EFREI 2020/2021
M1 ITF
ST2NOM
Project report

Group id.	3
Members of the group	BEN SIDHOM Oussama, DURAT Nicolas, ES-SEFFAR Sophia, LANGLET Loïc, MAJOUL Mohamed, STANKOVIC Adrien

Table of content

Progress of work

« Item of work »: a functional and/or technical component of the solver.
"Status": either of 'done' (implemented and tested), 'under progress', 'not yet started'.
"Comment": any additional information about the status.

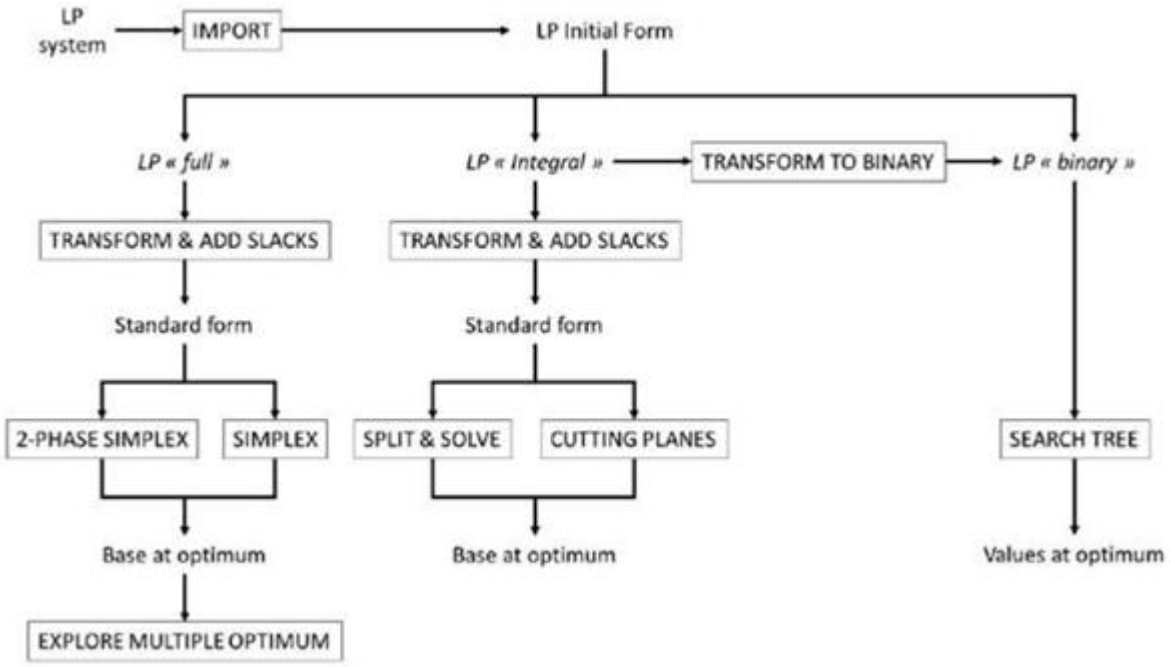
Item of work	Status	Comment
Add slacks	***Done***	/
Simplex	***Done***	/
Two-phase Simplex	***Done***	/
Multiple optimum	***Done***	
Artificial variables in base at end of phase 1	***Done***	
Search tree for "binary" systems	***Done***	/
Transform "integral" to "binary"	***Done***	/
"Split & Solve"	***Done***	/
"Cutting plane"	***Done***	/

Overall architecture

The main program named “Projet_1.0.R” loads all functionalities (R files) given in the following representation:

Initial architecture *(from 2019/2020 description of the work to be done)*

You can (but are not required to) start designing your architecture with the following diagram.



We followed this architecture to implement our project and every square is represented by an R file (except for IMPORT which is in the main).

The main program will also load 7 different linear systems written in excel files named “LP_System?.xlsx” with “?” the number of the file.

Then a menu displays asking the user which method he wants to use to solve its linear system: full, integral of binary system.

For the Full and Integral method, the TRANSFORM & ADD SLACKS corresponds to our “Slacks.R” file. Concerning the full method, the 2-PHASE SIMPLEX and SIMPLEX are implemented in “2-Phase_Simplex.R” (which also contains another function that checks if the system should use the simplex or 2-phase simplex resolution) and “Simplex.R”. At the end, the EXPLORE MULTIPLE OPTIMUM has not been implemented.

Concerning the Integral method, the file “Integral.R” checks if the system can be solved by the binary method. If not, then “Slacks.R” is used for TRANSFORM & ADD SLACKS. The SPLIT & SOLVE is coded in a function from “Split.R” file and CUTTING PLANES is defined by “Cut.R”. The user must decide between these two methods. However, the “Cut.R” file is empty because we couldn’t figure out how to implement it.

Main functions, classes, data structures

It is anticipated here that your solver will be made of different functions, and that you will have defined data structures to represent the systems that you have to solve. Describe all of this here. For instance, you may have defined a `gaussPivot(…)` function used by the Simplex, a `simplex (…)` function to run Simplex or 2-phase Simplex/ phase 1, You may also have defined a particular data table to use as input/output parameter to the `simplex ()` function.

Projet 1.0 file:

`main(data)` : Function to start the program, select the type of solution and launch the associated functions (Simplex, Cut, Binary, 2-Phase Simplex etc.).

Slacks file:

`slack(data)` : Function that transforms the LP into a standard form by adding a slack.

Simplex file:

`Simplex(data)` : Function that applies the gaussian pivot to the data without prior modification.

2-Phase Simplex file:

`Two_Phase_Simplex(data,max)` : General function that allows to launch all the functions below to realize the 2-Phase Simplex method.

`Initialization(data)` : Function that adds artificial variables and functions.

`Phase1_Part1(data)` : Function that allows to detect negative numbers to transform them into positive numbers on the Z column.

`Phase1_Part2(data)` : Function that allows to run the first Gaussian pivot on the data.

`Modified_Simplex(data,stop)` : Function that allows to compute the Gaussian pivot of the data.

`Phase2(data,line_to_add)` : Function that starts the second phase of the 2-Phase Simplex to remove the artificial variables and functions added in the first phase.

`End(data,max)` : Function to check if an optimum is reached for the data.

`Simplexs(data)` : Function that allows to choose between Simplex or 2-Phase Simplex resolution.

Binary file:

`Binary(data)` : Function that converts data into binary digits.
`check_constraints(data,node)` : Function that checks if at each step of the method the constraints are verified.

Integral file:

check_boundaries(data): Function that checks if the variables are bounded to apply the IntToBin method.

IntToBin file:

IntToBin(): Function that transforms data invalidated by the Binary function into binary values.

Split file:

Split(): Function that intervenes if the check_boundaries() function is invalid. Resolution method that consists in separating the variables and applying a binary search tree to find the maximum of our data.

Cut file:

Cut(): Function that represents an alternative to Split(). It establishes new constraints to reconcile the data of the whole solutions.

Data flow

Type of the Data needed at the first to run the hole program: Excel file. The program read the excel file in which we have all the necessary data to solve the problem and then we stock this data in a table.

Exchange of data between functions and components: the data is saved and exchanged in table form. For example: the 1-Phase Simplex take a table of data and make some transformation and return a table which can be used by the 2-Phase Simplex.

Major parts or component of the program take a table as parameter make some modifications and return also a table for to be used by another function.