GraphAlg: Algorithm Support in a Graph Database, Done Right

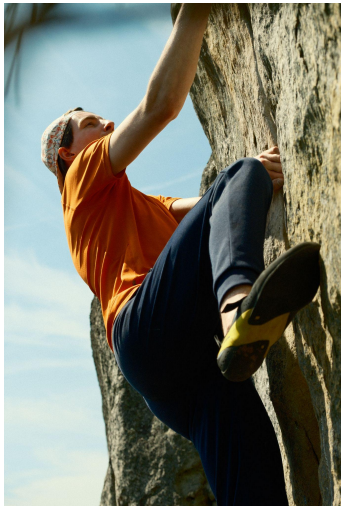
Daan de Graaf, Robert Brijder, Soham Chakraborty*, George Fletcher, Bram van de Wall, Nikolay Yakovets

* TU Delft

others TU Eindhoven

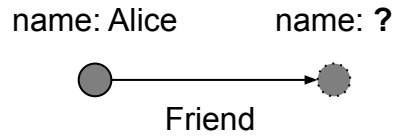
About Me

- 1st year PhD @TU Eindhoven (Database Group)
- Current Research: Algorithm support in GDBs
- Research Interests:
  - Query Processing
  - Hardware Acceleration
  - Compilers

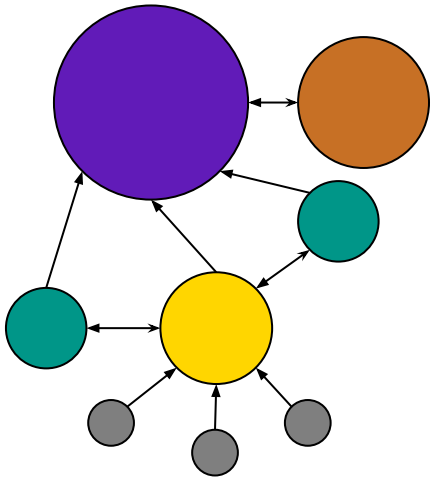


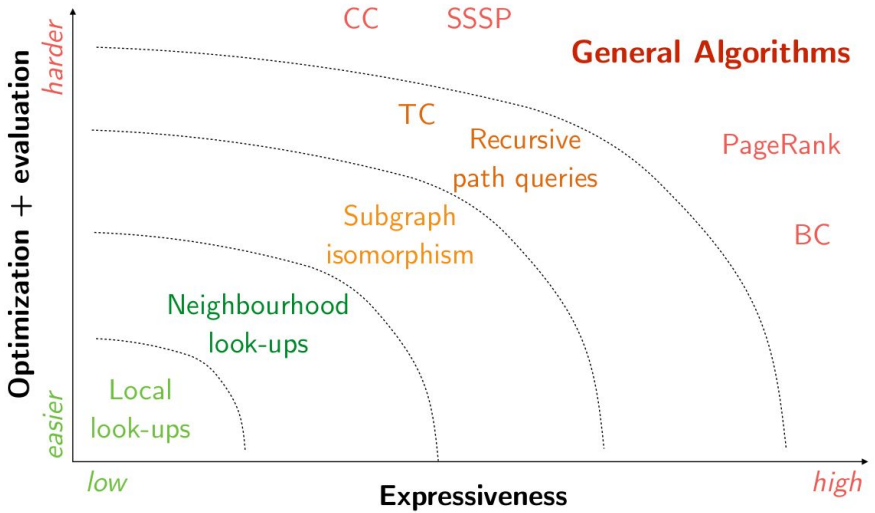
Types of Graph Queries

Local look-up



PageRank





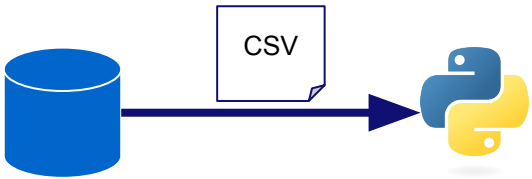
Limitations of Graph Query Languages

Graph query languages are great for simple queries

… but **lack expressive power** for Graph Analytics

Users **waste resources** and **complicate workflows**

by processing in external tools



Traditional Approaches to Analytics in Databases

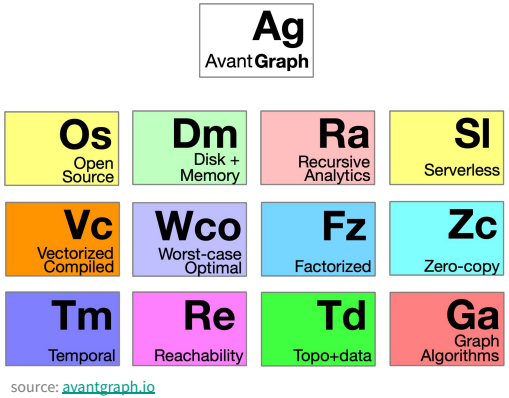
Approach	Key Problems	Available in
Built-in Algorithms Library	- Fixed set of Algorithms	neo4j, ArangoDB
Pregel API	- Performance issues - Not analysable	
User-defined operators	- Unsafe - Not analysable	UMBRA
Recursive CTE	- Difficult to write - Performance issues	
Procedural SQL	- Overhead - Limited analysis	ORACLE
Algorithm DSL	- Proprietary - No integration with queries	Oracle Labs PGX

What Should Algorithm Support Look Like?

- **Flexible:** Fully custom algorithms
- **User-friendly:** Clear and convenient syntax
- **Fully integrated:** Native support in the database
- **Optimizable:** Efficiently process large graphs

Introducing GraphAlg

- A language built for graph algorithms
- Fully integrated into **AvantGraph**[1]
- Highly optimizable
- Embed algorithms into queries



source: avantgraph.io

Cypher

GraphAlg

```
WITH ALGORITHM "
func TriangleCount(graph: Matrix<s1, s1, bool>) -> int {
   L = tril(graph);
   U = triu(graph);
   C = Matrix<int>(graph.nrows, graph.ncols);
   C<L> = cast<int>(L) * cast<int>(U.T);
   return reduce(C);
}"
CALL TriangleCount()
RETURN count
```

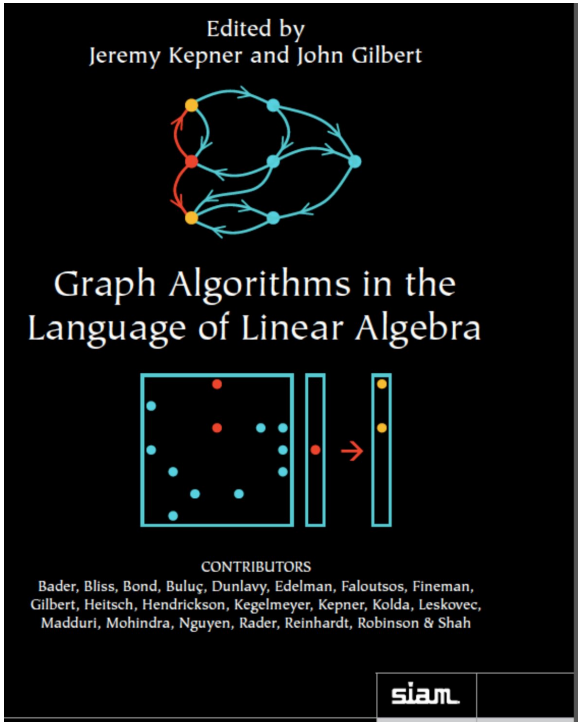
[1] v. Leeuwen, et al. 2022. AvantGraph query processing engine. Proc. VLDB Endow. 15, 12

Computational Model

- Vertex-centric (Pregel)
- Vertex/Edge sets
- **Linear Algebra**
  - High-level operations that are easily parallelized
  - Semantics are well-defined and widely taught
  - Proven efficient for graph analytics, see **GraphBLAS**.

GRAPHBLAS

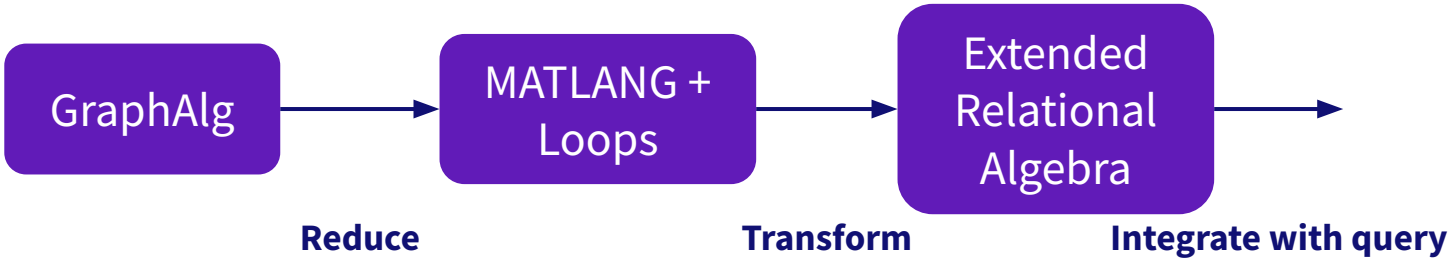
source: GraphBLAS Forum



source: SIAM

Powerful Language, Small Core

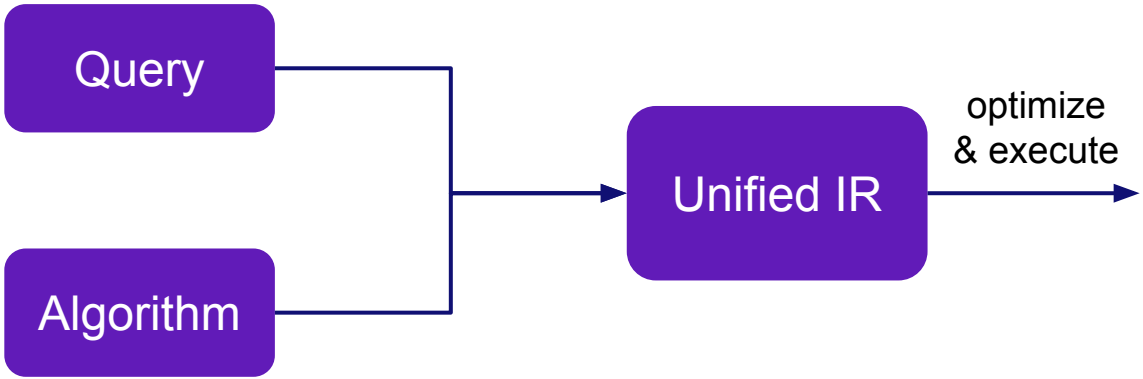
- Reducible to *core language*, **without loss of expressivity**
- Equivalent to **MATLANG**[1]
- Loop construct **balancing expressivity & optimizability**



[1] Brijder et al. 2019. On the Expressive Power of Query Languages for Matrices. ACM Trans. Database Syst.

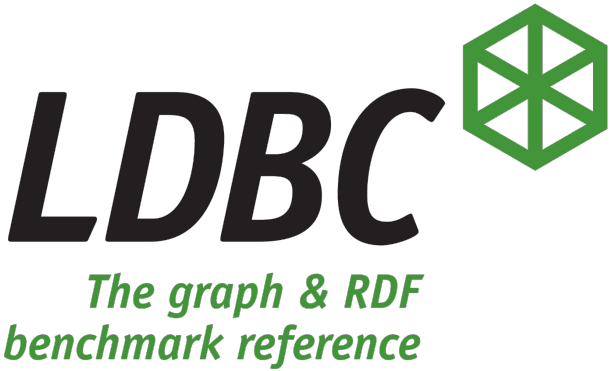
Cross-Optimization

- **Unified IR** for query and algorithm
- Eliminate query/algorithm interface boundary
- Holistic optimization & execution



Benchmark: LDBC Graphalytics

- Expressivity: Support **all algorithms** in Graphalytics spec.
- Performance: Use Graphalytics synthetic and real-world datasets, comparing:
  - Reference implementations
  - DuckDB (Python API)
  - Neo4J (Pregel API)



Implications

- A significant jump of programmability for graph databases
- Blurring the line with graph analytics frameworks
- Graph databases as a platform for large-scale data analysis

Thanks!

Questions?