Query Optimization
Exercise Session 7

Andrey Gubichev

December 1, 2014
Enumerating Complementary Subgraphs

EnumerateCmp\((G, S_1)\)
\[
X = B_{\min(S_1)} \cup S_1;
\]
\[
N = \mathcal{N}(S_1) \setminus X;
\]
for all \((v_i \in N\) by descending \(i\)) \{
   \begin{align*}
   &\textbf{emit} \ \{v_i\}; \\
   &\text{EnumerateCsgRec}(G, \ \{v_i\}, \ X \cup (B_i \cap N));
   \end{align*}
\}

- EnumerateCsg + EnumerateCmp produce all ccp
- resulting algorithm DPccp considers exactly \#ccp pairs
- which is the lower bound for all DP enumeration algorithms
Syntactic eligibility set - relations that have to be in the input

Total eligibility set - captures also reordering restrictions, construct bottom-up

Conflicts: \( C.x = E.y \) and \( C.y = D.x \), \( C.x = E.y \) and \( B.x = C.y \)
Important: consider all possible edge combinations, that is,

\[ \text{benefit}(R_0 \bowtie R_1, R_0 \bowtie R_2) \text{ together with } \]
\[ \text{benefit}(R_0 \bowtie R_2, R_0 \bowtie R_1) \]
Homework: Graph Simplification

\[
\begin{array}{c|c|c|c|c}
R_0 & 0.1 & 20 & 0.05 & 5000 \\
10 & & R_1 & & \\
R_3 & 0.01 & R_2 & & \\
0.2 & & & & 0.1 \\
500 & & 50 & &
\end{array}
\]

- \text{benefit}(R_0 \bowtie R_1, R_0 \bowtie R_3) = \frac{202}{300}
- \text{b}(R_0 \bowtie R_3, R_0 \bowtie R_1) = \frac{300}{202}
- \text{b}(R_1 \bowtie R_2, R_1 \bowtie R_0) = \frac{20}{12}
- \text{b}(R_3 \bowtie R_0, R_3 \bowtie R_2) = 2
- \text{b}(R_2 \bowtie R_3, R_2 \bowtie R_1) = \frac{5}{4}
- \text{b}(R_1 \bowtie R_4, R_1 \bowtie R_0) = \frac{500}{251}
- \text{b}(R_1 \bowtie R_4, R_1 \bowtie R_3) = \frac{300}{251}
- R_3 \bowtie R_2 \text{ before } R_3 \bowtie R_0. \text{ Remove } R_3 - R_0
Homework: Graph Simplification

\[\begin{array}{cccc}
R_0 & 10 & 0.1 & 20 \\
R_1 & 0.05 & 5000 & R_4 \\
R_3 & 0.2 & 0.01 & R_2 \\
R_2 & & & 500 \\
R_1 & 50 & & 5000 \\
\end{array}\]

- \(\text{benefit}(R_0 \times R_1, R_0 \times R_3) = \frac{202}{300}\)
- \(b(R_0 \times R_3, R_0 \times R_1) = \frac{300}{202}\)
- \(b(R_1 \times R_2, R_1 \times R_0) = \frac{20}{12}\)
- \(b(R_2 \times R_3, R_2 \times R_1) = \frac{5}{4}\)
- \(b(R_1 \times R_4, R_1 \times R_0) = \frac{500}{251}\)
- \(b(R_1 \times R_4, R_1 \times R_3) = \frac{300}{251}\)
- \(b(R_0 \times (R_3 \times R_2), R_0 \times R_1) = \frac{\text{C}(\text{C}(R_0 \times (R_3 \times R_2)) \times R_1)}{\text{C}(\text{C}(R_0 \times R_1) \times (R_3 \times R_2))} = \frac{850}{370}\)
- \(b((R_2 \times R_3) \times R_0, R_2 \times R_1) = \frac{\text{C}(\text{C}((R_2 \times R_3) \times R_0) \times R_1)}{\text{C}(\text{C}(R_2 \times R_3) \times (R_1 \times R_0))} = 1\)
- \(R_0 \times R_1 \text{ before } R_0 \times (R_3 \times R_2)\)
Generating Permutations

ConstructPermutationsRec($P$, $R$, $B$)

**Input:** a prefix $P$, remaining relations $R$, best plan $B$

**Output:** side effects on $B$

if $|R| = 0$

- if $B = \epsilon \lor C(B) > C(P)$
  - $B = P$

else

  for each $R_i \in R$

    if $C(P \circ < R_i >) \leq C(P[1 : |P| - 1] \circ < R_i, P[|P|] >)$
      - ConstructPermutationsRec($P \circ < R_i >, R \setminus \{R_i\}, B$)
Generating Permutations

1000
\[ R_1 \quad 0.005 \quad R_2 \]

0.05
\[ R_4 \quad 0.02 \quad R_3 \]

100
500

- Keep current prefix and the rest of relations
- Extend the prefix only if exchanging the last two relations does not result in a cheaper sequence
Memoization

- DP: bottom-up construction of the join tree
- Memoization: top-down construction
- Memoize already generated join tree to avoid duplicate work
- Sometimes more efficient
Algorithms: Roadmap

- Deterministic
  - Exact (IKKBZ, DP, Permutations, Memoization,...)
  - Heuristics (GOO, MVP, Query Simplification,...)
- Probabilistic
- Hybrid
Random left-deep trees with cross products

- there are $n!$ trees (every tree - permutation)
- let’s generate a random number in $[0, n!]$
- *unranking* - for a generated number construct a tree
- *ranking* - for a tree define it’s number
Generating random permutations

\[
\text{for each } k \in [0, n[ \text{ descending}
\]
\[
\text{swap}(\pi[k], \pi[\text{random}(k)])
\]

Array \( \pi \) initialized with elements \([0, n[. \text{ random}(k) \) generates a random number in \([0, k]. \)
Unranking

Unrank($n, r$)

**Input:** the number $n$ of elements to be permuted
and the rank $r$ of the permutation to be constructed

**Output:** a permutation $\pi$

for each $0 \leq i < n$

$\pi[i] = i$

for each $n \geq i > 0$ descending { 

swap($\pi[i - 1], \pi[r \ mod \ i]$)

$r = \lfloor r/i \rfloor$

}

return $\pi$;
Random join trees with cross products

- Generate a tree, then generate a permutation: $C(n - 1)$ trees, $n!$ permutations
- Pick a random number $b \in [0, C(n - 1)]$, unrank $b$
- Pick a random number $p \in [0, n!]$, unrank $p$
- Attach the permutation to the leaves
Unranking

- every tree is a word in \{ (, ) \}
- map such words to the grid, every step up is (, down )
Unranking

- every tree is a word in \{(), \}\n- map such words to the grid, every step up is (, down )
Unranking

- every tree is a word in \{ (, ) \}
- map such words to the grid, every step up is (, down )
- the number of different paths \( q \) can be computed (see lectures)
- Procedure: start in (0,0), walk up as long as rank is smaller than \( q \). When it is bigger, step down, \( \text{rank} = \text{rank} - q \)
Example

- Bushy tree number 56, 8 leaves
Random Join Tree Selection
Random Join Tree Selection
Random Join Tree Selection
Random Join Tree Selection
Random Join Tree Selection

![Diagram of Random Join Tree Selection](image)

- Node 429
- Node 198

arrows indicating connections between nodes
Random Join Tree Selection
Random Join Tree Selection
Random Join Tree Selection

![Diagram of a random join tree selection pattern](image_url)
Random Join Tree Selection
Random Join Tree Selection
Random Join Tree Selection
Random Join Tree Selection

![Diagram of Random Join Tree Selection with numbers 429, 198, 165, 75, 27, 20, and 1 on a grid.]
Random Join Tree Selection
Random Join Tree Selection
Random Join Tree Selection

![Diagram of Random Join Tree Selection]
Random Join Tree Selection

![Random Join Tree Selection Diagram]

- Numbers: 429, 198, 165, 75, 27, 20, 14, 5, 4
- Diagram shows a set of values connected in a triangular pattern.
Random Join Tree Selection
Random Join Tree Selection
Random Join Tree Selection

![Diagram of random join tree selection with numbers at each node.](image-url)
Random Join Tree Selection
Info

- Exercises due: 9 AM, December 8, 2014