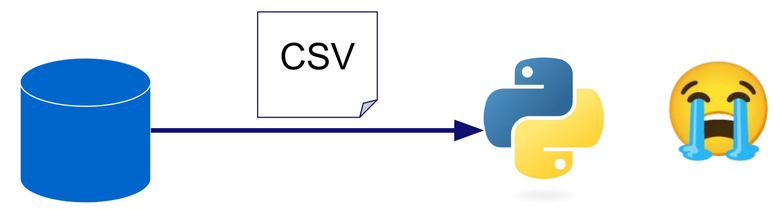
GraphAlg: Efficient Execution of User-Provided Graph Algorithms in a Graph Database

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

GraphAlg is a **domain-specific language** for high-performance **graph analytics**. Designed to be **embedded in the AvantGraph database**, it is more **expressive** than traditional query languages while remaining highly **amenable to optimization**.

Problem: Low Programmability

If your algorithm cannot be expressed in the query language (e.g., Cypher), the graph database quickly becomes a **dumb storage layer. Exporting large volumes of data is expensive** in terms of compute and storage, and in the process all **data statistics are lost**.




```
CALL export.csv.all("database.csv")
```

Low programmability forces users to export data and use external tools.

Solution: GraphAlg

A language for writing graph algorithms, designed to be integrated into graph databases.

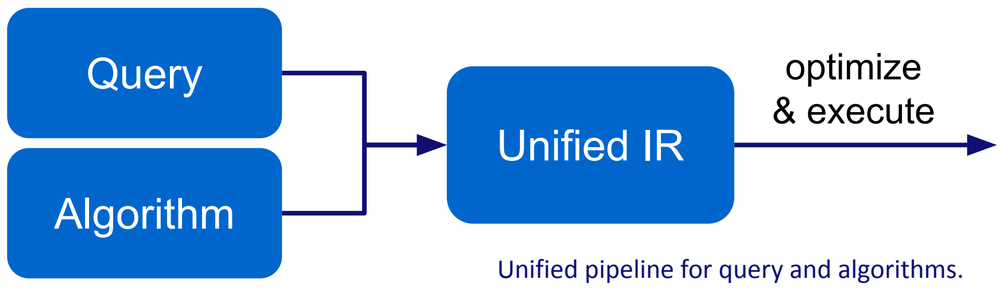
- **Fully integrated with AvantGraph**. Embed algorithms into Cypher queries.
- **Eliminates data wrangling** by operating inside of the database.
- **Purpose-built for graph algorithms**. Based on linear algebra, it can concisely express a wide variety of algorithms.
- **Highly optimizable**. A small, high-level core language with formal semantics.

```
func SSSP(graph: Matrix<s, s, trop_real>,
          source: Vector<s, bool>)
          -> Vector<s, trop_real> {
  v = cast<trop_real>(source);
  for i in graph.nrows {
    v += v * graph;
  }
  return v;
}
```

Single-Source Shortest Paths in GraphAlg.

Cross-Optimization

AvantGraph has **full visibility into GraphAlg programs**. The query and any embedded algorithms are transformed into a unified IR that is **holistically optimized and executed**, enabling optimizations that cross the border between query and algorithm.



Unified pipeline for query and algorithms.

Core Language

GraphAlg can be reduced to a small *core* language with well-defined **operational semantics**. The core language is **equivalent to MATLANG**[1] with a limited form of iteration. Our **novel loop construct** is expressive enough to support efficient implementations of commonly used algorithms, yet limited enough that it remains amenable to analysis and optimization.

GraphAlg compiler pipeline.

Approach	Key Problems	Used by
Built-in Algorithms Library	- Fixed set of Algorithms	neo4j
Pregel API	- Performance issues - Not analysable	ArangoDB
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

Different approaches to graph analytics in databases.

[1] Robert Brijder, Floris Geerts, Jan Van Den Bussche, and Timmy Weerwag. 2019. On the Expressive Power of Query Languages for Matrices. ACM Trans. Database Syst.

Database Group

Funded by the European Union
Supporting eosc

This project has received funding from the European Union's Horizon Europe framework programme under grant agreement No. 101058573. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Executive Agency. Neither the European Union nor the European Research Executive Agency can be held responsible for them.