SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Tuning Linear Programming Solvers for Query Optimization

Sarra Ben Mohamed

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Tuning Linear Programming Solvers for Query Optimization

Anpassung von Linear Programming Solvern für Anfrageoptimierung

Author:	Sarra Ben Mohamed
Supervisor:	Prof. Dr. Thomas Neumann
Advisor:	Altan Birler
Submission Date:	15/10/2023

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15/10/2023 Sarra Ben Mohamed

Acknowledgments

Abstract

Contents

1 Introduction

Our aim with this project is to investigate and compare different methods and techniques to solve small linear programming problems representing the problem of cardinality estimation. Our goal is to estimate realistic and useful upper bounds on query sizes. Studies have shown that cardinality estimation is the major root of many issues in query optimization. [Ngo22] And yet, theoretical upper bounds that are way too large would be useless since we want practical estimation to choose the best from data plans to run efficient queries. For this purpouse, we will introduce a formal description of the cardinality estimation problem, represent it in the form of a packing linear programming problem with the intention of maximizing the size of the query under some constraints. The result is hundreds of relatively small LP that we collect in datasets and solve them with different methods and algorithms. We then draw conclusions based on the results of our experiments, benchmarks and the previous work done on similar packing LP problems. This should guide us into constructing a thorough analysis of the particularities of these LP problems, what's unique about their structure and if their solution process is following any patterns. We then discuss and draw hypotheses on the ways this analysis can be exploited to further optimize the solution process: which methods or combination of methods deliver the best time and memory complexity.

2 Related work

2.1 Background

or Fundamentals: the knowledge the reader needs to understand my contribution, mostly definition of mathematical concepts needed

2.1.1 Cardinality Estimation

Defining the problem of upper bounding a multi-join query size as a packing linear programming problem. To illustrate the main ideas, we start with an example where the query is a simple join between two relations

$$Q(X, Y, Z) = R(X, Y) \wedge S(Y, Z)$$

In the context of our packing LP problem, we start with the inequality 2.1. Applying the natural logarithm to both sides yields 2.2. We then rename the variables, simplifying the inequality to 2.3. Normalizing by dividing both sides by r', we obtain 2.4. This leads us to the objective function for our packing LP problem.

$$|a| \cdot |b| \leq |R| \tag{2.1}$$

$$\ln |a| + \ln |b| \leq \ln |R| \tag{2.2}$$

$$a' + b' \leq r' \tag{2.3}$$

$$\frac{1}{r'}a' + \frac{1}{r'}b' \leq 1 \tag{2.4}$$

$$\text{maximize } a' + b' + c' + d' \quad \text{s.t.} \quad \frac{1}{r'}a' + \frac{1}{r'}b' \leq 1 \tag{2.5}$$

2.1.2 Linear Programming

The LP problem class that we are dealing with is called the packing LP problem. Additionally it is a special instance where:

- c, the vector of the variable coefficients in the objective function, is a vector of all ones

- b , or the right hand side vector, is a vector of all ones

Our specific problem is then expressed as follows:

$$\begin{aligned} \text{Maximize} \quad & \sum_{j=1}^{n} x_j \\ \text{subject to} \quad & \\ & \sum_{j=1}^{n} a_{ij} x_j \leq 1, && i = 1, \ldots, m \\ & x_j \geq 0, && j = 1, \ldots, n \end{aligned} \tag{2.6}$$

- x_j is the j^{th} decision variable.
- m is the number of constraints.
- n is the number of variables.

2.1.3 The Simplex Algorithm

The algorithm

In this subsection we will present the most widely used algorithm for solving LP problems and that we also used, among others, to solve our dataset. To be approachable by the simplex algorithm LP 2.6 needs to be cast in a computational form, that fulfills the requirement of the constraint matrix having to have full row rank and only equality constraints are allowed. This is done by introducing slack variables. We now have what is called the Simplex Tableau.

1. the standard algorithm is the tabular form
2. feasible dictionaries
3. the grand strategy of the simplex method is that of successive improvements
4. decision variables vs. slack variables
5. A maximization problem is optimized when the slack variables are “squeezed out,” maximizing the true variables’ effect on the objective function. Conversely, a minimization problem is optimized when the slack variables are “stuffed,” minimizing the true variables’ effect on the objective function.
6. feasibility, boundedness,

7. largest coefficient rule vs. largest increase rule.

8. the problem of stalling, degeneracy

9. Bland's rule guarantees termination.

The complexity

The simplex method is an active set method. Each step of the simplex method deactivates one box constraint and selects another one to be activated (general linear constraints are always satisfied). Typically for an active set method, O(N+M) steps are needed for an N-dimensional problem with M general linear constraints.

2.1.4 The Revised Simplex Algorithm

As explained in the book [Chv83]. Mention zero tolerances: A zero tolerance epsilon2 saefguards against divisions by extremely small numbers, which tend to produce the most dangerous rounding errors, and may even lead to degeneracy. diagonal entry in eta matrix should be fairly far from otherwise (in our experiment) degeneracy.

The product form update method

We will discuss the PFI, introduced by George Dantzig [DO54].

Data structures

Compressed Storage Formats: Eigen uses either the CSC or CSR format to store sparse matrices. These formats store the non-zero values, along with their corresponding row and column indices, in a compact way. This reduces memory usage and speeds up operations on sparse matrices

2.2 Previous Work

Here we will discuss alternative approaches that are superseded by my work.

2.2.1 Comparative studies of different update methods

We will focus on one study [HH15].

2.2.2 Other techniques

The primal simplex method starts from a trial point that is primal feasible and iterates until dual feasibility. The dual simplex method starts from a trial point that is dual feasible and iterates until primal feasibility. ALGLIB implements a three-phase dual simplex method with additional degeneracy-breaking perturbation:

- Forrest-Tomlin updates for faster LU refactorizations
- A bound flipping ratio test (also known as long dual step) for longer steps
- Dual steepest edge pricing for better selection of the leaving variable
- Shifting (dynamic perturbations applied to cost vector) for better stability

3 Tuning Linear Programming Solvers for Query Optimization

This is the body

3.1 Proposal

3.2 Experimental Design

3.2.1 Analysis of dataset properties

In this subsection we will conduct an analysis of our dataset properties. What are the particularites of the structure of these LP problems, is their any patterns in their solution process. This anaylsis is based on observing the statistical results we obtained from running different solvers on these problems. This will later provide us with insight regarding optimization of these problems. TPC-H is a Decision Support Benchmark The TPC-H is a decision support benchmark. It consists of a suite of business oriented ad-hoc queries and concurrent data modifications. The performance metric reported by TPC-H is called the TPC-H Composite Query-per-Hour Performance Metric (QphH@Size)

3.2.2 Dataset Structure

Our dataset stucture: as opposed to what the linear programming research has dealt with, which is very large problems, we are dealing with hundreds of small problems. These are represented in the revised simplex algorithm by sparse matrices but not as sparse as it would have been if the problem was large, small matrices that are not small enough to be dense. (they still have quite a number of non-zeroes).

3.3 Analysis

Some metrics:

- number of iterations

- runtime
- number of loops ?
- for matrix : number of columns and rows, nonzeros and density

3.4 Results

All the following results have been obtained on a personal computer with AMD 4000 series RYZEN, 16GB RAM running Ubuntu. Using the following settings:

- Presolve techniques are not used
- scaling techniques are not used
- The computed optimal solutions have been validated using the scipy python library.

4 Evaluation

4.1 Setup

4.1.1 Evaluation metrics

4.1.2 Evaluation baselines

4.2 Results

4.3 Discussion

5 Conclusion

List of Figures

List of Tables

Bibliography

[Chv83] V. Chvátal. *Linear programming*. Macmillan, 1983.

[DO54] G. B. Dantzig and W. Orchard-Hays. "The product form for the inverse in the simplex method." In: *Mathematical Tables and Other Aids to Computation* (1954), pp. 64–67.

[HH15] Q. Huangfu and J. J. Hall. "Novel update techniques for the revised simplex method." In: *Computational Optimization and Applications* 60 (2015), pp. 587–608.

[Ngo22] H. Q. Ngo. "On an Information Theoretic Approach to Cardinality Estimation (Invited Talk)." In: *25th International Conference on Database Theory (ICDT 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2022.