Query Optimization: Exercise
Session 9

Bernhard Radke

December 17, 2018
Homework
Give the permutation with rank 64 of 8 relations.
Give the shape of the random join tree with rank 125 of 8 relations.
Metaheuristics [2]
Metaheuristics

- Iterative Improvement
- Simulated Annealing
- Tabu Search
Genetic Algorithms
Big picture

- Create a “population”, i.e. create $p$ random join trees
- Encode them using ordered list or ordinal number encoding
- Create the next generation
  - Randomly mutate some members (e.g. exchange two relations)
  - Pairs members of the population and create “crossovers”
- Select the best, kill the rest

Details

- Encodings
- Crossovers
Encoding
Ordered lists

- Simple
- Left-deep trees: Straight-forward
- Bushy trees: Label edges in join-graph, encode the processing tree just like the execution engine will evaluate it

Ordinal numbers

- Are slightly more complex
- Manipulate a list of relations (careful: indexes are 1-based)
- Left-deep trees: \[ (((R_1 \bowtie R_4) \bowtie R_3) \bowtie R_2) \bowtie R_5 \] $\mapsto$ 13211
- Bushy trees: \[ (R_3 \bowtie (R_1 \bowtie R_2)) \bowtie (R_4 \bowtie R_5) \] $\mapsto$ 12 21 23 12
Subsequence exchange for ordered list encoding

- Select subsequence in parent 1, e.g. $abcdefg$
- Reorder subsequence according to the order in parent 2

Subsequence exchange for ordinal number encoding

- Swap two sequences of same length and same offset
- What if we get duplicates?

Subset exchange for ordered list encoding

- Find random subsequences in both parents that have the same length and contain the same relations
- Exchange them to create two children
Combinations
Combinations

- 2PO (II and then SA)
- AB Algorithm (IKKBZ and then II)
- Toured SA (SA for each join sequence produced by GreedyJoinOrdering-3)
- GOO-II (run II on the result of GOO)
Iterative Dynamic Programming [1]
Iterative Dynamic Programming

IDP-1
- build solutions up to size k using DP
- replace the cheapest with a compound relation
- repeat until all relations are covered

IDP-2
- greedily build a solution for the complete query (e.g. using GOO)
- find the most expensive subtree that covers at most k relations
- optimize that subtree using DP
- replace the original subtree a compound relation representing the DP solution
- repeat until a single compound relation remains (or out of budget)
Next Homework

- Give an example where II does not find the optimal solution
- Implement Quick-Pick
  - choose your own queries on the TPC-H dataset
  - run them (either using the tester or a dedicated executable)
▶ Slides: db.in.tum.de/teaching/ws1819/queryopt
▶ Exercise task: gitlab
▶ Questions, Comments, etc:
  mattermost @ mattermost.db.in.tum.de/qo18
▶ Exercise due: January 7th, 2019, 9 AM
Iterative dynamic programming: a new class of query optimization algorithms.

Heuristic and randomized optimization for the join ordering problem.