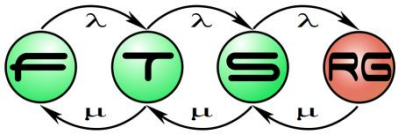
Multiplex graph analysis with GraphBLAS

Gábor Szárnyas

With contributions from Petra Várhegyi and Lehel Boér

Fault Tolerant Systems Research Group
MTA-BME Lendület Cyber-Physical Systems Research Group



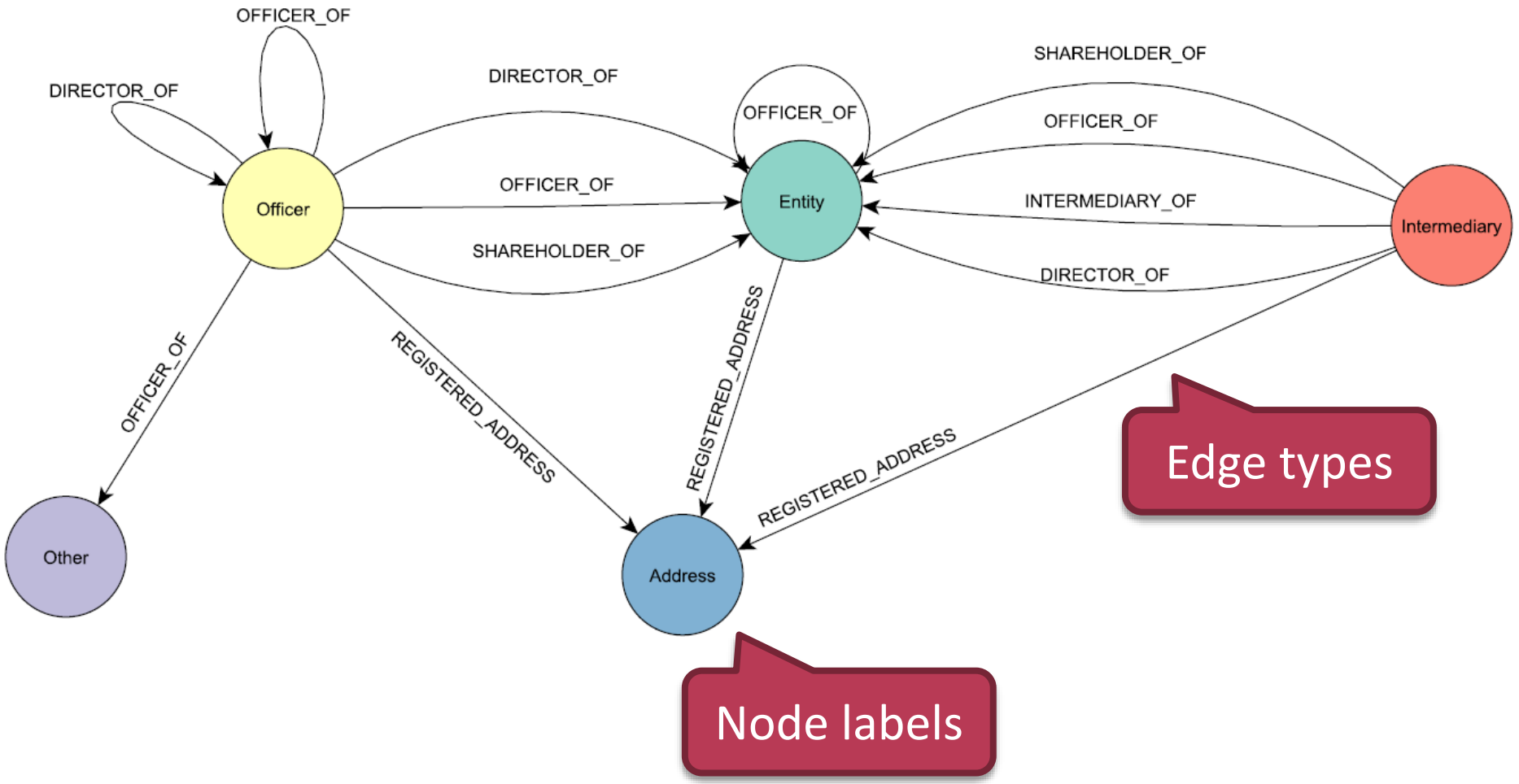
Hungarian Academy of Sciences

PARADISE PAPERS

Paradise papers

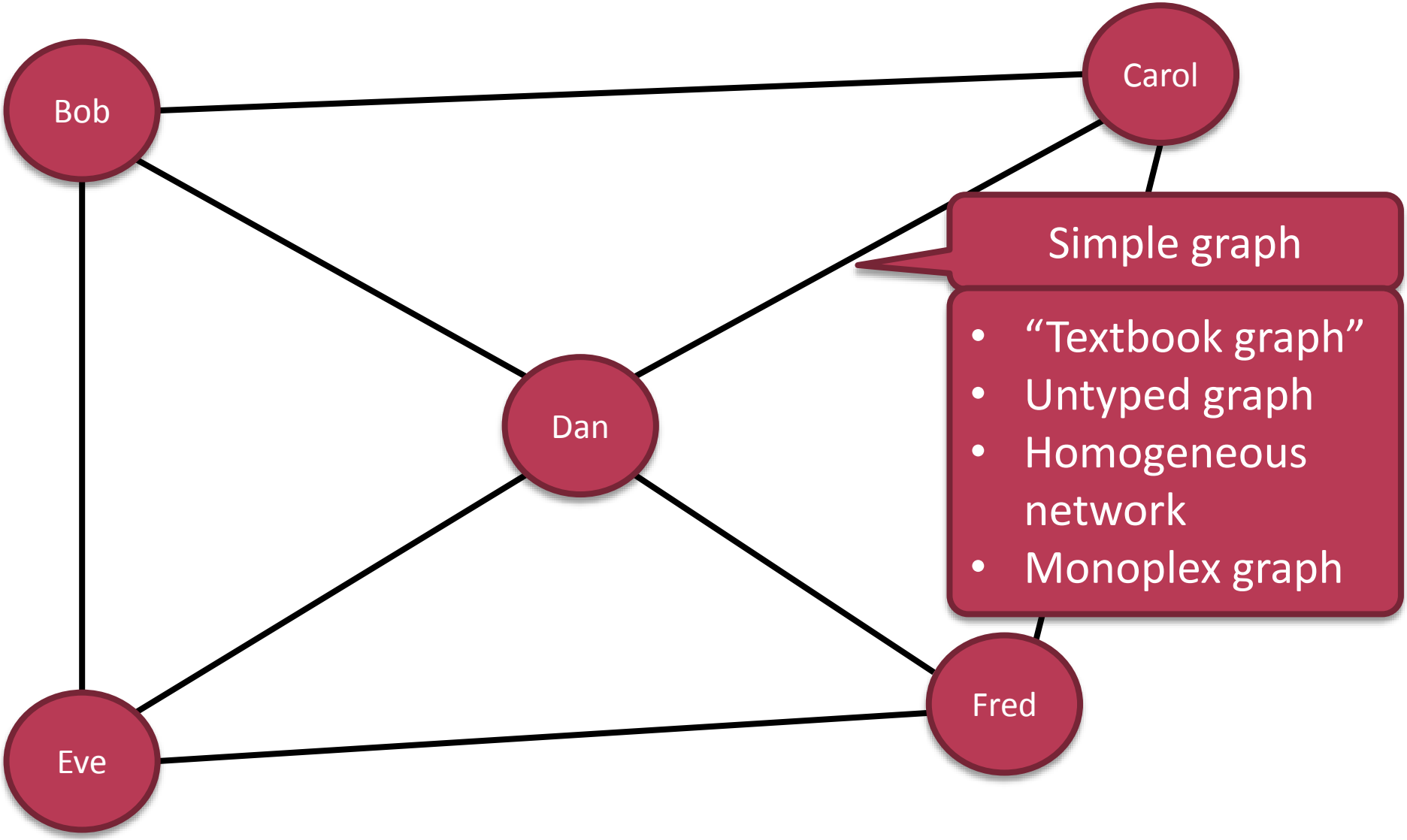
- Data set leaked to investigative journalists
- Similar to Panama Papers, but larger
- Mid-sized data set: 2M nodes, 3M edges

Paradise papers schema

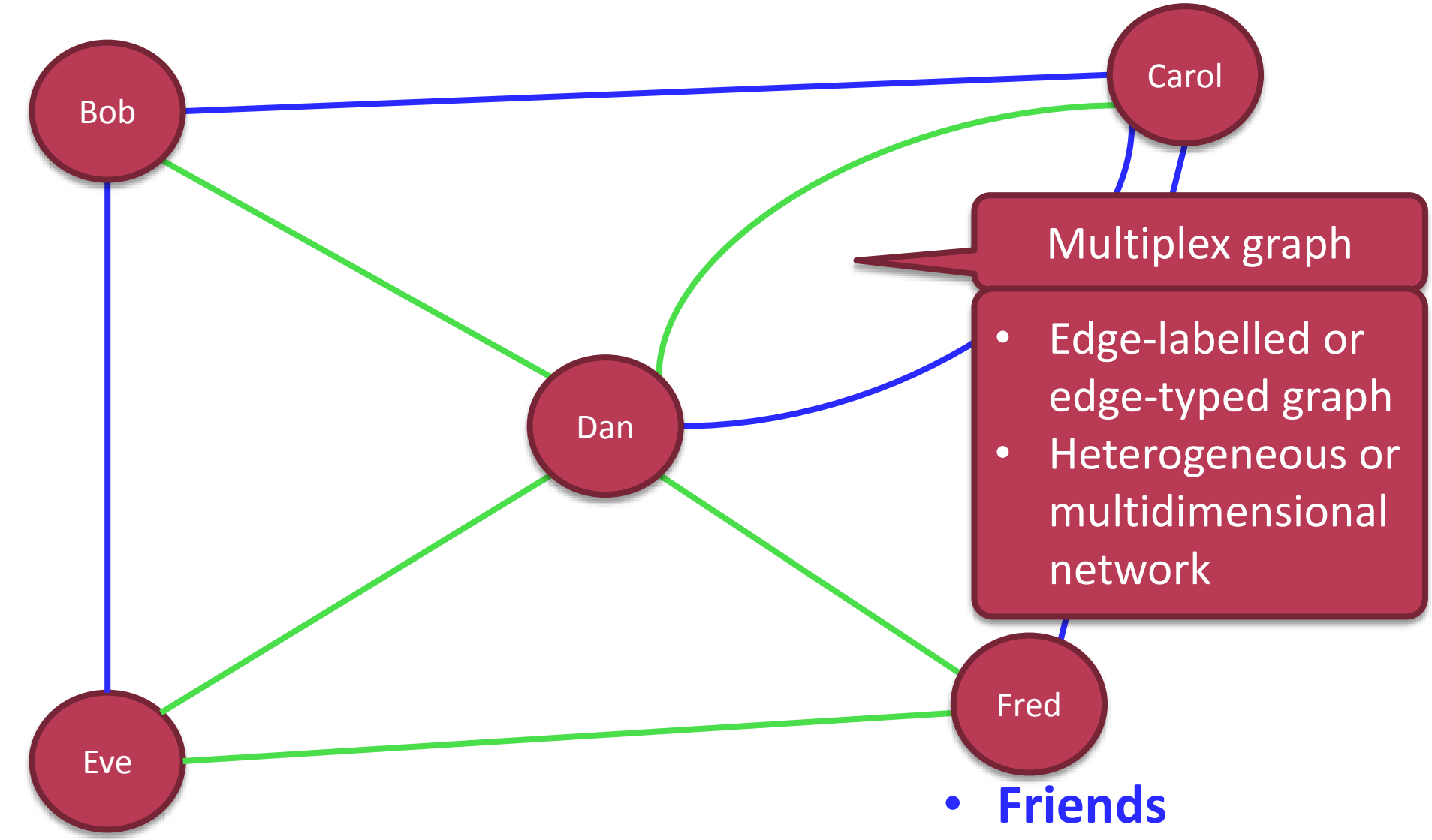


GRAPH DATA MODELS

Example


Bob
Carol
Dan
Eve
Fred
Simple graph
• “Textbook graph”
• Untyped graph
• Homogeneous network
• Monoplex graph

Example with edge types

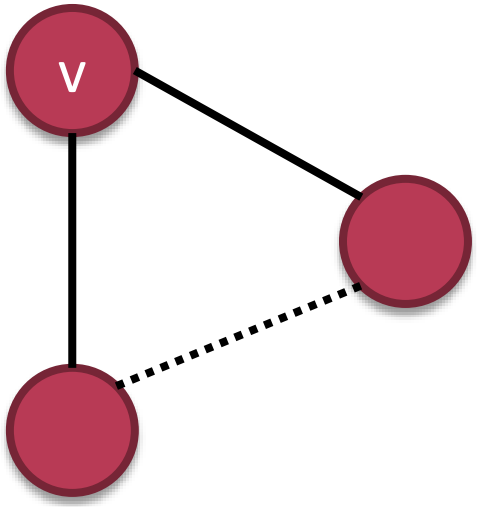


Expressive power: between untyped and property graphs.

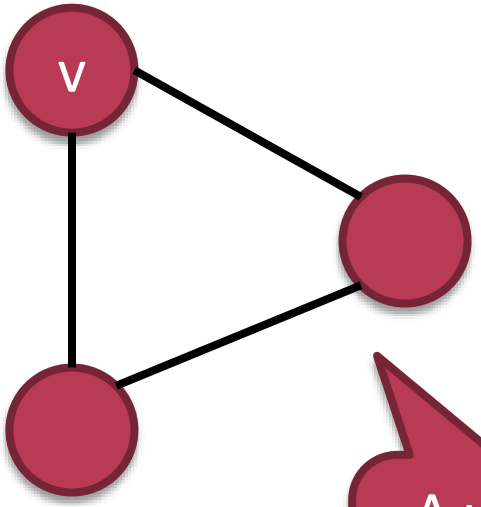
MULTIPLEX GRAPH ANALYSIS

Local clustering coefficient metric

Wedge

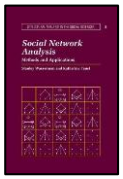


Triangle



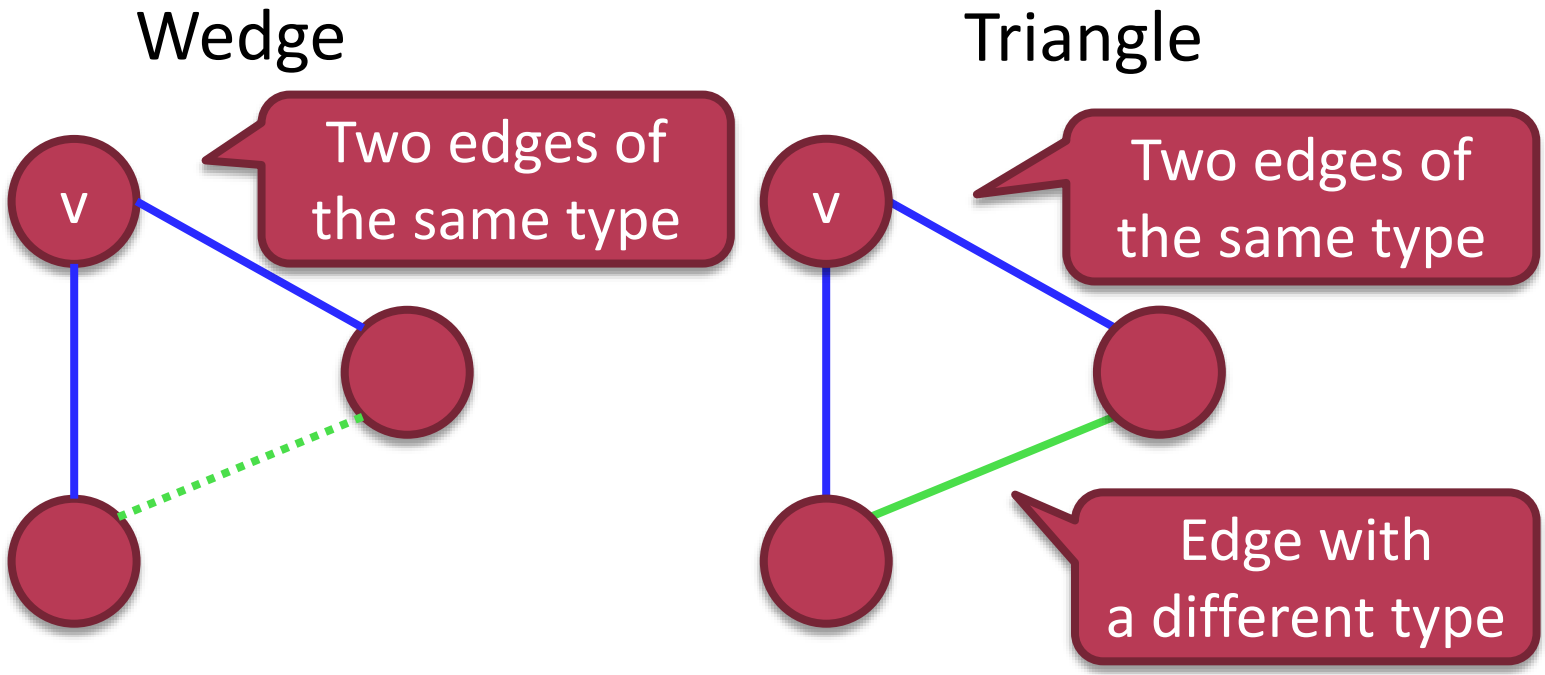
A wedge closes into a triangle

$$\mathrm{LCC}(v) = \frac{\text{No. of triangles in } v}{\text{No. of wedges in } v}$$



K. Faust, S. Wasserman (1994):
Social Network Analysis

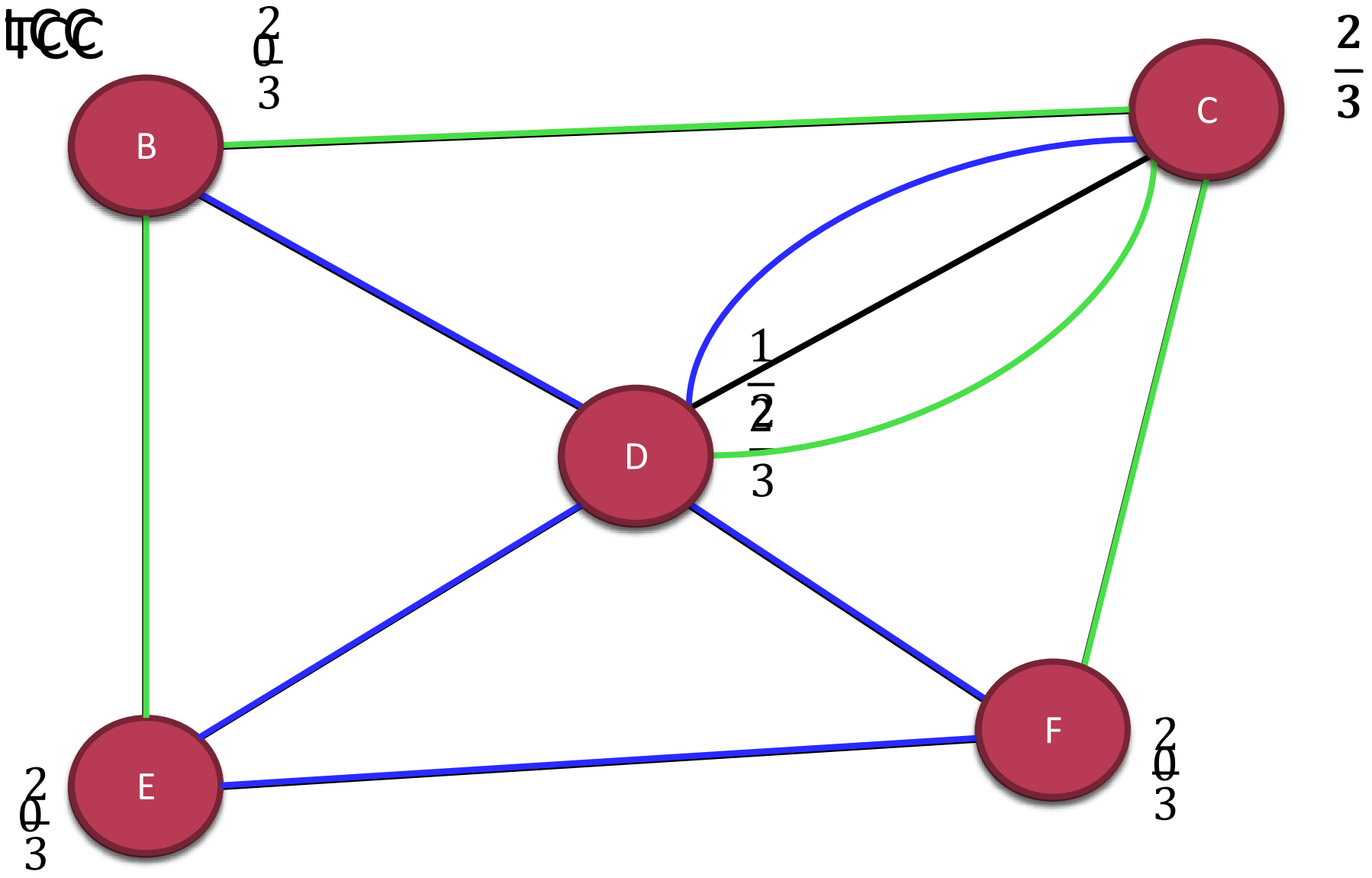
Typed clustering coefficient metric



$$\mathrm{TCC}(v) = \frac{\text{No. of typed triangles in } v}{\text{No. of typed wedges in } v}$$

F. Battiston et al. @ Physical Review E 2014
Structural measures for multiplex networks

Typed clusteredness example



Multiplex analysis on Paradise papers

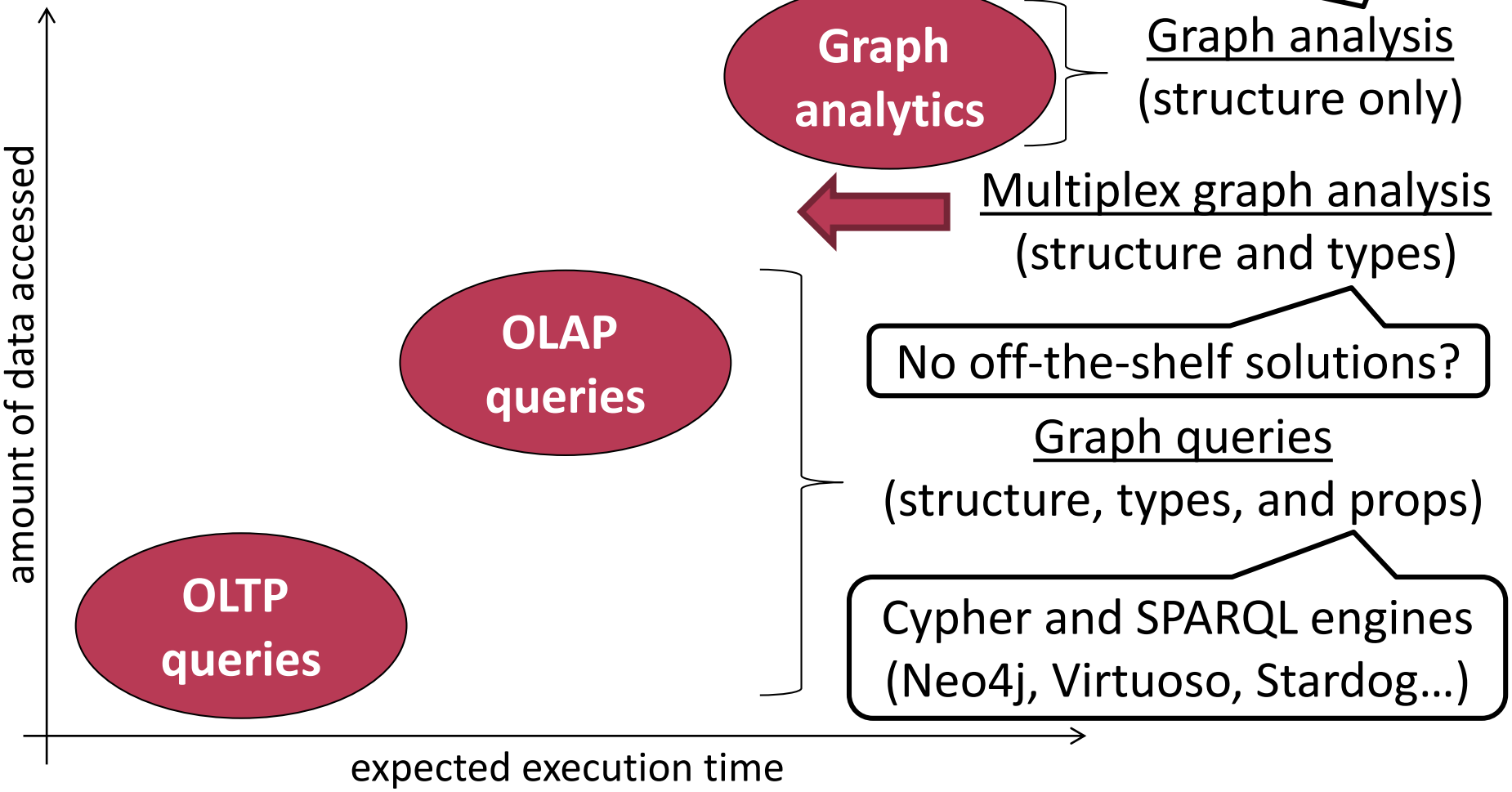
- Previous research
  - Characterization of HW/SW/building models
  - 100k–1M nodes/edges
  - Naïve Java implementation using edge lists
- Ran for Paradise papers data set
  - Clustering coefficient metrics did not complete in days
- What's going on?
  - Implementation and algorithmic aspects need to be tuned

G. Szárnyas et al. @ MODELS 2016
Towards the characterization of realistic models: evaluation of multidisciplinary graph metrics

GRAPH PROCESSING WORKLOADS

Graph processing landscape

- LDBC: Linked Data Benchmark Council



Graph analysis frameworks

Many Apache frameworks can be adapted

- Hama Graph
- Giraph on Hadoop
- Spark GraphX
- Flink Gelly

Flink

But most seem abandoned.

```
$ git rev-list --count --all --no-merges
```

	--since="Feb 2 2015" --before="Feb 2 2017"		--since="Feb 2 2017"
hama/graph	63		0
giraph	154		67
spark/graphx	362		120
flink/flink-libraries/gelly	139		112

Arabesque

- Open-source graph analytical framework over Hadoop
- Frequent subgraph mining:
  find subgraph with a minimum number of matches
- Optimized for distributed execution

```
arabesque$ git rev-list...                    125          91
```

C.H.C. Teixeira et al. @ SOSP 2015
Arabesque: A systems for distributed graph mining

G. Siganos @ FOSDEM 2016
Arabesque: A distributed graph mining platform

Qatar-Computing-Research-Institute/Arabesque

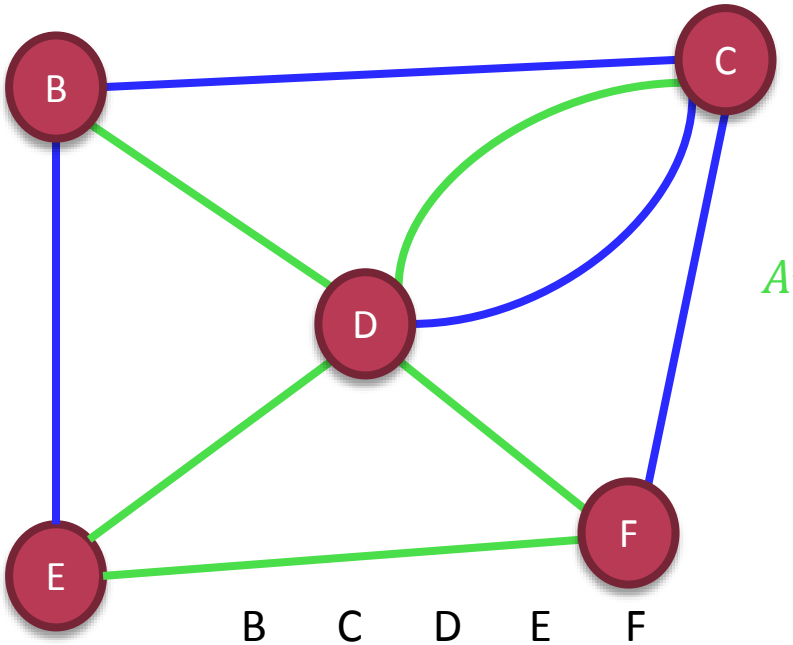
The complexity of graph computations

- Graph computation is difficult
  - The “curse of connectedness”
  - Computer architecture are suited for hierarchical data
- Graph analytics: vertex-centric programming model
  - “Think like a vertex”
  - Pregel, scatter-gather, gather-apply-scatter
- The majority of distributed graph engines use this
- Going distributed for a 2M node graph?

V. Kalavri et al. @ TKDE 2018
High-level programming abstractions for distributed graph processing

LINEAR ALGEBRA-BASED CLUSTERING COEFFICIENTS

Adjacency matrices



$$A = \begin{array}{c} \\ \mathrm{B} \\ \mathrm{C} \\ \mathrm{D} \\ \mathrm{E} \\ \mathrm{F} \end{array} \begin{array}{c} \begin{array}{ccccc} \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{E} & \mathrm{F} \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

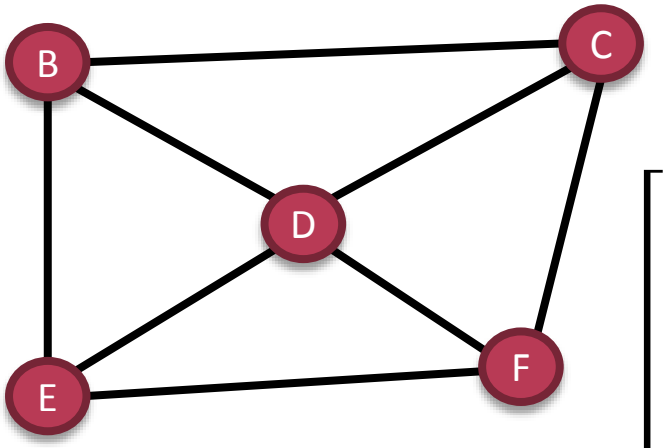
$$A_{\text{business}} = \begin{array}{c} \\ \mathrm{B} \\ \mathrm{C} \\ \mathrm{D} \\ \mathrm{E} \\ \mathrm{F} \end{array} \begin{array}{c} \begin{array}{ccccc} \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{E} & \mathrm{F} \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

$$A_{\text{friends}} = \begin{array}{c} \\ \mathrm{B} \\ \mathrm{C} \\ \mathrm{D} \\ \mathrm{E} \\ \mathrm{F} \end{array} \begin{array}{c} \begin{array}{ccccc} \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{E} & \mathrm{F} \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & & & \\ 0 & 1 & & & \end{bmatrix} \end{array}$$

Optimization #1

Key idea: Multiplication of adjacency matrices = 2-hop, 3-hop, etc. paths

Triangles – untyped



$$\mathrm{diag}^{-1}(A \cdot A \cdot A)$$

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

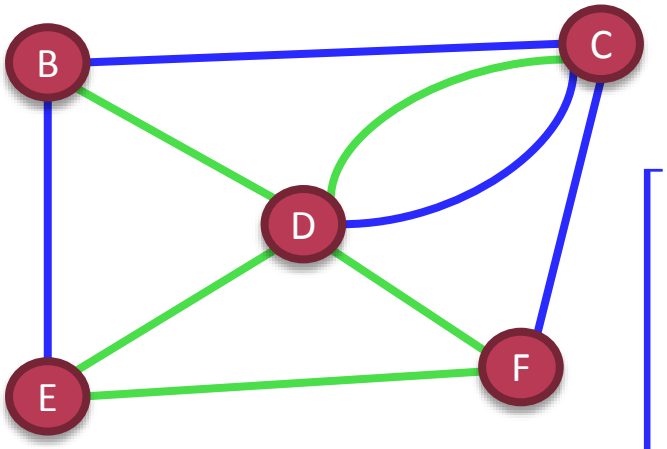
$$\begin{bmatrix} 3 & 1 & 2 & 1 & 3 \\ 1 & 3 & 2 & 3 & 2 \\ 2 & 2 & 4 & 2 & 2 \\ 1 & 3 & 2 & 3 & 1 \\ 3 & 1 & 2 & 1 & 3 \end{bmatrix} \quad \begin{bmatrix} 4 & 8 & 8 & 8 & 4 \\ 8 & 4 & 8 & 4 & 8 \\ 8 & 8 & 8 & 8 & 8 \\ 8 & 4 & 8 & 4 & 8 \\ 4 & 8 & 8 & 8 & 4 \end{bmatrix} \quad \begin{bmatrix} 4 \\ 4 \\ 8 \\ 4 \\ 4 \end{bmatrix}$$

2-hop paths

3-hop paths

Number of triangles for each node

Triangles – typed



$$\operatorname{diag}^{-1}(A_i \cdot A_j \cdot A_i)$$

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 2 & 2 & 1 & 1 & 1 \\ 0 & 2 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 6 & 2 & 2 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 6 \\ 0 \\ 0 \end{bmatrix}$$

Matrices get more dense

Optimization: element-wise multiplication

- $A \cdot A \cdot A$ enumerates all 3-hop paths
  - Matrices get more dense, but only the diagonal is used
- Idea: use element-wise multiplication

Optimization #2

$$\mathrm{LCC}(v) = \mathrm{diag}^{-1}(A \cdot A \cdot A) = A \cdot A \odot A \cdot \vec{1}$$

- Typed clustering: $\mathcal{O}(t^2)$ matrix multiplications

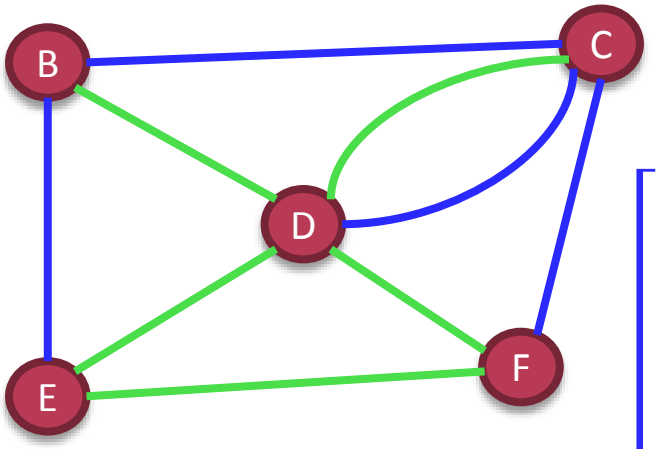
$$\mathrm{TCC}(v) = \frac{\sum_{i \neq j} A_i \cdot A_j \odot A_i \cdot \vec{1}}{(n-1) \cdot \sum_i \left[(A_i \cdot \vec{1}) \odot \left((A_i \cdot \vec{1}) - \vec{1} \right) \right]}$$

$\mathcal{O}(t^2)$ matrix multiplications for each node

P. Várhegyi – Master's thesis (2018)
Multidimensional graph analytics

Embarrassingly parallel –> opt. #3

The example using the optimization



$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \odot \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 2 & 2 & 1 & 1 & 1 \\ 0 & 2 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 6 \\ 0 \\ 0 \end{bmatrix}$$

More sparse matrix

IMPLEMENTATION 1: Java

Question on SoftwareRecs

Sparse matrix library for Java

Ask Question

7

2

I am looking for a sparse matrix library in Java that can do multiplications on sparse integer matrices, where the matrices represent the adjacency relations of a graph. The requirement is roughly the following: the library should be able to load and multiply a few matrices of 10M×10M elements, containing approx. 5M non-zero elements each, when running on a commodity machine (~16 GBs of RAM). The Eigen library for C++ satisfies this requirement. However, I couldn't find a good alternative for Java.

I looked at the following libraries:

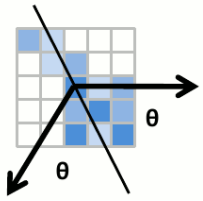
- The `SparseMatrix` class in the Spark ML library only seems to support multiplication with a dense matrix.
  - Digging a bit deeper, the Breeze library used by Spark ML states the following: "CSCMatrices are not fully supported yet. They are missing several basic operations."
  - It's also worth noting that internally, Breeze uses the `netlib-java` library.
- The `OpenMapRealMatrix` class of Apache Commons Math throws a `NumberIsTooLargeException`, as it only supports matrices with 2B elements (*"40,000,000,000 is larger than, or equal to, the maximum (2,147,483,647)"*)
- The `SparseDoubleMatrix2D` class of the Colt library throws a Java heap space error.
- The `DMatrixSparseCSC` class of the Efficient Java Matrix Library ~~throws a `java.lang.NegativeArraySizeException` when initializing a large matrix.~~ This has been fixed since the question -- see the author's comment and the accepted answer.
- The `LinkedSparseMatrix` class of Matrix Toolkit Java is very quick to initialize, but does not handle multiplication well - multiplying an empty 1M×1M matrix takes ~6 minutes. `CompDiagMatrix` runs out of memory for a matrix of this size. Neither `FlexCompColMatrix`, nor `FlexCompRowMatrix` finish in 10 minutes. `CompRowMatrix` and `CompColMatrix` have good performance for ~20k×20k matrices, but break down in performance for larger ones. (The Javadoc for the latest stable version, 1.0.4 does not advise which sparse matrix to use for static cases. A pull request submitted more detailed documentation since, but 1.0.5-SNAPSHOT never made it to release and the project is now archived.)
- Graphulo is an implementation of GraphBLAS, but is very complicated to use as it is designed to run on top of the Apache Accumulo database.
- SuiteSparse is a package of sparse matrix algorithms, implemented in C. While the repository has some Java code, it only accounts to a single class that connects to a SuiteSparse server through HTTP.
- The Univ
- Finally, v

I have also fo
libraries supp

See also the

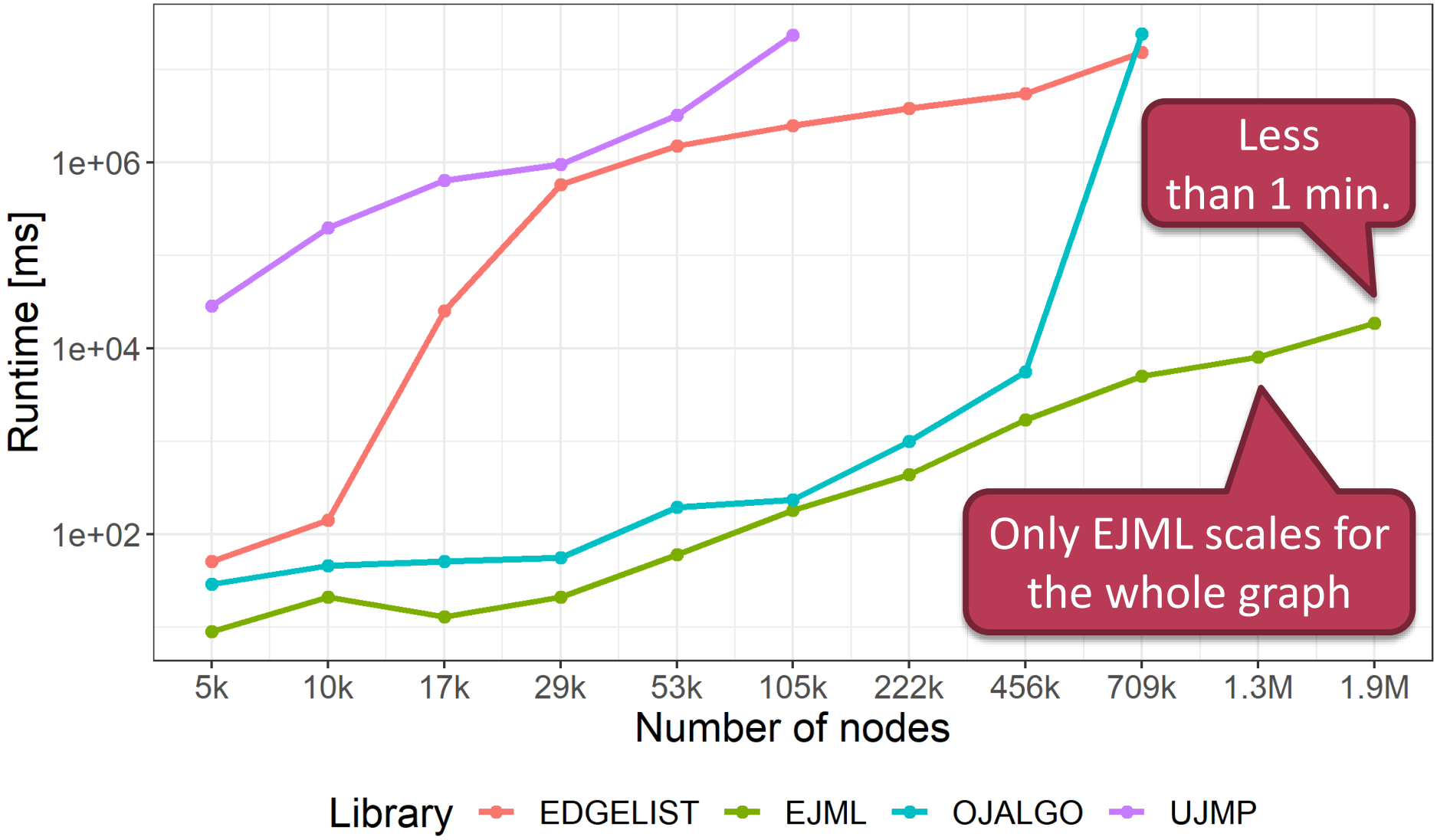
java matrix

EJML



Efficient Java Matrix Library
(2014–ongoing)

TCC on Paradise papers subsets



Graph analyzer library

- Java implementation
  - Linear algebra-based implementations
  - Uses EJML library with CSC compression
  - Single-threaded
  - Runs on top of Neo4j/EMF/CSV graphs
- Number of graph metrics:
  - For nodes, types, node pairs, node-type pairs, …
  - TCC variants, typed degree distribution, degree entropy
  - Pairwise multiplexity, multiplex participation coefficient

ftsrg/graph-analyzer

P. Várhegyi – Master's thesis (2018)
Multidimensional graph analytics

IMPLEMENTATION 2: Julia

Julia language

- A high-performance, high-level dynamic language
- v1.0 last August, v1.1 just out

Search docs

Edit on GitHub

Sparse Arrays

Julia has support for sparse vectors and sparse matrices in the `SparseArrays` stdlib module. Sparse arrays are arrays that contain enough zeros that storing them in a special data structure leads to savings in space and execution time, compared to dense arrays.

Compressed Sparse Column (CSC) Sparse Matrix Storage

In Julia, sparse matrices are stored in the Compressed Sparse Column (CSC) format. Julia sparse matrices have the type `SparseMatrixCSC{Tv,Ti}`, where `Tv` is the type of the stored values, and `Ti` is the integer type for storing column pointers and row indices. The internal representation of `SparseMatrixCSC` is as follows:

```
struct SparseMatrixCSC{Tv,Ti<:Integer} <: AbstractSparseMatrix{Tv,Ti}
    m::Int                  # Number of rows
    n::Int                  # Number of columns
    colptr::Vector{Ti}      # Column i is in colptr[i]:(colptr[i+1]-1)
    rowval::Vector{Ti}      # Row indices of stored values
    nzval::Vector{Tv}       # Stored values, typically nonzeros
end
```

Julia implementation

- Preliminary TCC implementation: ~25 lines
- Written in a few days (incl. learning the language)

$$A \cdot A \odot A \cdot \vec{1}$$

```
A * A .* A * ones(n)
```

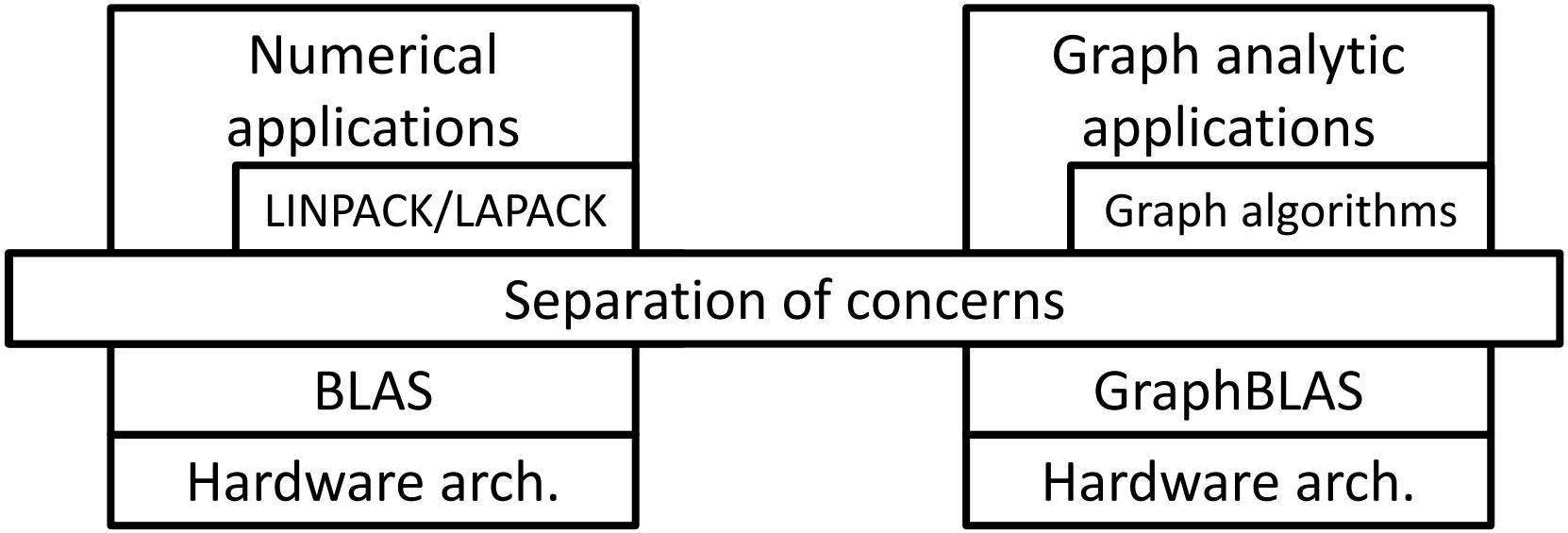
Performance on Paradise papers:

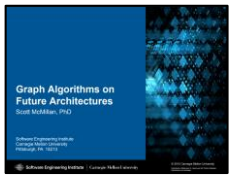
- similar to the best Java implementation
- but easier to write and extend

IMPLEMENTATION 3: GraphBLAS

Approach for defining graph algorithms

- GraphBLAS is an effort to define **standard building blocks for graph algorithms** in the language of linear algebra
- Idea: BLAS (Basic Linear Algebra Subprograms), since 1979





S. McMillan @ SEI Research Review (CMU)
Graph algorithms on future architectures

Graph operations with semirings

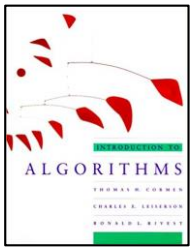
Many graph algorithms can be captured over arbitrary semirings –> requires overloading of $\oplus.\otimes$

Examples:

- real numbers $+ \; . \; \cdot$ clustering
- tropical semiring min.+ shortest path
- boolean semiring $\vee \; . \; \wedge$ traversal

Old idea, but very few libraries support this.

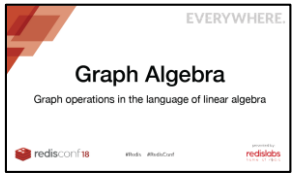
Aho, Hopcrof, Ullman (1974): The Design and Analysis of Computer Algorithms



Cormen, Leiserson, Rivest (1990): Introduction to Algorithms [only in the 1st edition]

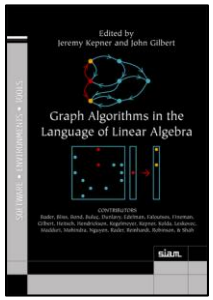
Graph operations with semirings

- SuiteSparse:GraphBLAS
- C API and single-threaded implementation
- Steep learning curve
- Good performance even with a single thread
  - Benchmark work in progress for LCC/TCC

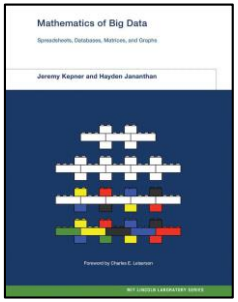


R. Lipman, T. Davis @ redisconf18
Graph algebra: graph operations in the language of linear algebra

sergiud/SuiteSparse/



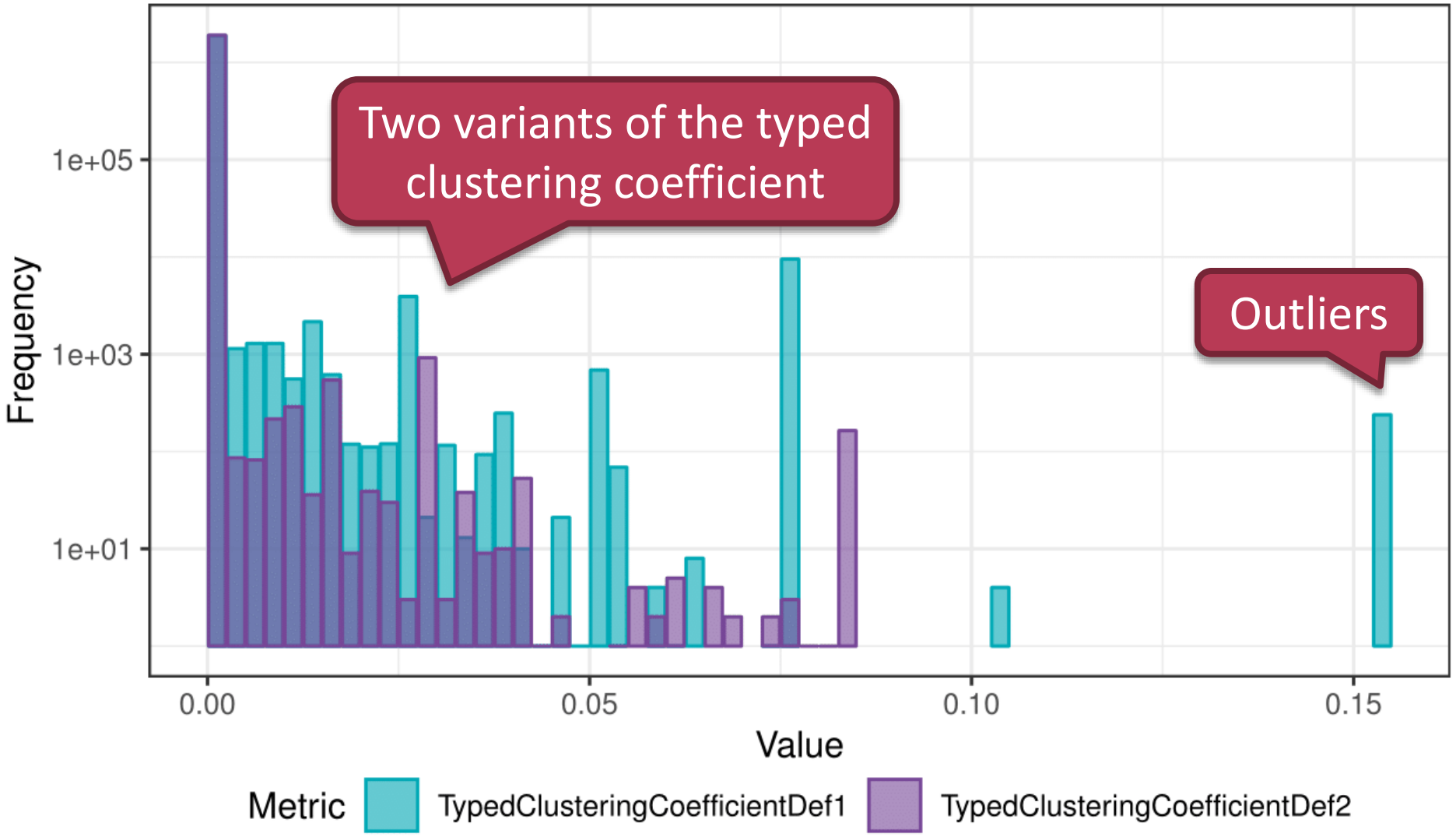
J. Kepner, J. Gilbert,
Graph algorithms in the language of linear algebra.
SIAM, 2011



H. Jananthan, J. Kepner,
Mathematics of Big Data.
MIT Press 2018

RESULTS OF THE ANALYSIS

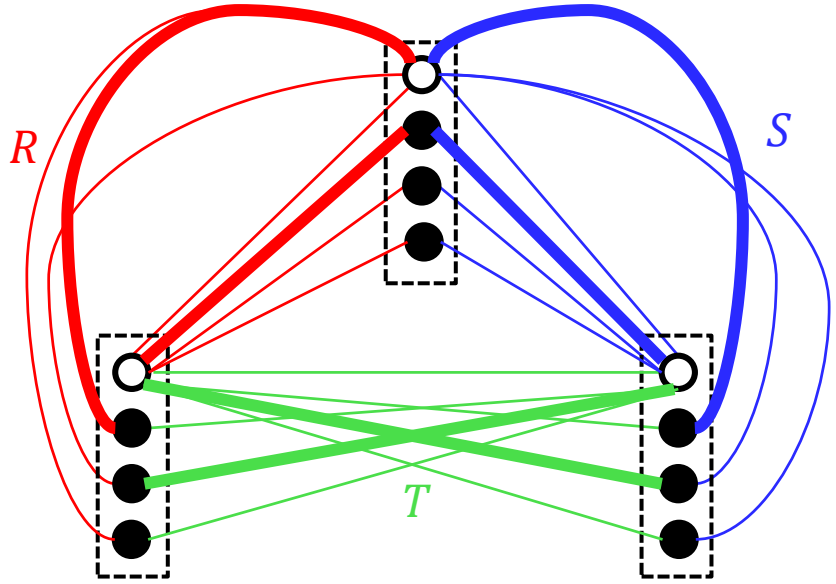
Results on the Paradise papers



Note. Further analysis needs domain-specific expertise.

ALGORITHMIC OPTIMIZATION

Skewed data distribution: joins

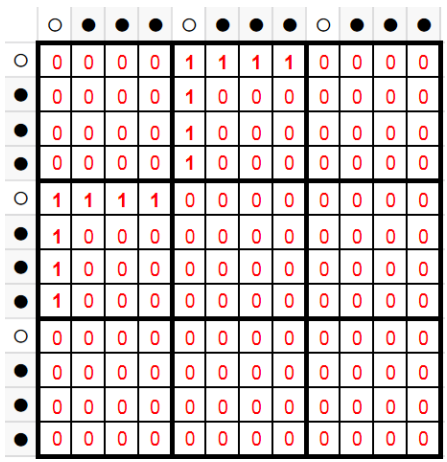


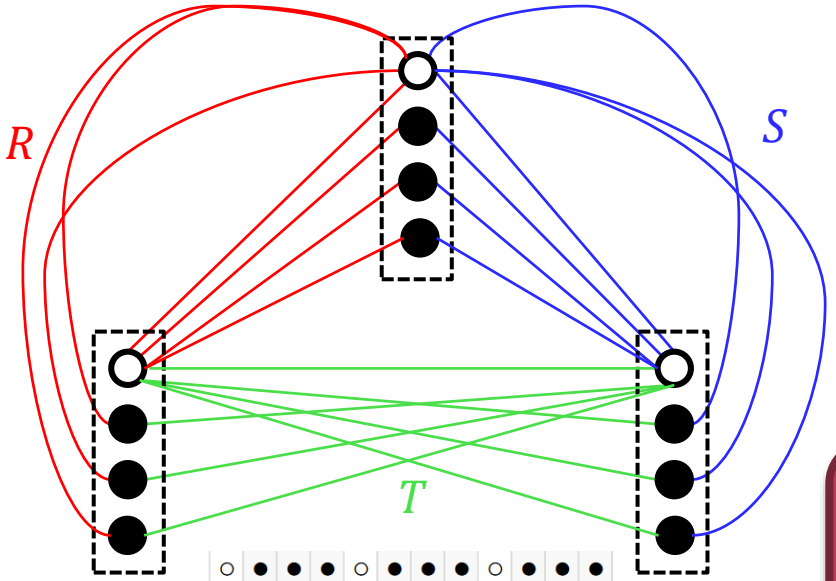
Enumerate all triangles: $R \bowtie S \bowtie T$

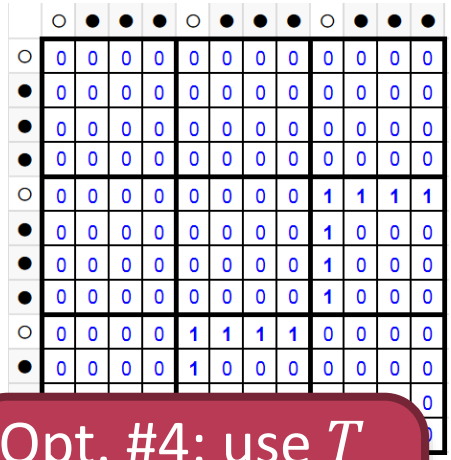
Any solution using binary joins requires $\mathcal{O}(n^2)$ time, but the theoretical lower bound is $\mathcal{O}(n^{1.5})$.

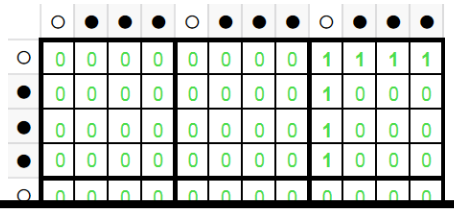
H.Q. Ngo et al. @ SIGMOD Record 2013
Skew strikes back: new developments in the theory of join algorithms.

Skewed data distribution: matrices



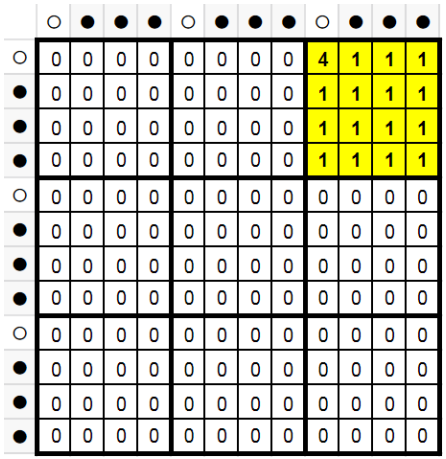






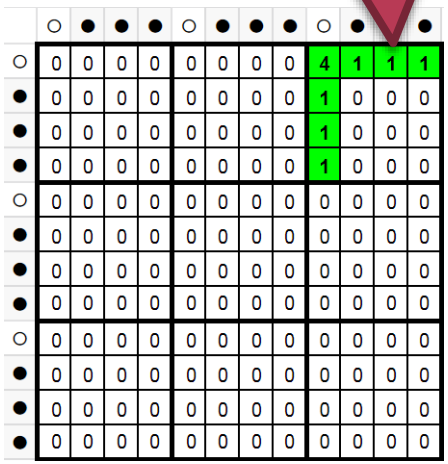
Opt. #4: use T as mask for multiplication

$R \cdot S =$



19 wedges
$\mathcal{O}(n^2)$

$R \cdot S \cdot T =$



10 triangles
$\mathcal{O}(n)$

Skewed data distribution

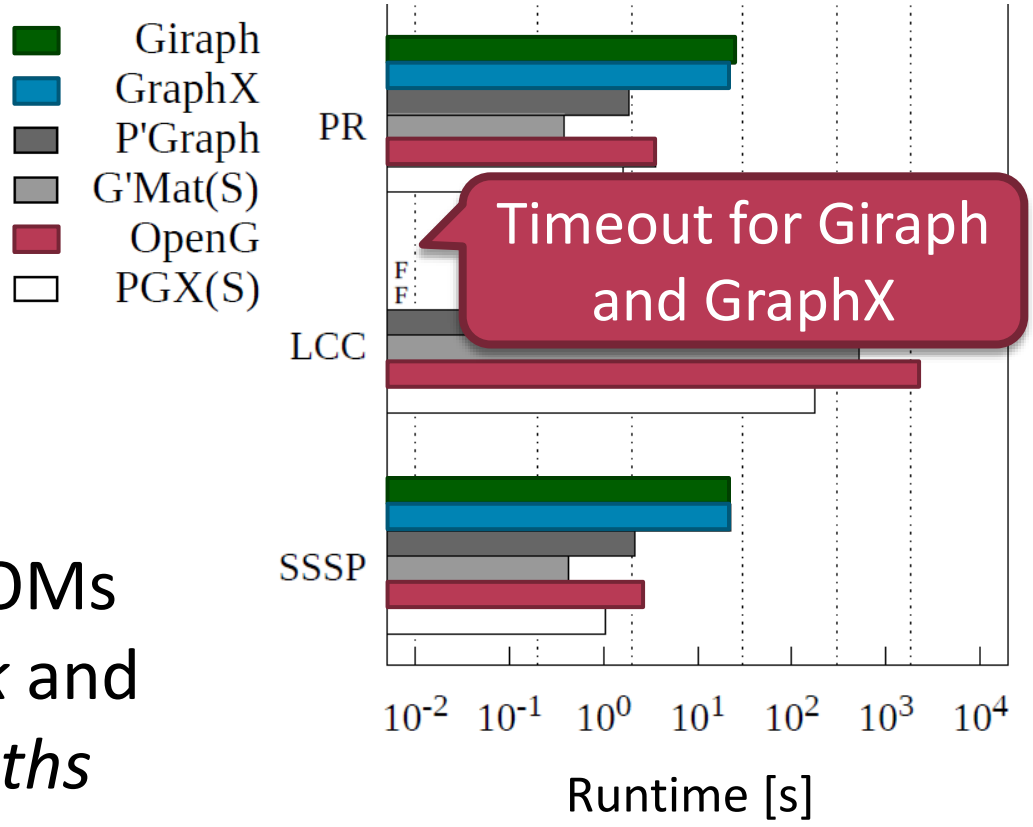
- **Worst-case optimal join algorithms** can compute this example with multiway joins $\bowtie (R, S, T)$ in $\mathcal{O}(n^{1.5})$.
- **Does this occur in practice?** To some degree, due to the power-law distribution of scale-free networks.
- The problem itself is known in graph analytics, e.g. the GraphBLAS API offers masked matrix multiplication.
  - But only supported by libs tailored for graph processing.

H.Q. Ngo @ Journal of the ACM 2018
Worst-case optimal join algorithms

T.M. Low, S. McMillan et al. @ HPEC 2017
First look: linear algebra-based triangle counting without matrix multiplication

Benchmark

- LDBC Graphalytics
- Synthetic social graph
  - 4M nodes/300M edges
  - Single machine



Bottom line:

- *LCC* performances are OOMs worse than for *PageRank* and *single-source shortest paths*
- Slower systems time out

A. Iosup et al. @ VLDB 2016
LDBC Graphalytics

TAKEAWAYS

Summary of requirements for graph proc.

- Graph representation
  - sparse matrices of integers/floats/booleans
  - edge types
- Operation
  - semirings with arbitrary $\oplus$ and $\otimes$ operators
  - parallelization
  - handle skewed distribution
- No high-level off-the-shelf solutions

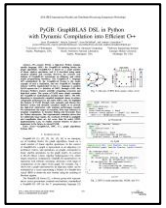
Building blocks for implementations

- C/C++
  - SuiteSparse:GraphBLAS
  - CombBLAS
- Java:
  - Graphulo (GraphBLAS for Apache Accumulo)
  - EJML (Efficient Java Matrix Library)
- Julia
  - SparseArrays with overrides for custom semirings
- Python
  - PyGB (Python wrapper for GraphBLAS)

Papers on linear algebra for graphs

Works on linear algebra-based graph analysis

V.B. Shah et al. @ HPEC 2013
Novel algebras for advanced analytics in Julia.



J. Chamberlin @ GABB 2018
PyGB: GraphBLAS DSL in Python

Y. Ahmad et al. @ VLDB 2018
LA3: A scalable link- and locality-aware linear algebra-based graph analytics system

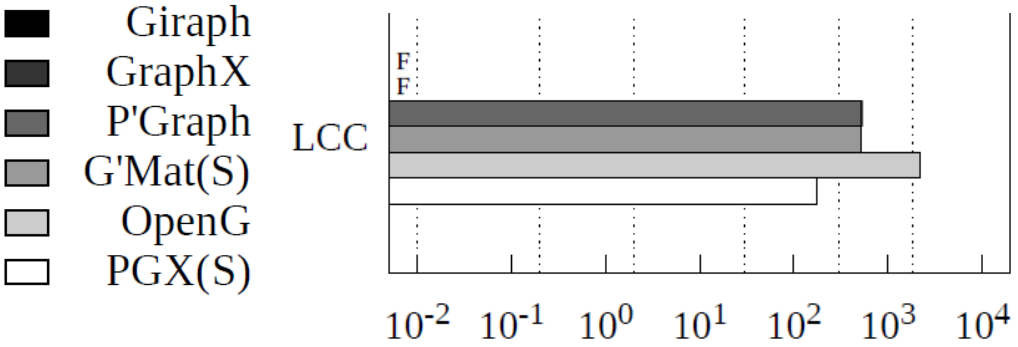
Distributed and LA-based

T. Davis – preprint (2018)
Algorithm 9xx: SuiteSparse:GraphBLAS

Implementations and libraries

- Difficult to find an ideal platform (Hadoop? Spark?)
  - EJML is the best Java lib for sparse matrices
  - GraphBLAS is great but takes some time to learn
  - Julia is promising but has no graph support yet
- Some Java libs could have worked for Paradise papers
  - Arabesque
  - Flink Gelly
  - GPS

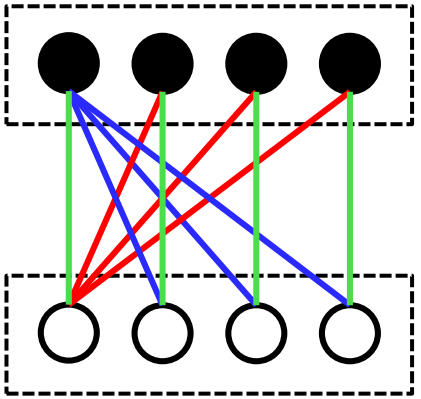
None were included in this benchmark



FUTURE WORK

Future work: typed HO clustering

- A “counterexample” for LCCs
  - bipartite graph
  - no triangles
  - but interesting 4-hop cycles



- Use more sophisticated metrics:
  - Higher order (HO) clustering is a generalization of LCC
  - Meta-paths are paths on certain node/edge types
  - Gets very complex very soon

Fronczak et al. @ Physica A 2002
Higher order clustering coefficients in Barabási–Albert networks

Shi et al. @ TKDE 2017
A survey of heterogeneous information network analysis

Future work: analysis of huge graphs

	Benchmark	Subjects	Predicates	Objects	Triples
Synthetic	Bowlogna [11]	2,151k	39	260k	12M
	TrainB. [30]	3,355k	16	3,357k	41M
	BSBM [8]	9,039k	40	14,966k	100M
				19,347k	49M
				9,753k	108M
				17,544k	46M
Real	FishMa	395k	878	1,148k	10M
	BioBench [33]	278,007k	299	232,041k	1,451M
	FEASIBLE [24]	18,425k	39,672	65,184k	232M
	DBPSB [18]	18,425k	39,672	65,184k	232M

Biological RDF data set:
290M nodes, 800M edges

M. Saleem, G. Szárnyas et al. @ WWW 2019
How representative is a SPARQL benchmark? An analysis of RDF triplestore benchmarks.

Acknowledgements

This research was partially supported by the MTA-BME Lendület Cyber-Physical Systems Research Group and the ÚNKP-18-3 New National Excellence Program of the Ministry of Human Capacities.

Hungarian Academy of Sciences

MTA-BME Lendület
Cyber-Physical Systems Research Group

Department of Measurement and Information Systems

MINISTRY OF HUMAN CAPACITIES