University of Zurich[UZH]

Department of Informatics

University of Zürich
Department of Informatics
Binzmühlestr. 14
CH-8050 Zürich
Phone. +41 44 635 43 11
Fax +41 44 635 68 09
www.ifi.uzh.ch/dbtg

UZH, Dept. of Informatics, Binzmühlestr. 14, CH-8050 Zürich

Prof. Dr. Michael Böhlen
Professor
Phone +41 44 635 43 33
Fax +41 44 635 68 09
boehlen@ifi.uzh.ch

Zürich, 28. Juni 2017

Bachelor Thesis: Linear Optimization in Relational Databases

Introduction to Simplex

The concept of optimization refers to solving complex decision problems, involving the selection of values for a number of interrelated variables. The goal of an optimization problem is to maximize (or minimize) an objective function that quantifies the quality of the decision, subject to a set of constraints that limits the selection of decision variable values. A special case of optimization problems are Linear Programs (LPs) [1]. The term LP refers to optimizing a linear objective function under constraints that are expressed as a set of linear equalities or inequalities.

LP are utilized in the context of feed mills to determine feed compounds that satisfy various restrictions, which domain experts specify to optimize the efficiency and health of the livestock. Information about nutrient content of feeds is provided by the Swiss Feed Database[1]. Example data is shown below; the containment of each nutrient in a feed is given by attribute `g` (g/Kg) of `fact_table`. Various nutrient constraints indicate the minimum and maximum containment (g/Kg) of a nutrient in a feed compound. A farmer selects a subset of these domain expert recommendations that must be satisfied. For example, a farmer may select all general nutrient constraints ('general-LYS', 'general-DEP') and one of the protein constraints (e.g., 'low-protein'), and ask for a diet with the minimum cost. Alternatively it is possible to also set a constraint for the cost and ask for a nutrient, such as energy (DEP), to be maximized.

[1] `http://feedbase.ch`

fact_table		
fid	**nid**	**g**
Barley	LYS	1
Barley	CP	3
Barley	DEP	12
Hay	LYS	4
Hay	CP	1
Hay	DEP	1
Soy	LYS	0.7
Soy	CP	2
Soy	DEP	10
...		

nutrient_constraints			
diet	**nid**	**min**	**max**
general-LYS	LYS	9.6	9.8
general-DEP	DEP	NULL	3
low-protein	CP	NULL	2
medium-protein	CP	2	4
high-protein	CP	4	6
...			

feed_prices	
fid	**price**
Barley	1
Hay	3
Soy	1
...	

Relational database systems do not support optimization problems and the usual solution is to extract relevant data, call an external solver, and import the solution back into the RDBMS. This is a time consuming and error prone process. The goal of this project is to study the integration of linear optimization solutions in an RDBMS like PostgreSQL.

A LP is defined as follows:

$$\begin{aligned} \text{Minimize/Maximize:} \quad & z = \Sigma_v(c_v x_v) \\ \text{Subject to:} \quad & l_c \leq \Sigma_v(g_{vc} x_v) \leq u_c \\ & l_v \leq x_v \leq u_v \\ & x_v \in \mathbb{R}, \forall v \end{aligned} \tag{1}$$

Where,

- v and c go through all ids of variables (feeds) and constraints respectively.
- x_v represents the proportion of each feed that comprises the result diet compound.

Simplex

An instance of a LP can be the following where we ask for a diet with maximum energy that satisfies a maximum constraint for the crude protein (CP) containment and a maximum constraint for the diet cost.

$$\begin{aligned} \text{Maximize} \quad & z = 12x_{Barley} + 1x_{Hay} + 10x_{Soy} \\ \text{Subject to} \quad & 3x_{Barley} + x_{Hay} + 2x_{Soy} \leq 2 \\ & 1x_{Barley} + 3x_{Hay} + 1x_{Soy} \leq 4 \\ \text{where} \quad & x_{Barley}, x_{Hay}, x_{Soy} \geq 0 \end{aligned} \tag{2}$$

Relational Implementation

A possible representation of the LP above in the context of relational databases is the following Simplex table:

Constraint id	Barley	Hay	Soy	Constraint expression
CP	3	1	2	'≤ 2'
price	1	3	1	'≤ 4'
DEP	12	1	10	maximize

In order to decouple the bound of a constraint from its direction (whether it is an upper or a lower bound), we create internally one more column for every constraint. This column represents a surplus value so that inequality is transformed into equality.

Regarding optimization (minimize/maximize) constraints we make the following assumptions:

1. A LP has only one optimization constraint. This constraint is always stored in a tuple of the Simplex table with `Constraint id` 'optimize'.
2. The bound of an optimization constraint is 0.
3. Only minimization constraints are used. In case of a maximization constraint the corresponding values are multiplied by -1 so that the constraint is transformed into a minimization constraint.

Constraint id	Barley	Hay	Soy	s_1	s_2	RHS
CP	3	1	2	1	0	2
price	1	3	1	0	1	4
optimize	-12	-1	-10	0	0	0

The first row of the table above is read as follows: "Crude Protein (CP) can be found in Barley (3 g/Kg), Hay (1 g/Kg), Soy (2 g/Kg) and a surplus placeholder (1 g/Kg). The goal is to determine proportions of Barley, Hay, and Soy that along with a surplus result in 2 g/Kg of CP". The second row is read in a similar way. The last row is about the maximization constraint and is read as follows "Energy (DEP) can be found in Barley (12 MJ/Kg), Hay (1 MJ/Kg), Soy (10 MJ/Kg). The goal is to determine proportions of Barley, Hay, and Soy that maximize the energy content".

This Simplex table encapsulates an initial solution as follows. The coefficients of the surplus columns formulate an identity matrix. So, a trivial solution is determined by setting these surplus columns equal to RHS and setting the remaining columns equal to 0. This solution is an empty diet that is represented by the following table.

Barley	Hay	Soy	s_1	s_2	DEP
0	0	0	2	4	0

The idea of Simplex is that in each iteration it determines a zero variable to increase and a positive variable to set to zero. The Simplex table is transformed so that it represents an equivalent LP. After the transformation the identity matrix corresponds to the new solution. In more detail, Simplex is applied on the table to improve the initial solution as follows:

Loop:

1. *Determine pivot column (pc).* Find the column with the minimum negative value in the last row (objective function). By increasing the value of this column we decrease the value of the objective function. We call this column `pivot column`. Exit if new pivot column cannot

be determined, then the optimal solution has been found.

2. *Determine pivot row (pr).* Compute the ratio between the RHS column and the pivot column for each constraint. The result is the maximum value by which the pivot column can be increased without breaking a constraint. The minimum ratio corresponds to the `pivot row`, and is the maximum value that the pivot column can be increased without breaking a constraint. Exit if all ratios are non positive; in this case the linear program is unbounded and no optimal solution exists.
3. *Update the pivot row* (U_1). Divide the pivot row with the value of the pivot cell.
4. *Update the remaining rows* (U_2). Subtract the pivot row multiplied by the value of the pivot column.

Simplex is applied iteratively to the intermediate tables until the optimal solution is found or unboundness is proved. After applying one iteration of Simplex to our example the intermediate table is:

Constraint id	Barley	Hay	Soy	s_1	s_2	bound
CP	1	1/3	2/3	1/3	0	2/3
price	0	8/3	1/3	-1/3	1	10/3
DEP	0	3	-2	4	0	8

The solution that is encapsulated in this table is the following:

Barley	Hay	Soy	s_1	s_2	DEP
2/3	0	0	0	10/3	8

For the table above the pivot column is Soy and the pivot row is CP. After applying the update steps we get the following matrix:

Constraint id	Barley	Hay	Soy	s_1	s_2	bound
CP	3/2	1/2	1	1/2	0	1
price	-1/2	5/2	0	-1/2	1	3
DEP	3	4	0	5	0	10

All values of the last row are non positive. So, the optimal solution has been found:

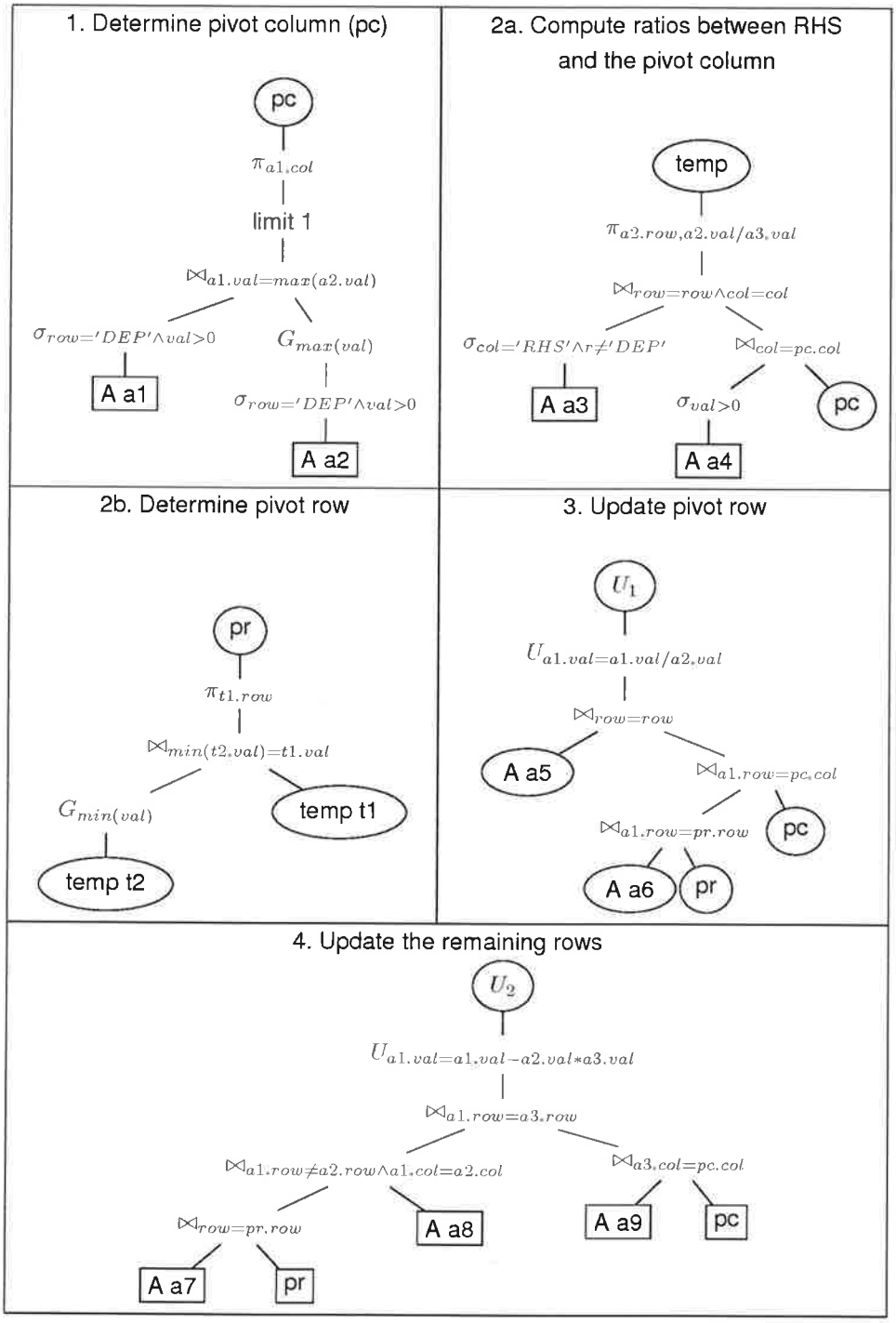
Barley	Hay	Soy	s_1	s_2	DEP
0	0	0	0	1	10

Simplex in SQL

Assume that the Simplex table is stored in the vertical schema `R(row varchar, col varchar, val float)`.

row	col	val
CP	Barley	3
CP	Hay	1
CP	Soy	2
CP	s_1	1
CP	s_2	0
CP	RHS	2
price	Barley	1
price	Hay	3
price	Soy	1
price	s_1	0
price	s_2	1
price	RHS	4
DEP	Barley	12
DEP	Hay	1
DEP	Soy	10
DEP	s_1	0
DEP	s_2	0
DEP	RHS	0

Every step of Simplex can be expressed as a SQL query. Hereafter a query tree is given for each step of the example.

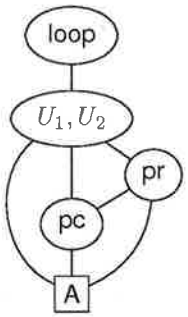


Combining these queries into an iterative process (to fully implement Simplex) is not possible in SQL. The goal of this project is to develop a SQL operator that implements algorithm Simplex following three different approached and study advantages and disadvantages of these approaches.This operator must be called as in the following statement

```
SELECT * FROM A SIMPLEX;
```

The first approach is to use PL/pgSQL in order to combine the queries presented below in a for loop. The second approach is to implement these query trees internally in PostgreSQL kernel.

In the analysis stage the operator must be extended to the following tree



where every node represents one of the query trees presented above. The loop node is a special node that executes U_1 and U_2 iteratively until no tuple is changed. Finally, the third approach is to implement Simplex as an execution node of PostgreSQL kernel in C.

Tasks

1. Study and understand linear optimization problem and the Simplex algorithm [1, 2].
2. Study related work [3].
3. Implement Simplex using PL/pgSQL (first approach).
4. Develop a `plan node` that implements Simplex algorithm as the extended query tree described above (second approach).
5. Implement Simplex as an execution node of PostgreSQL (third approach).
6. Compare the three approaches and determine advantages and disadvantages of each.
7. Describe your work in a report.

References

[1] D. G. Luenberger, Y. Ye, Linear and Nonlinear Programming, Springer International Publishing, 2016, Ch. Basic Properties of Linear Programs, pp. 11–31.

[2] D. G. Luenberger, Y. Ye, Linear and Nonlinear Programming, Springer International Publishing, 2016, Ch. The Simplex Method, pp. 33–82.

[3] L. Šikšnys, T. B. Pedersen, Solvedb: Integrating optimization problem solvers into sql databases, in: Proceedings of the 28th International Conference on Scientific and Statistical Database Management, SSDBM '16, ACM, New York, NY, USA, 2016, pp. 14:1–14:12.

Supervisor: Georgios Garmpis (ggarmpis@ifi.uzh.ch)

Presentation Date: Tuesday, November 14th 2017

University of Zurich
Department of Informatics

Prof. Dr. Michael Böhlen
Professor