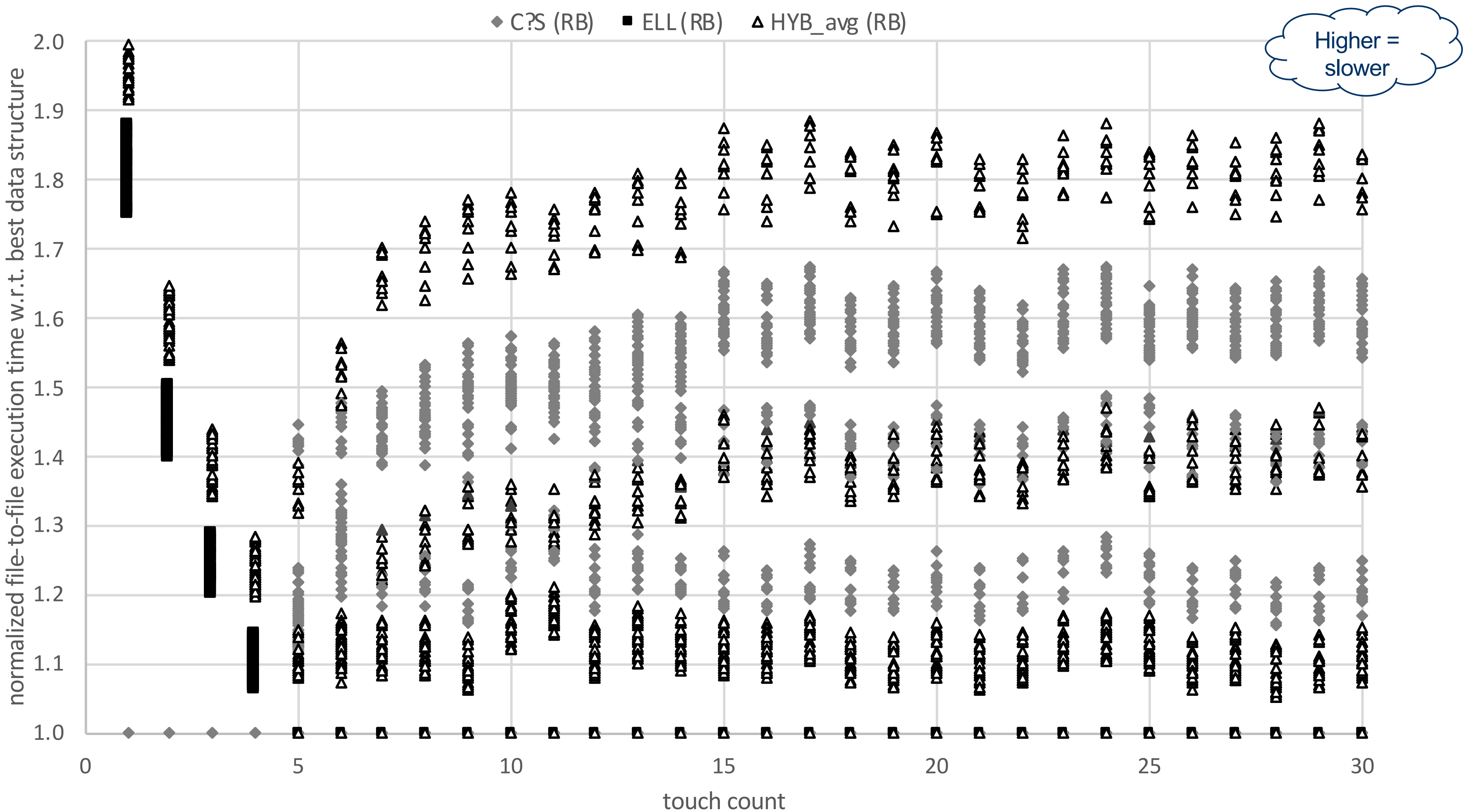
$$\begin{array}{cc} succ & weights \\ \begin{bmatrix} 1 & * & * \\ 3 & * & * \\ * & * & * \\ 0 & 2 & 5 \\ 3 & * & * \\ 6 & * & * \\ * & * & * \end{bmatrix} & \begin{bmatrix} 5 & * & * \\ 3 & * & * \\ * & * & * \\ 2 & 6 & 2 \\ 1 & * & * \\ 4 & * & * \\ * & * & * \end{bmatrix} \end{array}$$

- **HYB**rid format
  - Better memory efficiency than *ELL*.
  - Heuristics (avg, dstr) find smaller the row length.
  - Additional successors are stored in *EL*.

$$\begin{array}{ccc} ell_v & ell_w & el \\ \begin{bmatrix} 1 \\ 3 \\ * \\ 0 \\ 3 \\ 6 \\ * \end{bmatrix} & \begin{bmatrix} 5 \\ 3 \\ * \\ 2 \\ 1 \\ 4 \\ * \end{bmatrix} & \left[\begin{Bmatrix} \begin{pmatrix} 3 \\ 2 \\ 6 \end{pmatrix} \end{Bmatrix}, \begin{Bmatrix} \begin{pmatrix} 3 \\ 5 \\ 2 \end{pmatrix} \end{Bmatrix} \right] \end{array}$$

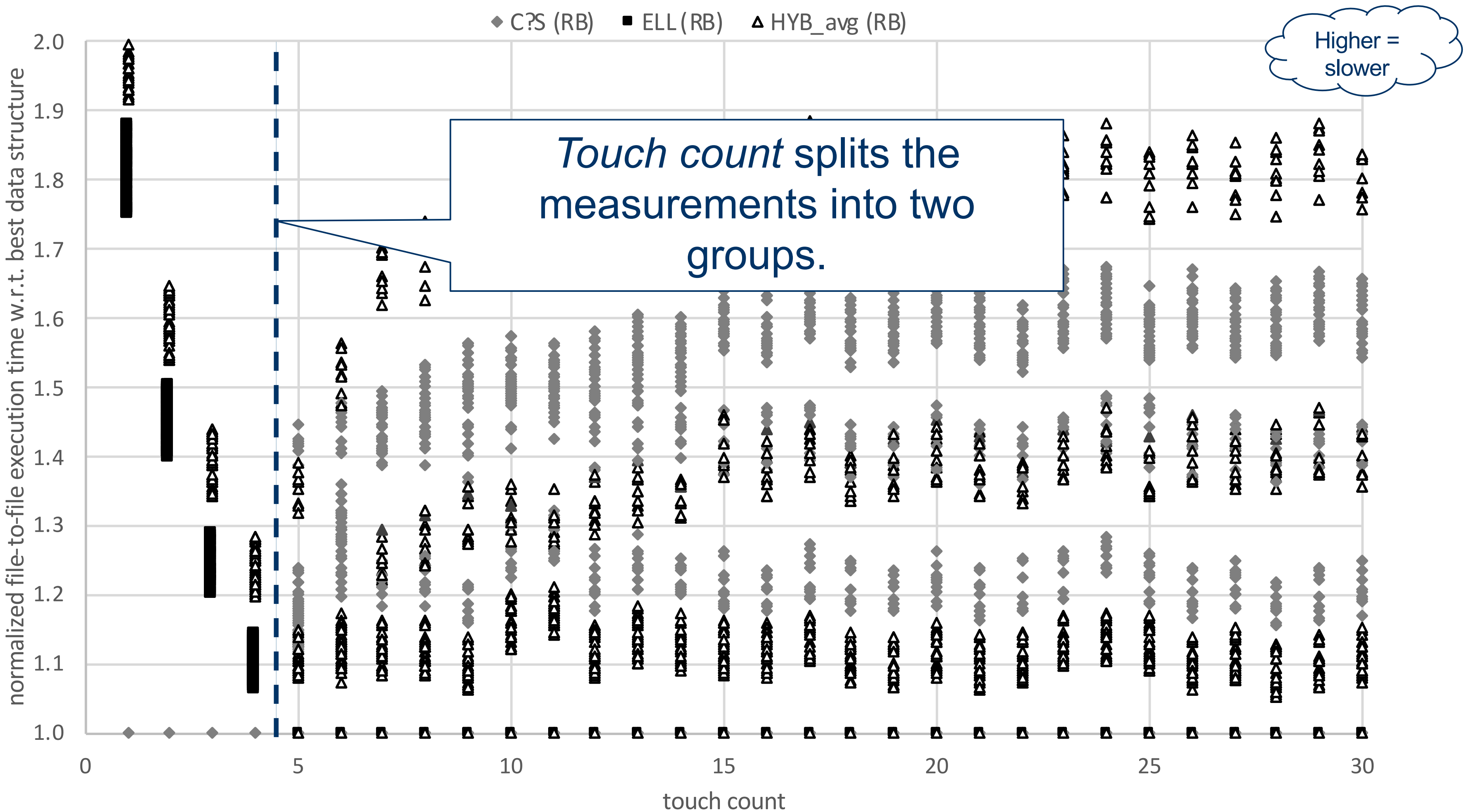
Decision Tree – Second Layer (III)

- Data structures cause costs for construction, shipping, ...



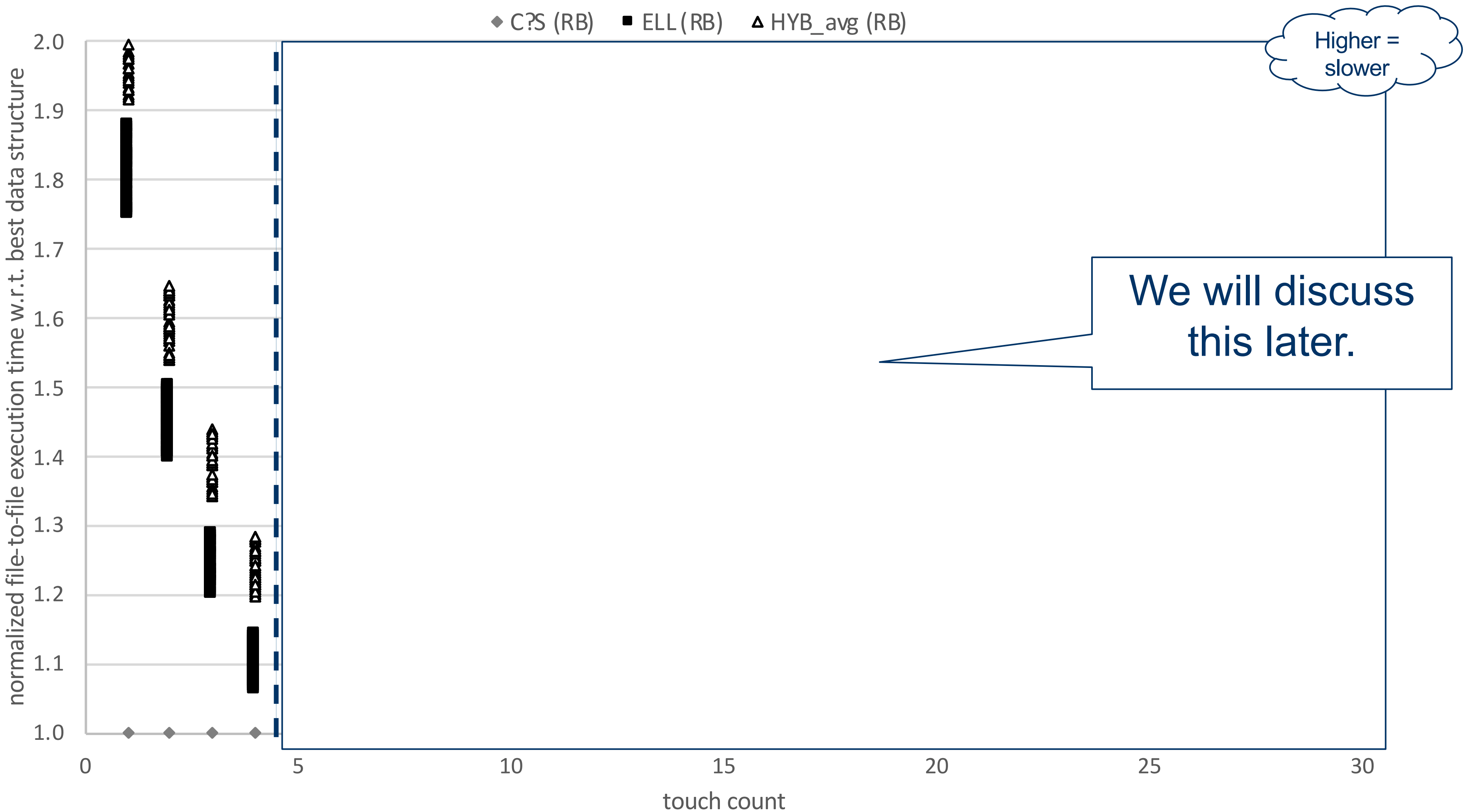
Decision Tree – Second Layer (III)

- Data structures cause costs for construction, shipping, ...



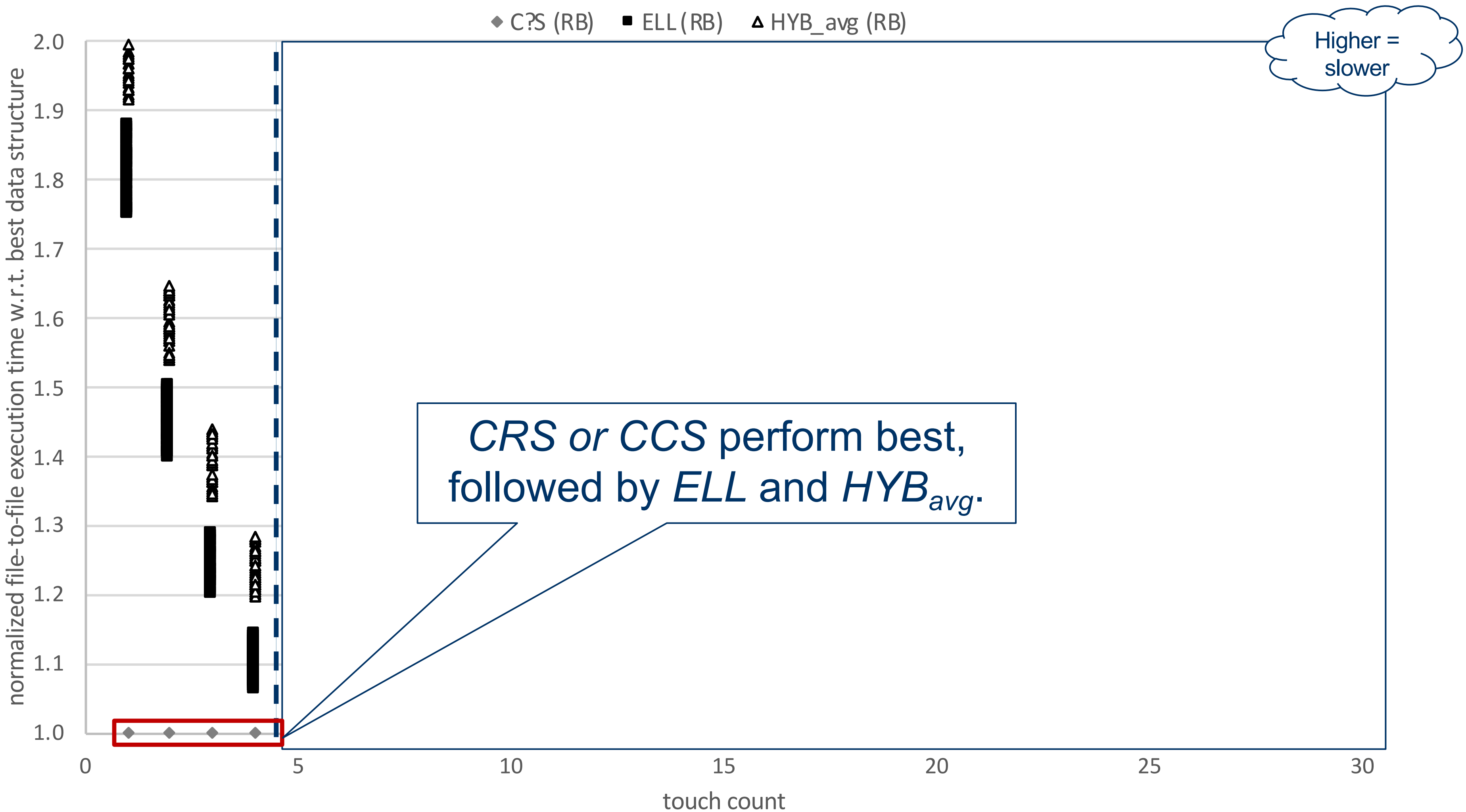
Decision Tree – Second Layer (III)

- Data structures cause costs for construction, shipping, ...



Decision Tree – Second Layer (III)

- Data structures cause costs for construction, shipping, ...



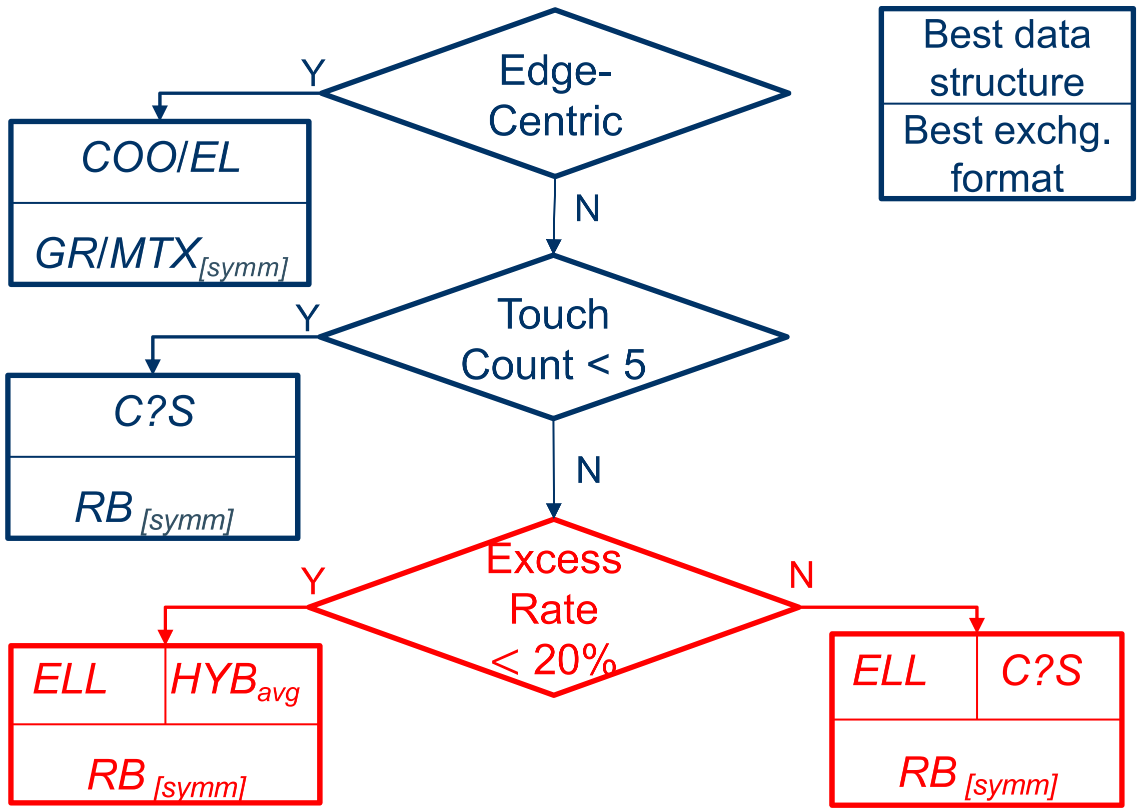
Decision Tree – Second Layer (IV)

- The Rutherford Boeing (RB) format is a dump of the *CRS* format.

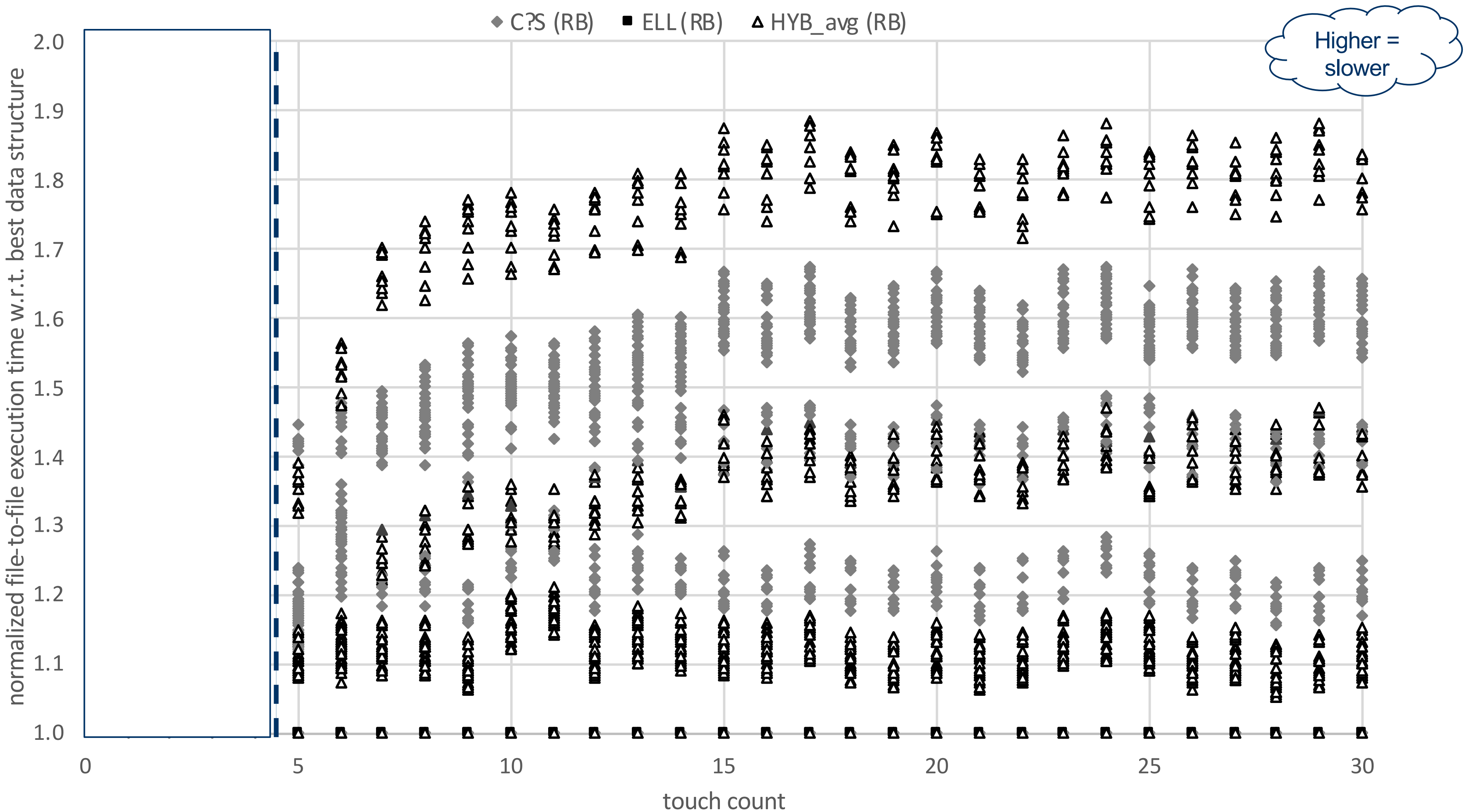
$$
\begin{array}{l}
\left.\begin{array}{llll}
8 & 2 & 2 & 2 \\
iga & 7 & 7 & 7 \\
\multicolumn{4}{l}{(11I4)(11I3)}
\end{array}\right\} \textit{Information about the structure of the encoded graph and the file.} \\
\left.\begin{array}{llll}
0 & 1 & 2 & 2 \\
5 & 6 & 7 &
\end{array}\right\} src_{ptr} \\
\left.\begin{array}{llll}
1 & 3 & 0 & 2 \\
5 & 3 & 6 &
\end{array}\right\} succ/pred \\
\left.\begin{array}{llll}
5 & 3 & 2 & 6 \\
2 & 1 & 4 &
\end{array}\right\} weights
\end{array}
$$

- Other file formats are 40% slower in our measurements.

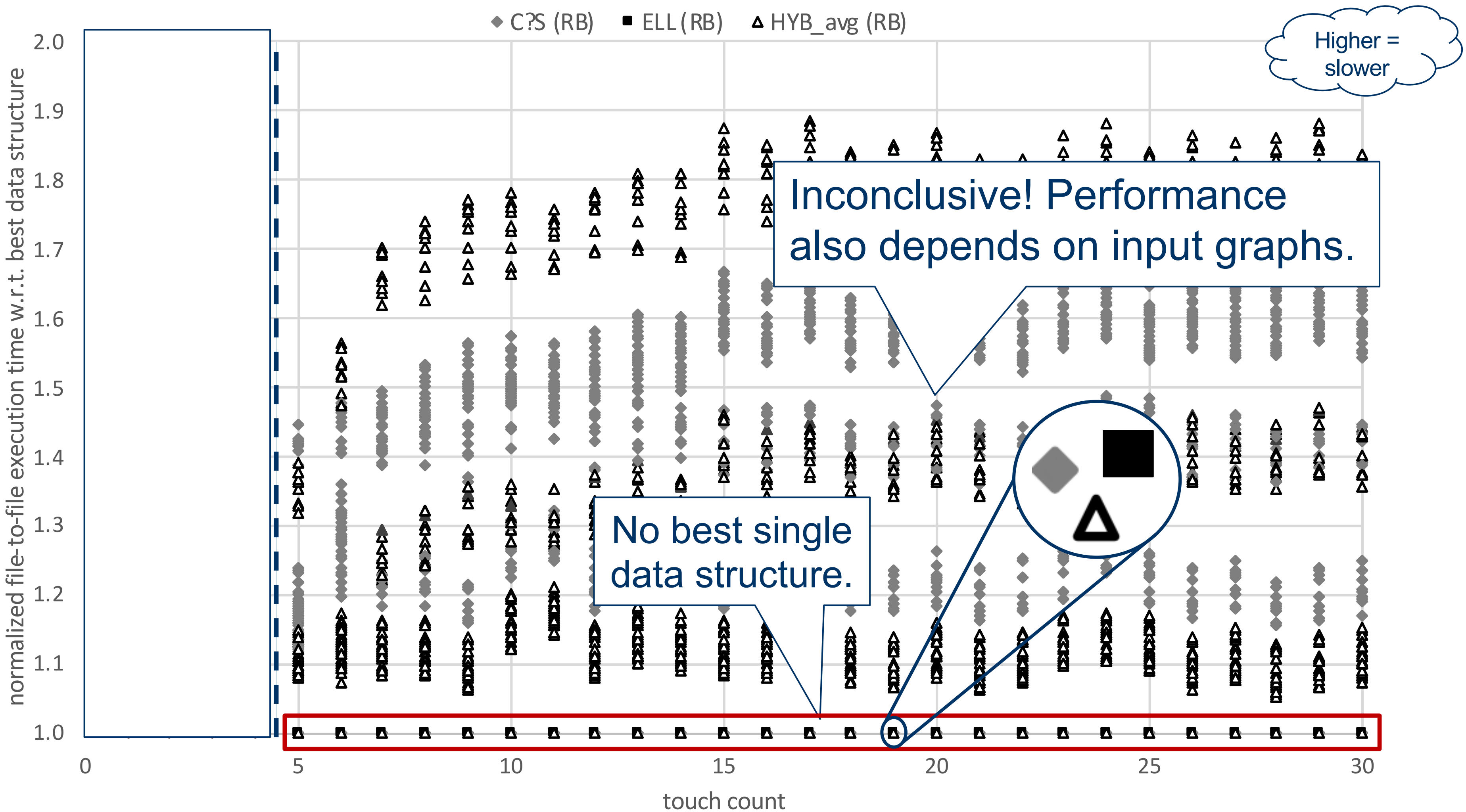
Decision Tree (file-to-file)



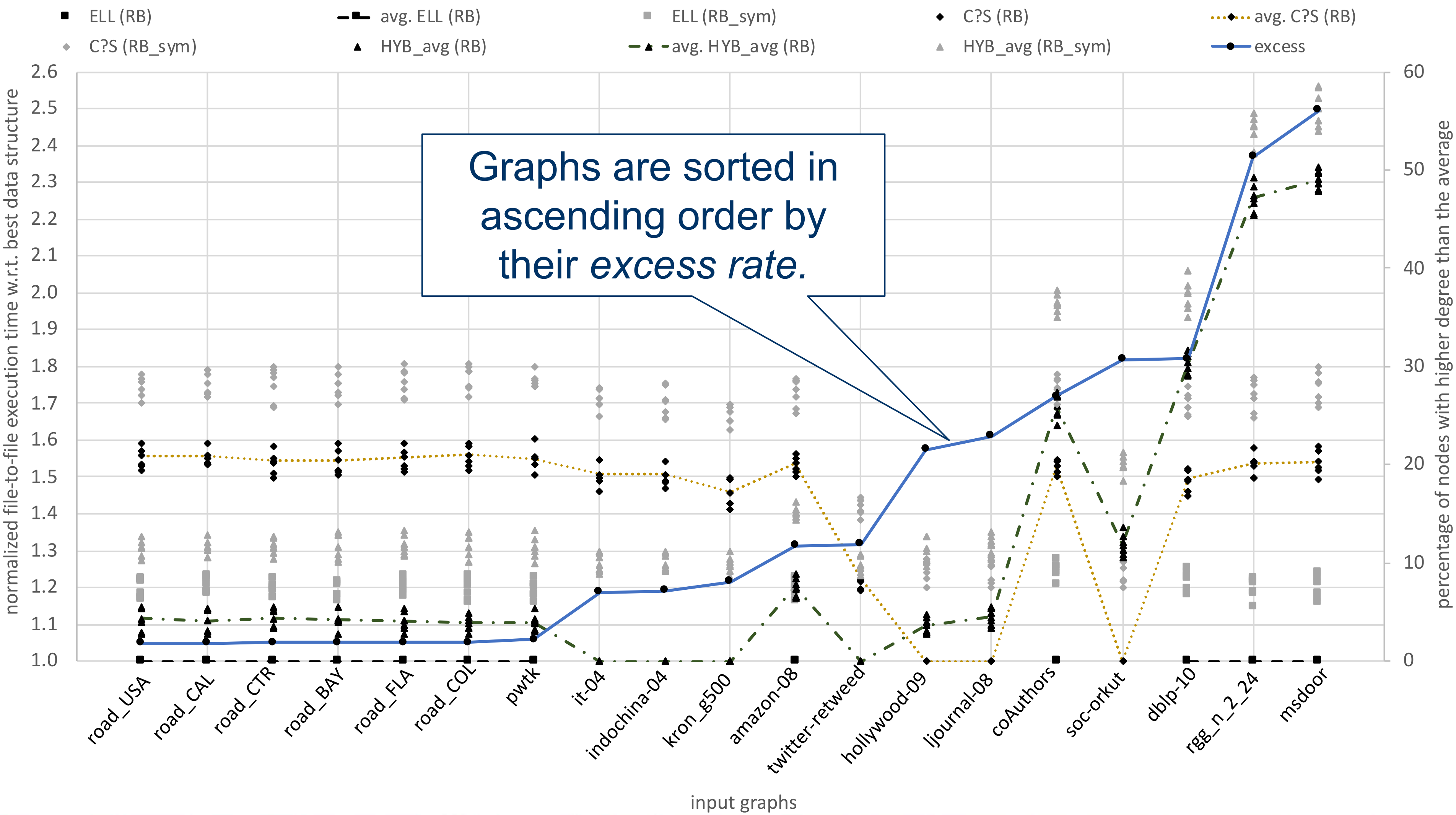
Decision Tree – Third Layer (I)



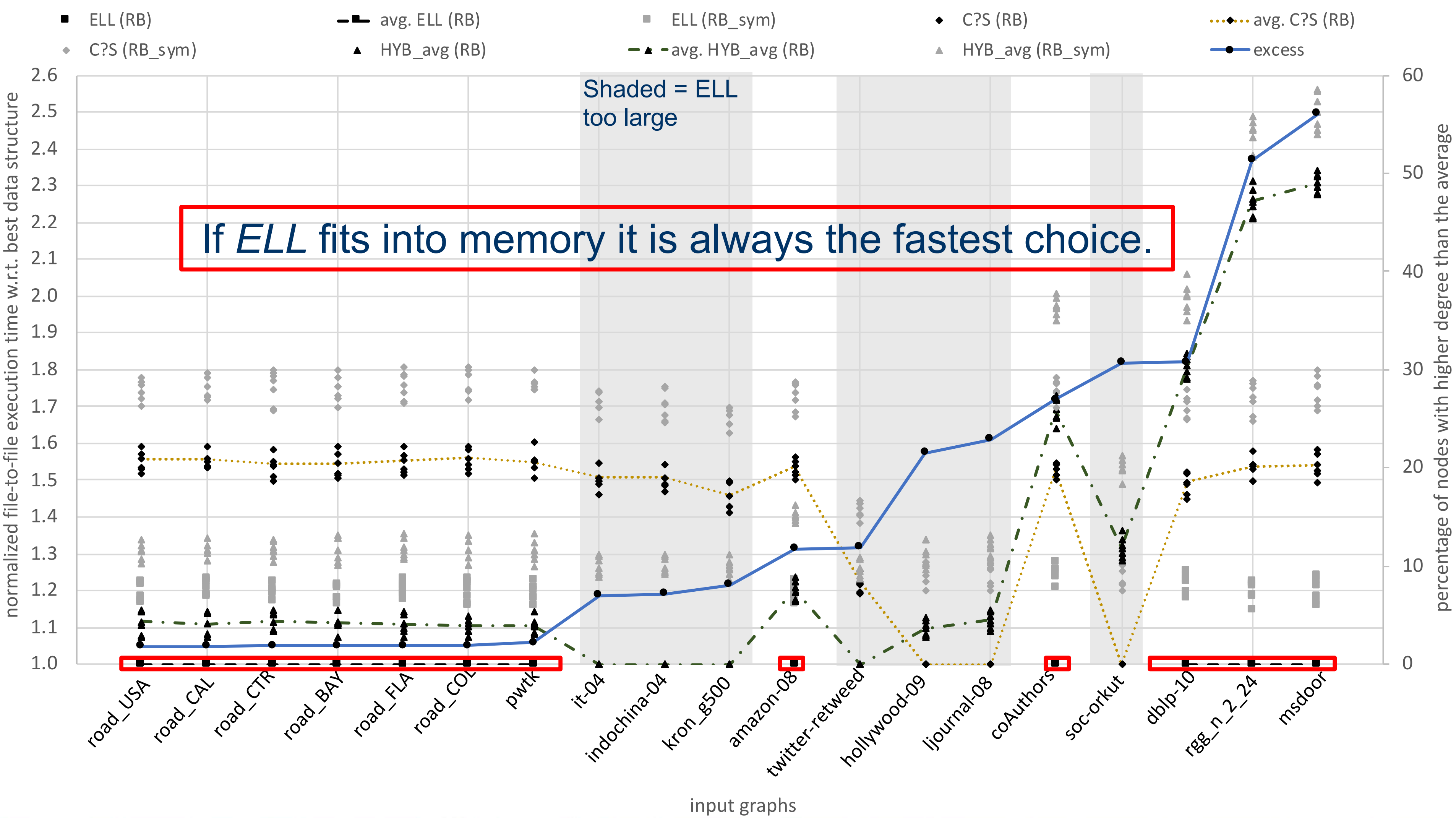
Decision Tree – Third Layer (I)



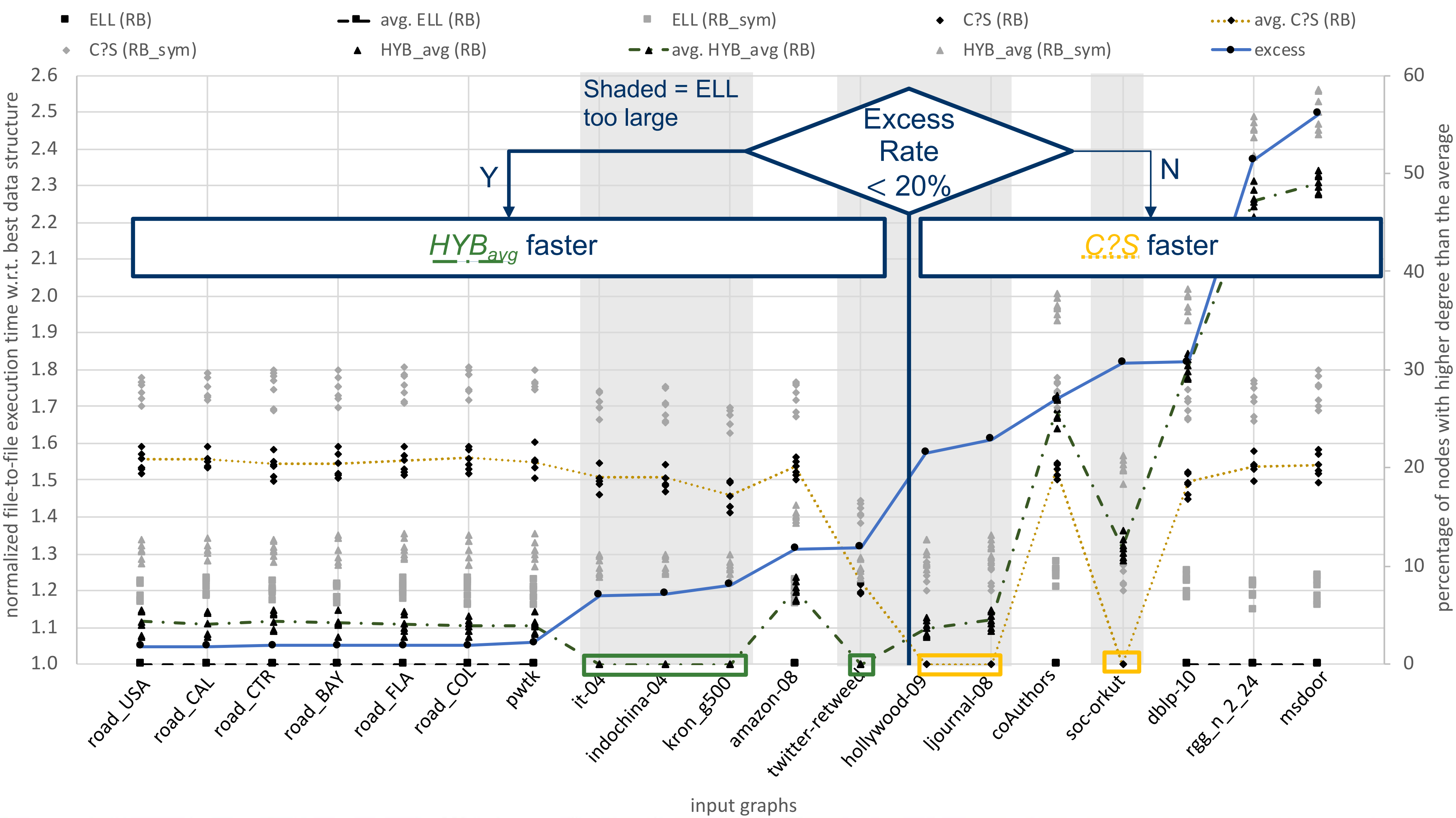
Decision Tree – Third Layer (II)



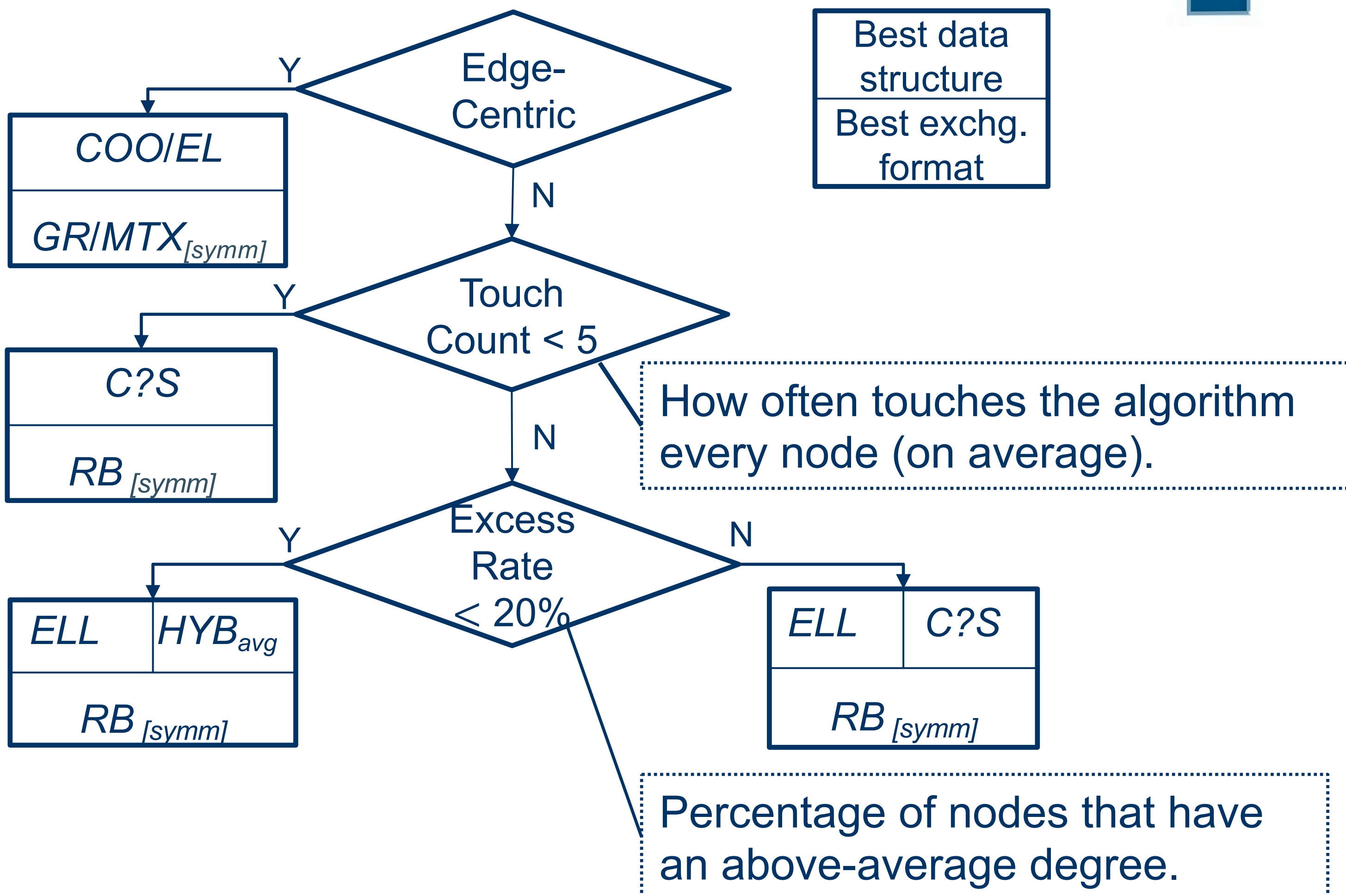
Decision Tree – Third Layer (II)



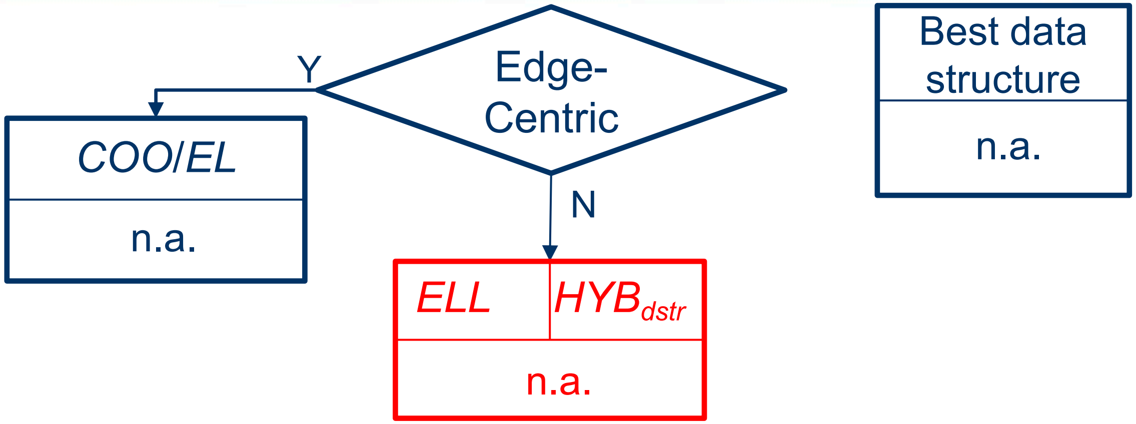
Decision Tree – Third Layer (II)



Decision Tree (file-to-file)

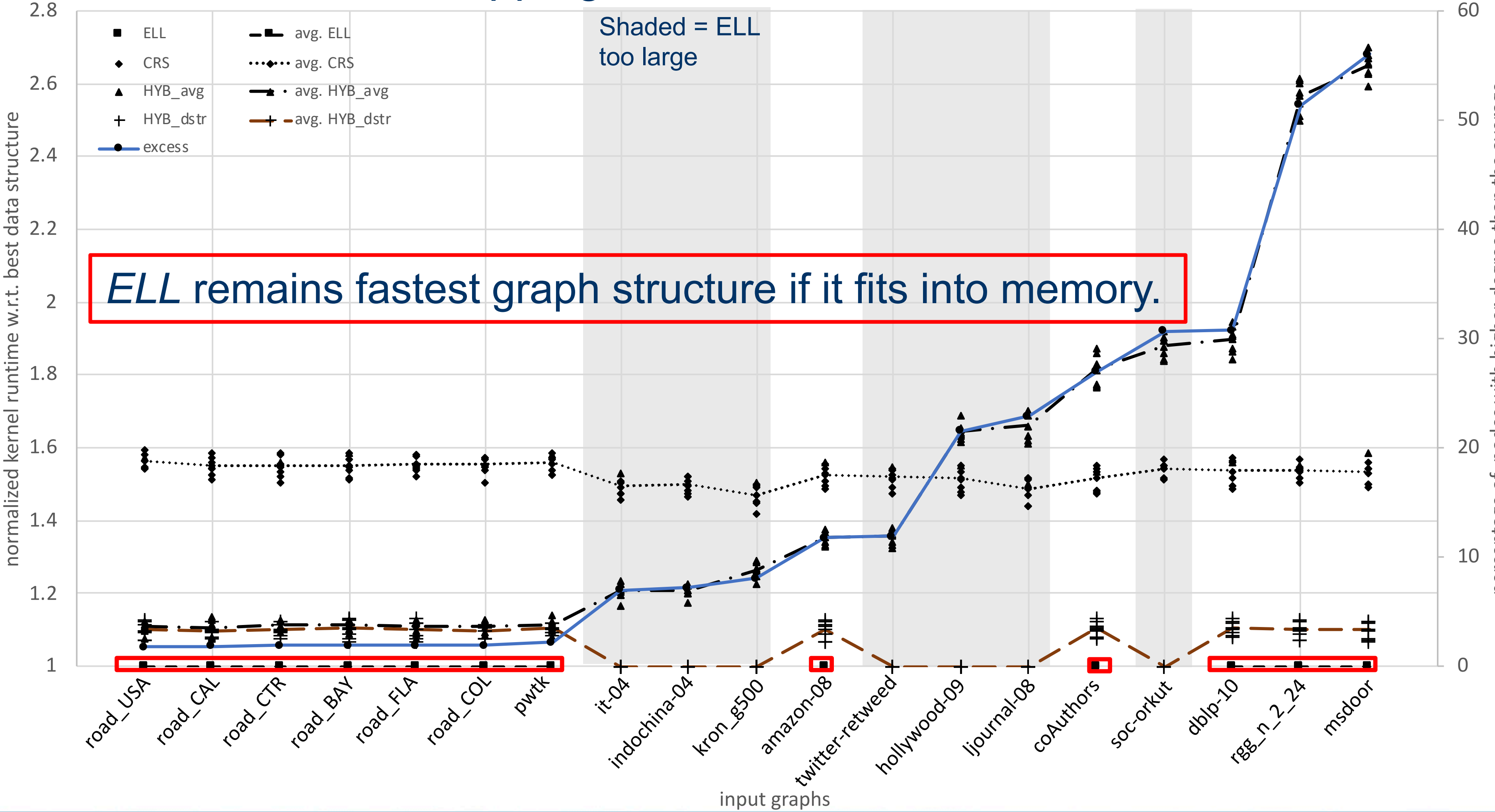


Decision Tree – Kernel only



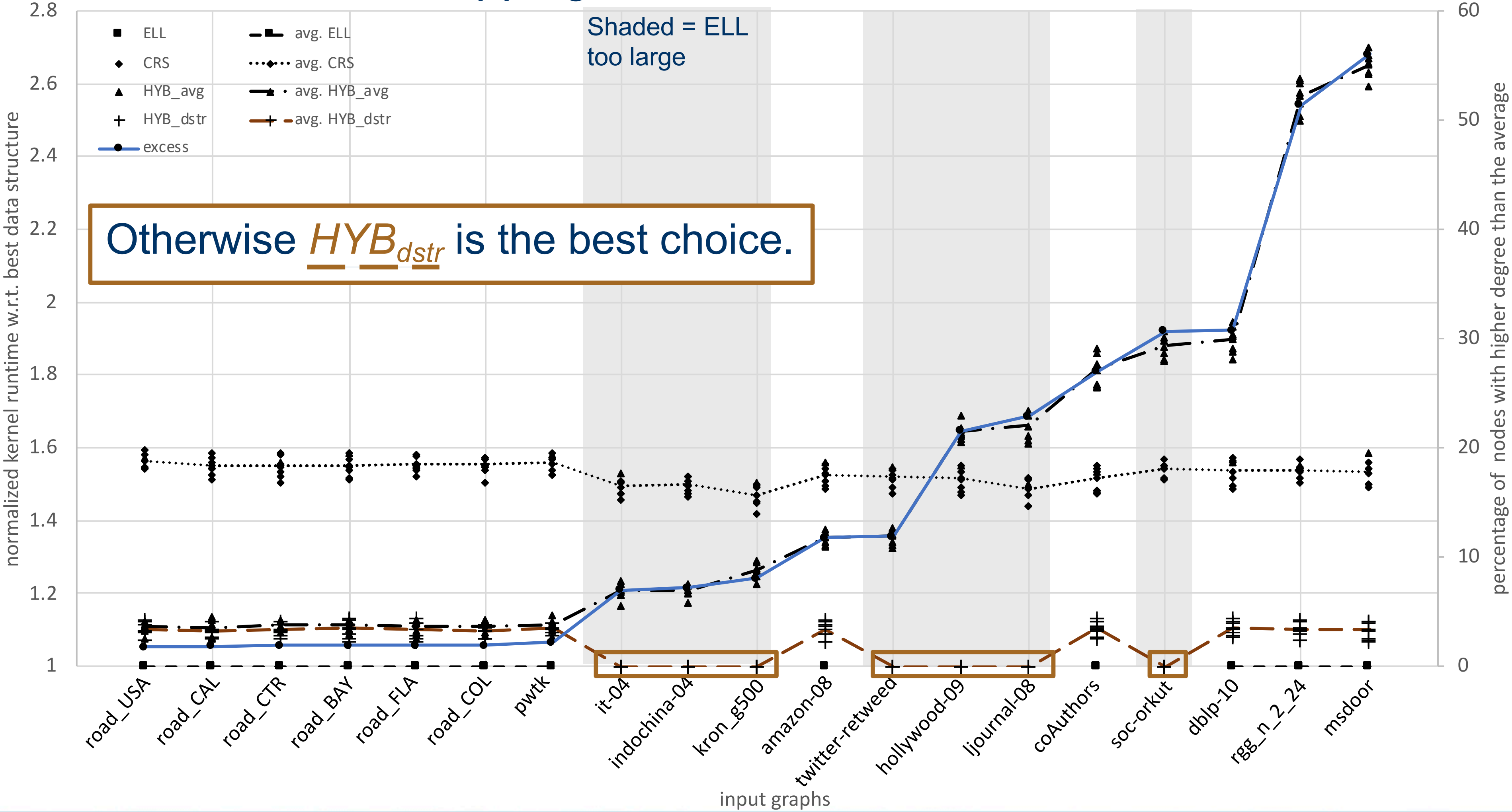
Kernel-only results

- Without I/O, data structure construction, and CPU ←→ GPU shipping.



Kernel-only results

- Without I/O, data structure construction, and CPU ←→ GPU shipping.



Conclusion

- There is no single best graph data structure for GPUs.
- Three decision layers:
  - Edge-centric vs. node-centric
  - Touch count threshold around 5
  - Excess rate threshold around 20%
- Our Decision Tree can help developers
  to pick an adequate data structure for their use case.
- Choosing the adequate data structure can
  speed up graph algorithms by up to 45%.

Conclusion

- There is no single best graph data structure for GPUs.
- Three decision layers:
  - Edge-centric vs. node-centric
  - Touch count threshold around 5
  - Excess rate threshold around 20%
- Our Decision Tree can help developers
  to pick an adequate data structure for their use case.
- Choosing the adequate data structure can
  speed up graph algorithms by up to 45%.

Thank you for your attention!
Any questions?