
Strengthening the A* Heuristic of the ITB Query Planner

Krish Patel

Student ID: 21261858 WAT ID: k587pate

Email: kmp.patel@uwaterloo.ca

CS 848: Modern Database Systems, University of Waterloo

Abstract

Before ITB can run a query it must find a plan, and its planner searches for the cheapest one with A*. How quickly that search ends depends on the heuristic, the estimate of the cost a partial plan still needs, and ITB's own list of open problems asks for a stronger one. This project designs a heuristic from two terms, one for the quantification work a fragment still needs and one that ties a fragment to the closing sets it can still complete, argues when it is admissible, and evaluates it on the test suites that ship with ITB. Because ITB accepts a plan when it is generated rather than when it is expanded, the evaluation also runs every estimate under the textbook goal test.

1 Goal

ITB compiles a relational query by interpolation: a tableau prover computes closing sets, and a planner searches with A* for the cheapest fragment that meets the closing condition, which makes it an interpolant and so a plan (Hudek et al. 2015; Toman and Weddell 2018). I study the heuristic that guides this search.

A fragment is a partial plan built from atoms of the closing sets. The planner expands fragments in order of $f = g + h$, where g is the fragment's cost under the cost model and h estimates the cost still missing. Textbook A* returns the cheapest plan when h is admissible, never overestimating, because it tests for the goal when a fragment is expanded (Hart et al. 1968). ITB tests when a fragment is generated: a new plan cheaper than the best so far gets priority 0 and is returned at the next step (planner/flist.c:354–359), so in ITB admissibility alone does not make the first plan the cheapest. The current h is computed from the closing-set atoms a fragment has not yet covered (Feng 2023, Sec. 5.1) and, in the code, falls back to a flat constant once none is missing. Search effort is measured in A* iterations and fragments accepted, and plan quality in plan cost.

Q1: informedness. Can h rank fragments by the work they still need, so that A* needs fewer iterations with no plan lost and none made dearer? My method adds two terms. The quantification term h_Q replaces the flat constant by ε , the cost of one quantifier in g , times the number of variables the fragment must still quantify, and is added to every fragment, so that fragments missing the same atoms are ranked by the quantification still ahead of them. Quantifying a variable then raises g by ε and lowers h by ε , so f does not change. The closing-set term h_C follows Prof. Toman's suggestion: it charges a fragment against the closing sets consistent with the atoms it already covers, instead of a fresh minimum over all of them, tracked per position so that the goal test $h = 0$ stays exact.

Q2: admissibility. Can the stronger h stay admissible under ITB's naive cost model, and cheap enough to compute at every step? Each variable a fragment must still quantify needs its own quantifier, so h_Q is a lower bound on the cost still to be paid. I will argue that its sum with the atom term is admissible, or find where it is not, and test the argument on every returned plan (E4 in Table 2).

2 Prior Work and What Is Open

Hudek et al. (2015) enumerate query plans from analytic tableau proofs, and ITB implements this with a planner that searches fragments with A* (Toman and Weddell 2018); Feng (2023, Sec. 5.1) describes its closing-set heuristic but does not analyse the heuristic’s properties. Benedikt et al. (2014) also derive low-cost plans from proofs, under access restrictions. Outside interpolation, Haffner and Dittrich (2023) cast join ordering as shortest-path search: an admissible heuristic keeps A* optimal, an informed but inadmissible one built from a greedy completion finds plans faster but may miss the optimum, and, unless $P = NP$, no optimal heuristic is computable in polynomial time, so informativeness has to be traded against cost per step. ITB’s list of open problems asks for a stronger heuristic, noting that a weak one drives the search toward breadth-first order, and Prof. Toman has described the current heuristic as weak and suggested tying each fragment to particular closing sets (meetings, August and September 2026). My thesis work has measured that the current heuristic separates fragments mainly by whether any atom is missing, that its flat constant makes it inadmissible on a constructed input, and that ordering by g alone costs up to 33 times its iterations on 32 inputs. Still open are a heuristic for ITB’s fragment search that separates fragments by their distance from a plan, and a proof that it is admissible.

3 Feasibility: What Is in Place

Setup. ITB’s `reorg` branch builds with `clang` and `SWI-Prolog` under `WSL2` on my laptop and runs through the sanctioned pipeline (`transwip`, `itb`, `ppclosingset`, `plan`). From my thesis work I have a regression harness that records each input’s plan text as a golden file, so a change that alters any plan is caught even when the number of plans found stays the same.

Where the change goes. The planner reaches the heuristic through a cost-model table, and the naive model computes it in one file, `naive_cost_model.c`; the queue priority is set in one place, `planner.c:343`. Both terms therefore go behind compile-time flags in a scratch copy of ITB, and the ITB source itself is not modified. The new estimate replaces h only in the queue priority, so goal detection, eligibility and pruning keep the current h in every arm and any change in output comes from search order. A further flag moves the goal test: `add_new_frag` gives a new plan priority 0 (`flist.c:359`), and the flag lets it keep its own priority, so that it is returned only at the top of the queue.

4 Benchmark, Evaluation, and Computing

Benchmark. The inputs are the 63 files in ITB’s `tests/old_format` directory, grouped into the suites of Table 1; two of them are shell scripts rather than inputs, which leaves 61. They include the `multiway_join` ladder of 3- to 10-way joins. Every run uses `-I 300` and at most 20,000 A* iterations, and E7 adds a few constructed inputs.

Evaluation. Table 2 gives each component, what it compares and what it reports. All arms come from one patched tree and differ only by compile-time flags. The primary comparison is h_Q against the current h on the inputs where search order matters, which E1 finds as those whose iteration count changes when the queue is ordered by g alone. It succeeds if iterations fall there and no input loses a plan or gets a dearer one; everything else is descriptive.

Follow-ups. E1 to E5 can show that an estimate helps or hurts without showing why. Three follow-ups are planned for that, with hypotheses written in the project log after E2 and E3 and before each runs: ties in f broken toward larger g ; E6, every arm with the goal test at expansion, which also shows whether today’s first plans are the cheapest the search can reach and what proving that costs; and E7, which tests any pattern E2 suggests about where h_Q helps on constructed inputs that differ in one property, starting from the `exor/orex` pair of my thesis work.

Computing and timeline. CPU only, on my laptop under `WSL2`; nothing else is needed. h_Q with E1 and E2 by November 8, h_C with E3 by November 22, and E4 to E7 and the report by December 6, two days before the deadline. Risks: if h_Q changes little, its per-fragment values will show why, which still answers Q1. I keep ITB’s naive cost model, under which every atom costs 100 on the sanctioned pipeline. Timings come from one shared laptop, so iteration and fragment counts are the primary measures.

Table 1: Benchmark: the files in ITB’s `tests/old_format` directory. Two of the single files, `bptests.fol` and `formulas.fol`, are shell scripts rather than inputs, so 61 inputs remain.

Suite	Example inputs	Files
848ex	base, horizontal, indexonly, subclass, two-level	22
multiway_join	3way to 10way, with -dis variants	15
update	students_update, students_full_update	5
book-em	book-em, book-em-v2 to book-em-v4	5
magicprop	base	4
eqtests	eq, eqtest, eqhsa	3
single files	nash, tableaux15, rw, csv-index, bbfunceqkey	9
total		63

Table 2: Evaluation plan. Arms: *current h* (the stock planner), *g only* (queue ordered by *g*, while *h* still detects goals and prunes), *h_Q*, *h_C* and *h_Q+h_C* (Section 1). All arms come from one patched tree and differ only by compile-time flags; every run uses the setup of Section 4. E6 and E7 are the follow-ups of Section 4.

Component	Inputs	Compared	Reported
E1. Baselines	all 61	current <i>h</i> ; <i>g only</i>	iterations, fragments and plan cost; the inputs where search order matters
E2. Quantification term	all 61	current <i>h</i> ; <i>h_Q</i>	iterations, fragments, queue size, time, plan cost and plan text; inputs that newly plan; primary: iterations on the inputs from E1
E3. Closing-set term	all 61	<i>h_Q</i> ; <i>h_C</i> ; <i>h_Q+h_C</i>	as E2
E4. Admissibility	nodes of every returned plan	every estimate, on the plans of every arm	nodes whose estimate exceeds the cost the plan adds above them; one refutes admissibility
E5. Join ladder	multiway_join, 3- to 10-way	all arms	largest arity that plans; plan cost against the optimum
E6. Goal test at expansion	all 61	each arm; the same arm with the goal test at expansion	iterations, plans lost within the budget and plan cost; whether the first plan is the cheapest the search reaches
E7. Constructed inputs	exor, orex and the same construction with conjunctions, 2 to 5 operands	all arms, under both goal tests	iterations and plan cost

References

- Benedikt, M., B. ten Cate, and E. Tsamoura (2014). “Generating Low-cost Plans from Proofs”. In: *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*. ACM, pp. 200–211.
- Feng, E. H. (2023). “Enumerating Query Plans via Conditional Tableau Interpolation”. MMath thesis. Waterloo, Ontario, Canada: University of Waterloo.
- Haffner, I. and J. Dittrich (2023). “Efficiently Computing Join Orders with Heuristic Search”. In: *Proceedings of the ACM on Management of Data* 1.1, 73:1–73:26.
- Hart, P., N. Nilsson, and B. Raphael (1968). “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. In: *IEEE Transactions on Systems Science and Cybernetics* 4.2, pp. 100–107.
- Hudek, A., D. Toman, and G. Weddell (2015). “On Enumerating Query Plans Using Analytic Tableau”. In: *Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2015)*. Lecture Notes in Computer Science. Springer, pp. 339–354.
- Toman, D. and G. Weddell (2018). “An Interpolation-based Compiler and Optimizer for Relational Queries (System Design Report)”. In: *IWIL Workshop and LPAR Short Presentations*. Vol. 1. Kalpa Publications in Computing. EasyChair, pp. 1–10.