Comments on the implementation of the simplex algorithm

Disclaimers

Disclaimers

The content of these slides will not be asked on the quizzes, midterm of final.

Disclaimers

The content of these slides will not be asked on the quizzes, midterm of final.

I am not a specialist of the the implementation of the simplex method. Information that follows may contain inaccuracies.

Implementations of the simplex

An LP solver based on the simplex method is usually found in the scientific library of a programming language.

Implementations of the simplex

An LP solver based on the simplex method is usually found in the scientific library of a programming language.

If you try to implement naively what we have seen in class, you won't be able to solve more than textbook examples.

Implementations of the simplex

An LP solver based on the simplex method is usually found in the scientific library of a programming language.

If you try to implement naively what we have seen in class, you won't be able to solve more than textbook examples.

Caracteristics of "real life" examples:

- Large scale (number of variables and constraints from 10^3 to 10^8).
- The matrix A is sparse (no more than 1 to 5% of non zero entries).
- The data is structured: you don't solve a "random" LP.
- (You may want to do computations in a distributed fashion).

Complexity

Of course,

$$[\text{Number of operations}] = [\text{Number of pivots}] \times [\text{Operations by pivot}]$$

Complexity

Of course,

$$\text{[Number of operations]} = \text{[Number of pivots]} \times \text{[Operations by pivot]}$$

The number of pivots is in general polynomial in n and m. Reported to be proportional to m for small instances by Dantzig in the 60s. Large literature on the topic for the average case analysis.

Complexity

Of course,

$$[\text{Number of operations}] = [\text{Number of pivots}] \times [\text{Operations by pivot}]$$

The number of pivots is in general polynomial in n and m. Reported to be proportional to m for small instances by Dantzig in the 60s. Large literature on the topic for the average case analysis.

Number of operations by pivot: the bottleneck is inverting B.

Inverting the matrix B

The main bottleneck is about inverting the matrix B, or at least solving systems of the form ($\mathbf{x}, \mathbf{y}$ unknown, $\mathbf{b}, \mathbf{c}_B$ data)

$$B\mathbf{x} = \mathbf{b} \quad \text{or} \quad B^\top \mathbf{y} = \mathbf{c}_B$$

All the tools from numerical linear algebra are *a priori* available: LU factorization, *sparse* LU factorization, preconditioning, etc.

Inverting the matrix B

The main bottleneck is about inverting the matrix B, or at least solving systems of the form ($\mathbf{x}, \mathbf{y}$ unknown, $\mathbf{b}, \mathbf{c}_B$ data)

$$B\mathbf{x} = \mathbf{b} \quad \text{or} \quad B^\top \mathbf{y} = \mathbf{c}_B$$

All the tools from numerical linear algebra are *a priori* available: LU factorization, *sparse* LU factorization, preconditioning, etc.

If B_k is the basis matrix after k iterations we can write it

$$B_k = B_0 E_1 E_2 \dots E_k \quad \text{hence} \quad B_k^{-1} = E_k^{-1} E_{k-1}^{-1} \dots E_1^{-1} B_0^{-1}$$

with the E_k simple to store and invert.

Inverting the matrix B

The main bottleneck is about inverting the matrix B, or at least solving systems of the form ($\mathbf{x}, \mathbf{y}$ unknown, $\mathbf{b}, \mathbf{c}_B$ data)

$$B\mathbf{x} = \mathbf{b} \quad \text{or} \quad B^\top \mathbf{y} = \mathbf{c}_B$$

All the tools from numerical linear algebra are *a priori* available: LU factorization, *sparse* LU factorization, preconditioning, etc.

If B_k is the basis matrix after k iterations we can write it

$$B_k = B_0 E_1 E_2 \dots E_k \quad \text{hence} \quad B_k^{-1} = E_k^{-1} E_{k-1}^{-1} \dots E_1^{-1} B_0^{-1}$$

with the E_k simple to store and invert.

But numerical instabilities accumulate and more and more E_k need to be stored and used: a new factorization of the matrix B is computed from scratch every 100 iterations or so.

Other catches

The problem can have a structure and you want to leverage that: standard form is not always the most appropriate.

Other catches

The problem can have a structure and you want to leverage that: standard form is not always the most appropriate.

The number 0 (or more specifically non-negativity or non-positivity of coefficients) plays a special role. One replaces 0 by a tolerance `tol` very small, but for stability reason you must avoid dividing by small numbers. There could be a trade off between accuracy and stability.

Other catches

The problem can have a structure and you want to leverage that: standard form is not always the most appropriate.

The number 0 (or more specifically non-negativity or non-positivity of coefficients) plays a special role. One replaces 0 by a tolerance `tol` very small, but for stability reason you must avoid dividing by small numbers. There could be a trade off between accuracy and stability.

How to choose entering and leaving variable?

Other catches

The problem can have a structure and you want to leverage that: standard form is not always the most appropriate.

The number 0 (or more specifically non-negativity or non-positivity of coefficients) plays a special role. One replaces 0 by a tolerance `tol` very small, but for stability reason you must avoid dividing by small numbers. There could be a trade off between accuracy and stability.

How to choose entering and leaving variable?

- Minimize the number of iterations to reach optimality.

Other catches

The problem can have a structure and you want to leverage that: standard form is not always the most appropriate.

The number 0 (or more specifically non-negativity or non-positivity of coefficients) plays a special role. One replaces 0 by a tolerance `tol` very small, but for stability reason you must avoid dividing by small numbers. There could be a trade off between accuracy and stability.

How to choose entering and leaving variable?

- Minimize the number of iterations to reach optimality.
- Choosing the most stable pivots to decrease numerical instabilites.

Other catches

The problem can have a structure and you want to leverage that: standard form is not always the most appropriate.

The number 0 (or more specifically non-negativity or non-positivity of coefficients) plays a special role. One replaces 0 by a tolerance `tol` very small, but for stability reason you must avoid dividing by small numbers. There could be a trade off between accuracy and stability.

How to choose entering and leaving variable?

- Minimize the number of iterations to reach optimality.
- Choosing the most stable pivots to decrease numerical instabilites.
- "Partial pricing": compute $\mathbf{c}_N^\top - \mathbf{c}_B^\top B^{-1} A_N$ one coefficient after another and pick among the first ones which are negative as entering.