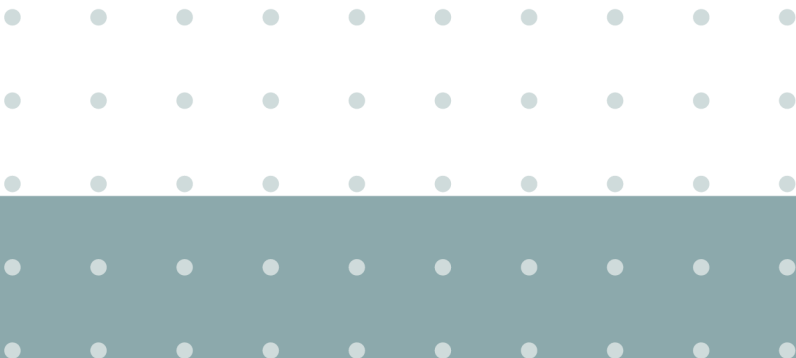




BAOLITE

An ML-Inspired Join Operator Heuristic
for PostgreSQL

Iqra Irfan
Steven Xie
Sajid Shaikh



1. INTRODUCTION

2. Abstract

3. Methodology

4. Advantages

5. Results

6. Q/A



The Problem Statement

PostgreSQL's cost model select suboptimal joins because :

- **Doesn't exploit stable workload patterns**
- **Can't adapt from prior executions**
- **Misses opportunities for better I/O locality**



Introduction

BaoLite is a PostgreSQL extension that injects lightweight, machine-learning-inspired heuristics into the join planning phase to improve operator selection without modifying the core optimizer or changing join order.

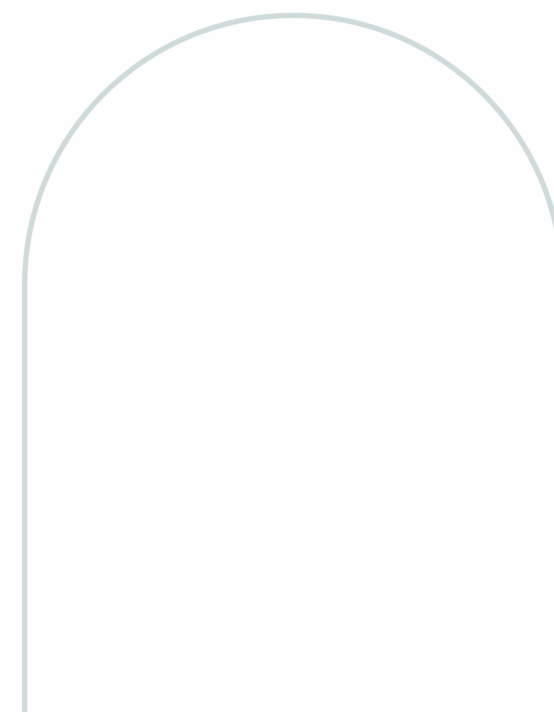


Abstract

Instead of embedding a heavy neural model, BaoLite trains a surrogate cost model offline on thousands of EXPLAIN plans, categorizes tables as small, mixed, or large, and encodes these categories into a simple rule that biases PostgreSQL toward Nested Loop or Hash Join where appropriate. The extension is implemented as a planner hook that dynamically adjusts join-related GUCs per join, allowing PostgreSQL's native cost-based optimizer to retain full control and override hints when they are unsafe, while still achieving speedups of up to roughly 28× on real StackExchange queries by reducing disk reads and improving buffer locality.

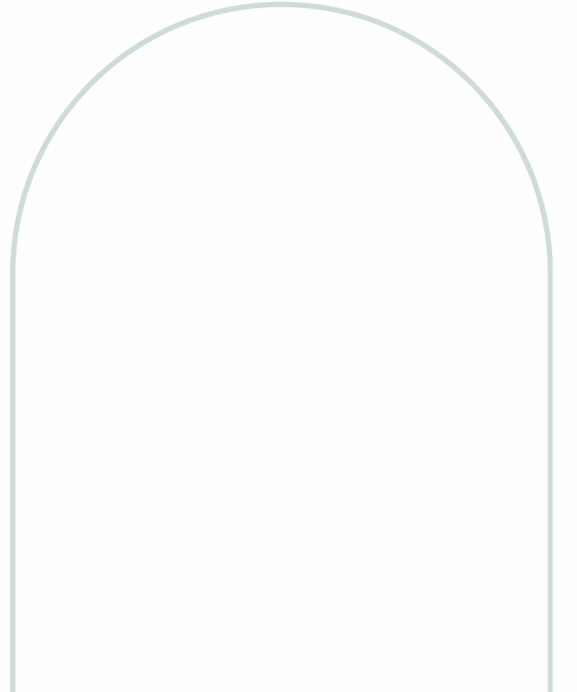


Methodology



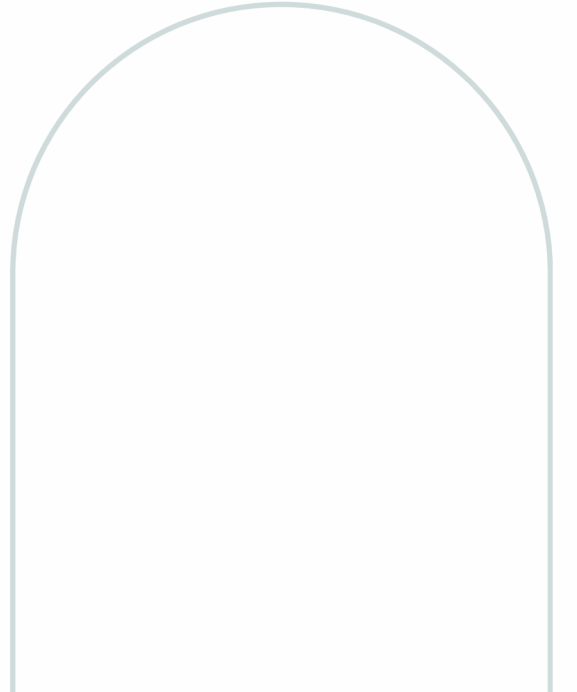


1. Plan Data Extraction & Feature Engineering

- **Generate thousands of join queries (StackExchange)**
 - **Run EXPLAIN (JSON)**
 - **Python parser → operator tree, table IDs, sizes, join type, est/actual rows, costs**
 - **Output: normalized feature CSV**
- 



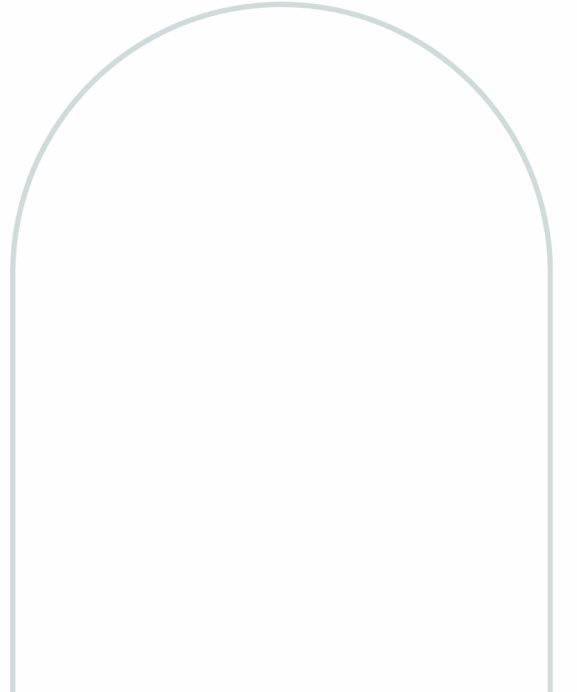
2. Surrogate Cost Model Training

- **Input: feature CSV (sizes, join type, cardinalities, operator flags)**
 - **Train small neural model to predict join cost**
 - **Track MSE, validation metrics, R^2 , convergence**
 - **Use DQN-style exploration to probe alternative join decisions**
- 



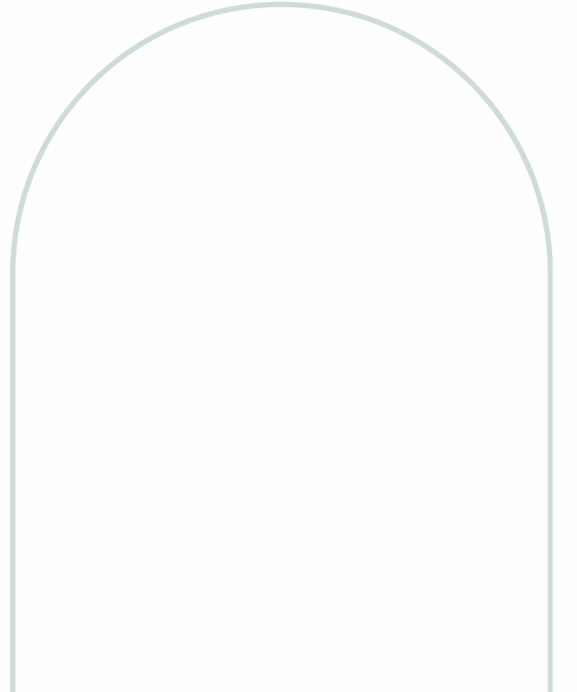
3. ML-derived table categorization

Compress model behavior into 3 categories per table:"

- **Small** → Nested Loop efficient
 - **Large** → Hash Join efficient
 - **Mixed** → generally Hash Join
 - **Materialize as mapping file** (e.g., vocab.json: rel OID → {small|mixed|large})
- 

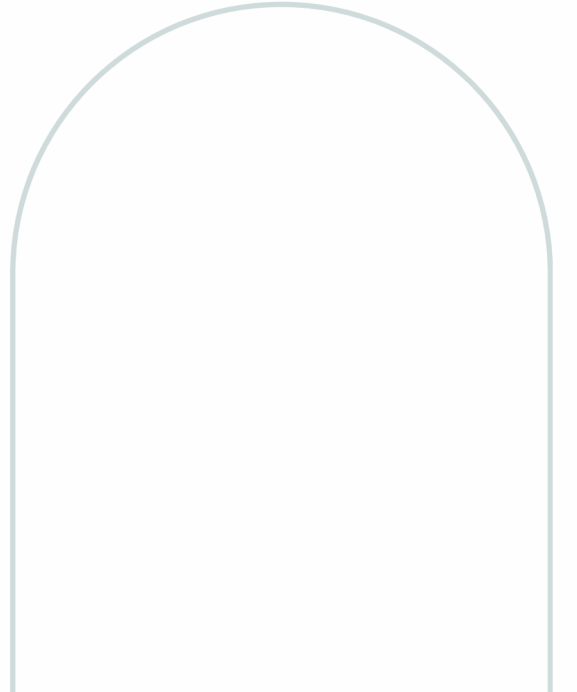


4. Heuristic design and rule encoding

- **Generate thousands of join queries (StackExchange)**
 - **Run EXPLAIN (JSON)**
 - **Python parser → operator tree, table IDs, sizes, join type, est/actual rows, costs**
 - **Output: normalized feature CSV**
- 

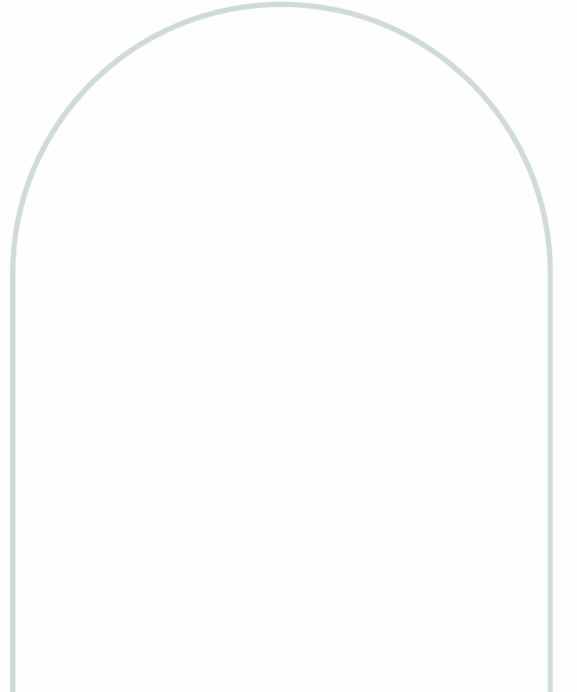


5. PostgreSQL integration via planner hook

- **Generate thousands of join queries (StackExchange)**
 - **Run EXPLAIN (JSON)**
 - **Python parser → operator tree, table IDs, sizes, join type, est/actual rows, costs**
 - **Output: normalized feature CSV**
- 

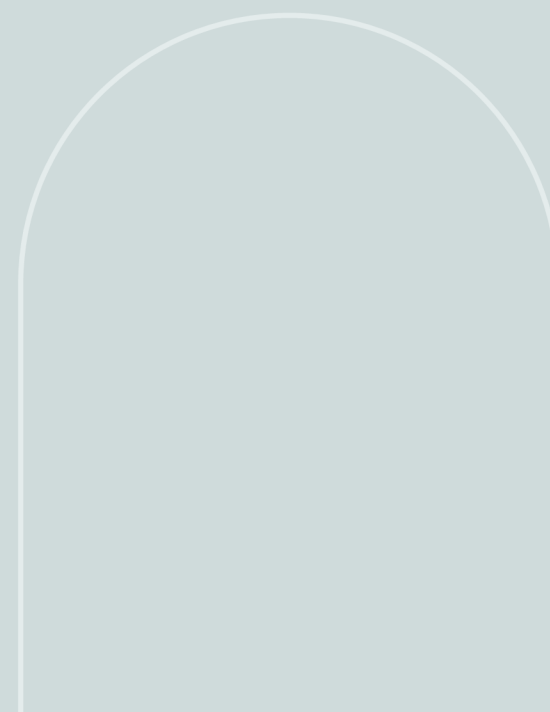


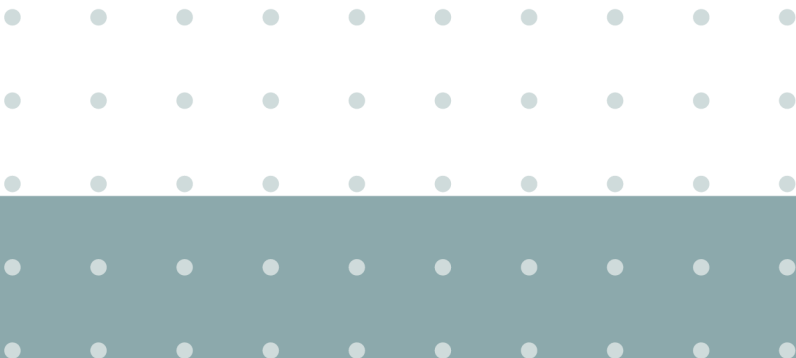
6. Experimental Evaluation Design

- **Generate thousands of join queries (StackExchange)**
 - **Run EXPLAIN (JSON)**
 - **Python parser → operator tree, table IDs, sizes, join type, est/actual rows, costs**
 - **Output: normalized feature CSV**
- 



RESULTS





BAOLITE

An ML-Inspired Join Operator Heuristic
for PostgreSQL

Iqra Irfan
Steven Xie
Sajid Shaikh

