An Incremental Anytime Algorithm for Multi-Objective Query Optimization

Immanuel Trummer and Christoph Koch
{immanuel.trummer, christoph.koch}@epfl.ch

Context

In multi-objective query optimization, we search the query plan that represents the best tradeoff between different cost metrics such as execution time, monetary fees, energy consumption, or result precision.

The best tradeoff is defined by user preferences. Prior approaches let users formalize their preferences before optimization starts. This is however tedious and error-prone.

Goals

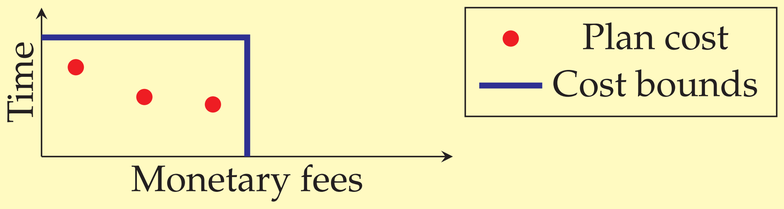
Users do not specify their preferences beforehand but rather pick their preferred cost tradeoff from a visualization of available tradeoffs.

Query optimization feels similar to using a hotel booking Web site: Users dynamically adapt their constraints based on a continuously refining visualization of optimal cost tradeoffs.

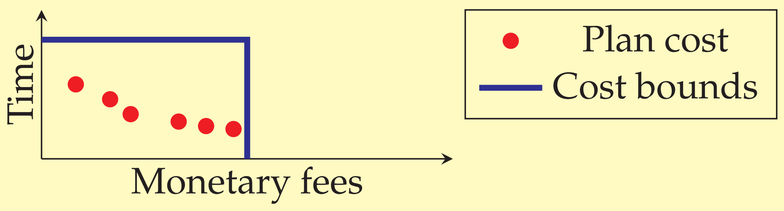
Example Session

We execute query plans in the Cloud and compare plans based on their execution time and their monetary execution fees.

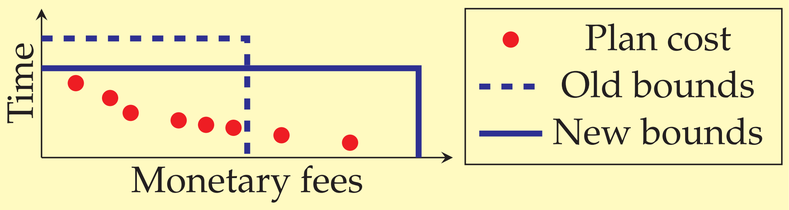
After entering a query, the user obtains quickly a coarse-grained approximation of optimal cost tradeoffs within the default cost bounds:



Without user interaction, the approximation is continuously refined:



The user may adapt the bounds at any time and the visualization is quickly updated:



Optimization ends once the user selects his preferred cost tradeoff.

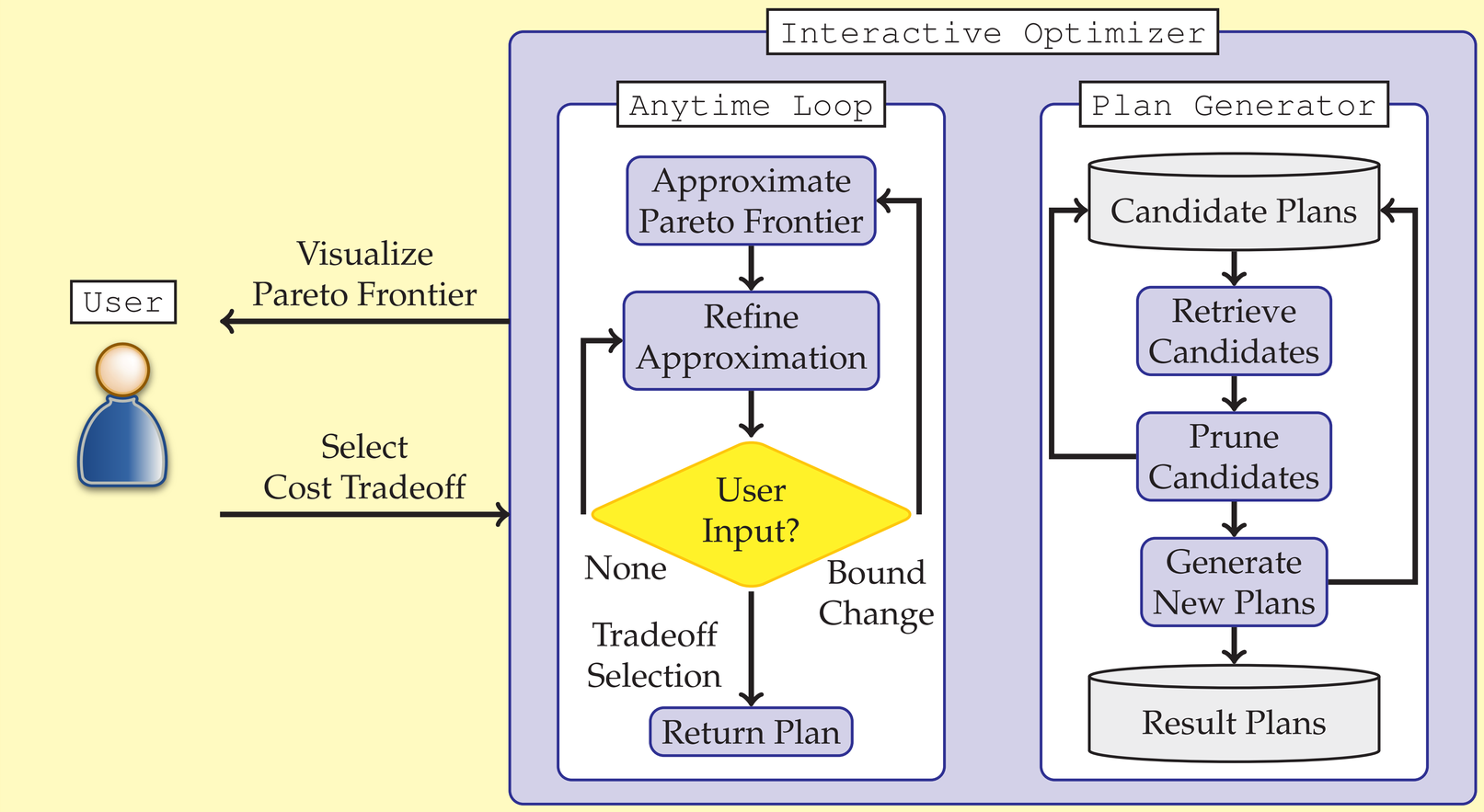
Anytime Property

Calculating a fine-grained approximation of the Pareto frontier takes a lot of time. To allow for a responsive user interface, our optimizer calculates a series of approximations with increasing resolution. We call this the anytime property.

Incrementality

The optimizer is invoked many times for the same query, for different resolution levels and cost bounds. We must avoid regenerating the same query plans over and over again. Our optimizer is incremental: it generates plans only once and discards them only if they cannot be relevant for future optimizer invocations.

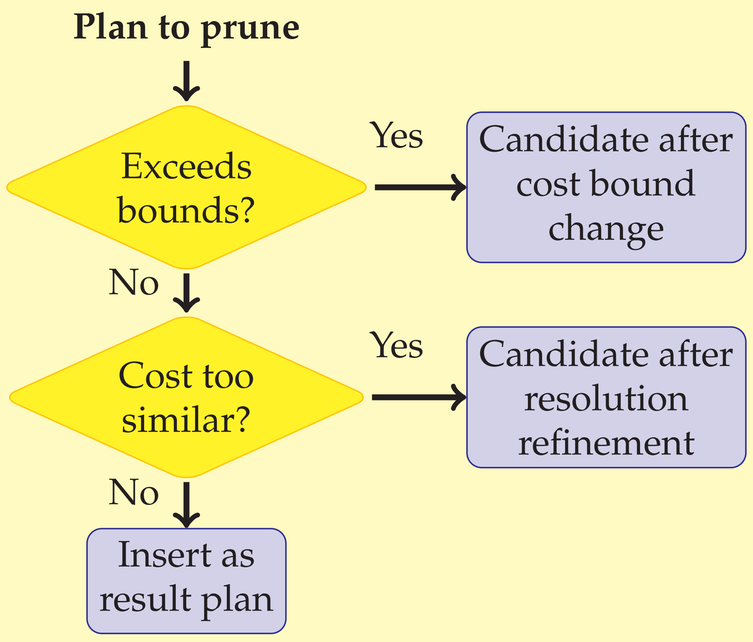
Overview of Approach



Explanation. After entering a query, the USER sees a continuously refining approximation of the query plan Pareto frontier. The user can set cost bounds to focus optimization on interesting segments of the Pareto frontier. Finally, the user selects the preferred cost tradeoff from the Pareto frontier.

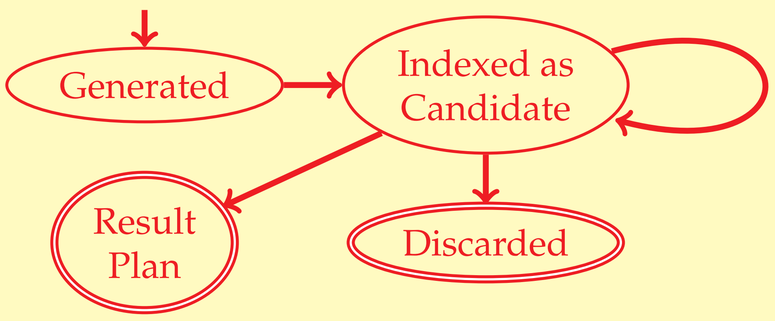
Internally, the ANYTIME LOOP controls the interaction with the user and the resolution refinements. It uses the PLAN GENERATOR as a subfunction to refine the plan Pareto frontier approximation by generating new plans. The plan generator is incremental and indexes plans as candidates that might become useful in the future.

Pruning



Explanation. We first check whether the pruned plan exceeds the current cost bounds. If yes then it is indexed as candidate to reconsider once the cost bounds change. If no then we check whether the cost of another plan is too similar to the pruned plan. If yes then we index the pruned plan as candidate to reconsider once the resolution is refined. If no then the pruned plan is inserted as result plan.

Plan Life Cycle



Explanation. After generation, plans are potentially re-indexed multiple times as candidates before either being discarded or inserted.

Experimental Setup

Goal. We compare our incremental anytime algorithm against baselines on TPC-H queries.

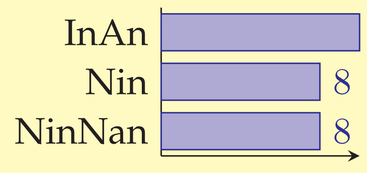
Metric. We measure optimization time when generating a sequence of Pareto frontiers with increasing resolution.

Algorithms. InAn is our incremental anytime algorithm. Nin is a non-incremental algorithm. NinNan is a non-incremental non-anytime algorithm.

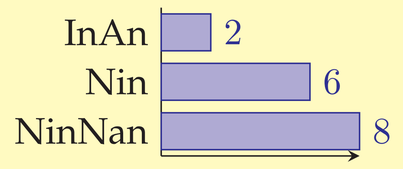
Implementation. All algorithms implemented as extensions of the Postgres optimizer.

Experimental Results (Extract)

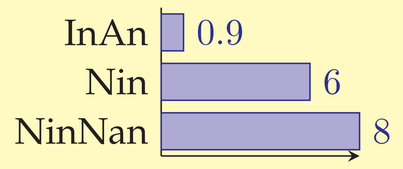
We vary the number of resolution levels and measure optimization time in seconds per invocation for three objectives and five joins:



1 Resolution Level
InAn $1.2\times$ slower than baselines.

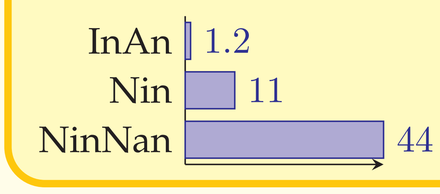


5 Resolution Levels
InAn $3\times$ faster than baselines.



20 Resolution Levels
InAn $7\times$ faster than baselines.

Only InAn can decompose optimization into incremental steps. The performance gap between InAn and the baselines grows for higher target resolutions:



20 Resolution Levels
InAn $9\times$ faster than baselines.