PLEMATH SYSTEM: Linear and Integer Linear Programming in Mathematica System

Mijail Borges-Quintana, Miguel A. Borges-Trenard
Department of Mathematics. Faculty of Sciences.
University of Oriente. Santiago de Cuba, 90500 Cuba
{mijail, mborges}@csd.uo.edu.cu

Edgar Martínez-Moro
Applied Mathematics Department. E.T.S. Arquitectura
University of Valladolid. Valladolid, 47014 Spain
edgar@vax631.cpd.uva.es

January 5, 1997

Abstract

The PLEMATH System has been built as an educational tool for linear Programming and Integer Linear Programming. In this work we show examples that illustrate the use of the system. We describe how the commands executing the different algorithms incorporated to PLEMATH have been made, as well as their possibilities and limitations. Moreover, we exhibit some ideas for further extensions of this work. No knowledge of Mathematica is supposed.

Keywords: Linear Programming, Integer LinearProgramming, Mathematica

1. Introduction

The main objective of this article is to contribute to the spreading of the PLEMATH System, so that the reader interested by its utilization could have a first touch with it and, furthermore, to know its sources as well as how to acquire more information about it. We will be pleased of helping anyone in the utilization of PLEMATH.

PLEMATH is made in the programming language of Mathematica for Windows, this has some advantages such as having a great number of pre-implemented functions, an userfriendly environment, and the possibility of working with exact rational arithmetic.

The algorithms incorporated to PLEMATH can be developed by steps. Therefore, they allow interaction with the user, as the learning process requieres. In this option, when a given problem is being solved, there are avaliable functions that execute the steps, but the order in which they should be used and the different criteria must be known, so that the procedure to follow is similar to how

it would be solved without computer. The advantage is that the user does not need to accomplish the calculations by hand and the partial results are showed on the display. The system has also integrated the functions for the direct calculation, being possible options such as to know the number of iterations when a problem is solved by a given algorithm, stop the calculation process keeping previously data so that it could be used as starting point of a different problem, etc.

The current state of PLEMATH allows its utilization in teaching of topics related to Linear Programming and Integer Linear Programming with excellent results. From the second semester of the course 1997-98, PLEMATH is being used by the Department of Mathematics at the University of Oriente (Santiago de Cuba) and soon it will be incorporated into some subjects of the department of Fundamental Applied Mathematics (University of Valladolid).

The birth of PLEMATH was conditioned by three initial purposes:

1. Provide to students of Mathematics and Computer Science a computer system for the subjects of Linear Programming and Discrete Programming in such a way that they could accomplish guided exercises without carring out by hand the great number of calculations that each of the methods of these subjects require and without needing much knowledge of Mathematica.

2. Allow the lecturer to make greater emphasis in the steps of the methods and in their theoretical basis and to fulfill a greater number of exercises, including those of greater complexity, as well as to use the system for the search of new ones.

3. Give to both, teachers and students,a simple system that allow to solve problems of linear programming and integer linear programming working with exact rational arithmetic.

1.1. Why to create a new system?

The systems that we have had to our scope are limited to their utilization for the solution of a given problem and not to the part of teaching the topics. Consequently, they do not allow interaction with the students as it is required the learning process. For example, they do not take into acount a pivoting function that allows an incremental development step by step of the algorithms. Moreover, the functions that are implemented are not enough and do not offer all the information that is required. An example of this fact can be found in the function Linear Programming of Mathematica. This function solves a problem of linear programming, but it does not show the simplex table resulting. Furthermore, most of the systems do not work with exact rational arithmetic.

To show the ideas of the previous paragraph, we suppose that we are teaching the topics of Discrete Programming, the chapter of Cutting Plans methods. It is of great interest to solve the problems by making use of the different algorithms that are studied in this chapter, for example, the algorithm Gomory 1 and the algorithm of the Primal Cut, having also the possibility of solving them by steps.

Furthermore, we would like functions for the direct calculation implementing the algorithm that it is taught in the class, i.e. the same that the student develops step by step in the practical lectures. This allows the lecturer to have an exact idea of the number of iterations needed to solve a problem proposed to the students. These ideas are in correspondence with the objective 2 stated above.

Among the systems that have been studied we could mention: Lindo, Mathprog, Storm and QS. All of them work with floating point arithmetic and although they have some functions for performing some algorithms by steps, they are not enough for the requirements of the teaching process . Of course, these system have larger objectives than PLEMATH, giving solutions to others kind of problems out of the Linear programming and Integer Linear programming, besides, they mainly devote to practical applications and maybe this is one of the reasons that all of them use floating point arithmetic, which is not so convenient for the teaching process, even more, in the integer programming subjects. These systems have the possibilities to work by steps in case of integer linear programming mainly for Branch and Bound Algorithm, on the other hand, the facilities of Mathematica are very good, taking into account that one could proceed as in the reasoning process like we were solving the problem in a paper.

1.2. Previous knowledge of Mathematica.

1.2.1. General topics. The user needs to be familiar with the environment of the Mathematica System. It is quite similar to any of other systems working under Windows (i.e. options File, Edit, etc). Also it must be know how to evaluate an instruction in Mathematica, as well to identify when it has ended and how to interrupt it in case it were wanted.

Some knowlege of how to work with Mathematica notebooks is necessary in order to study the examples provided with PLEMATH and for storing the solution of the problems with the wished commentaries. Thus, conferences and practical classes can be prepared in the computer, having the possibility of solving the examples within your own notebook.

The above remarks correspond to a general knowledge and any user of Windows will have no problem in learning it.

1.2.2. Some particularities of Mathematica the user must know. It is necessary to note some particularities of Mathematica that are more directly related with the use of PLEMATH. Some of them are:

i. The operator "equal" in Mathematica is the symbol ==, since = corresponds to one of the variants of the operation "assignment".

ii. The functions in Mathematica should have the symbol [ ], in spite of not to possess any argument. This is true also for the functions built in PLEMATH.

iii. The function "Get" is the only one that the user must know for working with PLEMATH (See Section 3). Clearly anyone who is interested in updating PLEMATH or to adapt it to more expecific purpouses must know much more about Mathematica.

2. Characteristics of PLEMATH.

2.1. Structure of PLEMATH.

This system is composed of a program **plemath** with 840 lines of programming in the Language of Mathematica, eight notebooks whose names are **Course, Plemath1,..., Plemath7**, which show several examples. In these examples we try to encompassing all the possibilities that are available with this system. The update to higher versions of PLEMATH must be accompanied with the update of these notebooks and the creation of others (see Section 4).

This system counts with **Hplemath**, a Help where a brief description of all variables and functions are given, as well as an introduction and more information about PLEMATH. Hplemath has the common facilities of a Help, such as the options: Find, Context and Glossary. Suggestions for more information about the uses of an specific variable or function are also given; for example, the notebooks where the specific topic is introduced or particular things are pointed out.

The study of PLEMATH should be started by this paper, the notebook **Course** and **Hplemath**. Later the examples illustrated in the notebooks from **Plemath3** to **Plemath7** could be better understood.

2.2. Variable and functions.

In this section we will show the list of variables and functions that are avaliable to the user. There are others that only are used in the programming process. In the following sections the reader will be able to obtain more information about the variables and functions that we are going to relate below.

Variables: *listvar*, *var*, *funobj*, *restric*, *varbas*, *varNbas*, *nsimplexiter*, *ndualiter*, *ngomiter*, *morecol*, *morerow*, *coladit*, *rowadit*.

Functions: *begin*, *showtable*, $findmin$, *mincocient*, *simplexpiv*, *clear*, *maxcocient*, *newvar*, *addrestric*, *delrow*, *delcol*, *delrowaux*, *delcolaux*, $simplexLPfobj$, $simplexLP1$, *simplexLP*, *dualsimplex*, *gomory*1, *gomory*12, *gomory*13.

There are some functions that are obtained from the previous ones by adding a n before their name. In this case, it can be known the number of iterations. For example, *nsimplexLP*, *nsimplexdual*, *ngomory*1, etc. To Make this distinction of both types is necessary to provide to PLEMATH some functions for the direct search of the solution without having to accomplish other auxiliar tasks, as it would be, for example, to count the number of iterations.

2.3. Implemented Algorithms. Possibilities and Limitations.

PLEMATH, up to now, is devoting to Linear Programming and Integer Linear Programming. We take the ideas, theory and examples from the bibliography cited in order to program the algorithms built in PLEMATH, besides all the functions constructed were tested with many examples and exercises that appear in the bibliography stated.

Simplex Algorithm: There are three functions that implement the simplex algorithm: $simplexLPfobj$, $simplexLP1$, *simplexLP*.

simplexLPfobj: Given an objective function list, the variables, constrains and basic variables; to maximize the first of the objective functions, keeping the auxiliary columns added to the simplex table.

simplexLP1: Just as $simplexLPfobj$, but eliminates the auxiliary columns added.

simplexLP: Solves a linear programming problem (maximum) given the objective function, variable and constrains.

These three functions are differentiated by the requirements needed for their call and in their utilization. In the notebooks incorporated to PLEMATH we show examples of the utilization of these functions and their differences. The functions $simplexLP1$ and *simplexLP* use internally the function $simplex\ LPfobj$. In the section devoted to examples, more characteristics of these functions are shown.

The simplex algorithm programmed in PLEMATH continues the ideas that we had already quoted in 1.1. It is the same that is imparted in the lecture with the exception of the fact that it is guaranteed the stop through the addition of auxiliary columns, which avoids the cycles. Clearly, we intend to build the functions that implement this algorithm and each one of those that have been incorporated to PLEMATH in the more possible efficient way (see Section 4).

PLEMATH also allows to develop the simplex algorithm step by step counting with all the required functions for this purpose, for example: *simplexpiv*, *showtable*, $findmin$, *mincocient*, etc.

Dual Simplex Algorithm:

There is also a function call *dualsimplex* that implements the dual algorithm of the simplex with the same characteristic exposed for the Simplex (outlined in 1.1). It is guaranteed the completion of the algorithm through the addition of auxiliary rows.

dualsimplex: Given $funobj$, *restric*, *var*, *varbas*, calculates the maximum of $funobj$ applying the dual algorithm of the simplex.

For this algorithm there are some options that allow to develop the problem step by step, these are the same as the ones stated for the simplex, one only needs to change *mincocient* for *maxcocient*.

Algorithm Gomory1:

It is the only algorithm that we have implemented for the direct calculation of integer linear programming problems. The outline of this algorithm continues also the ideas in 1.1.

gomory1: Given $funobj$, *restric*, *var*, computes the maximum value of $funobj$ for the integer linear problem associated with these data, making use of the algorithm gomory1.

There are also other functions *gomory*12, *gomory*13, for those problems that require a great number of iterations. These functions give the possibility of keeping the information each certain number of steps, so that the file could be studied and, at the same time, it could serve as an input so as to continue the calculation in case that this is interrupted.

This algorithm can be developed by steps having for this purpouse built in the functions: *simplexLP*, *showtable*, *newvar*, *addrestric*, *dualsimplex*, *delrow*. In addition to these algorithms for the direct calculation, the current state of PLEMATH allows to develop step by step the algorithm of Branch and Bound for mixed or integer linear problems, the algorithm of the Primal Cut, among other, which are taught in the subject of Discrete Programming (see Section 3).

3. Examples.

3.1. **Example 1:**

We show some of the possibilities that are within PLE-

MATH by using the following linear programming problem:

(1)

$$\begin{array}{rcl} Max\, f & = & -5x_1 - 21x_3 \\ \text{subject to:} & & x_1 - x_2 + 6x_3 \geq 2 \\ & & x_1 + x_2 + 2x_3 \geq 1 \end{array}$$

The formulation of the problem (1) in the equality form is:

(2)

$$\begin{array}{rcl} Max\, f & = & -5x_1 - 21x_3 \\ \text{subject to:} & & x_1 - x_2 + 6x_3 - x_4 = 2 \\ & & x_1 + x_2 + 2x_3 - x_5 = 1 \end{array}$$

First, we will solve the problem by applying the simplex algorithm step by step. If we observe the above problem we realize the fact that we are in presence of separation variables with negative coefficient. This happens when there are constrains with the greater or equal sign. Therefore, in this case the initial basic solution can not be obtained by taking as basic variables the separation ones. We recall that one of the forms for solving this problem is to apply the Method of the Two Phases. It is necessary to introduce an auxiliary objective function and artificial variables, which have the value of zero for the problem in (1).

If we proceed according to the Method of the Two Phases, in the first phase we should solve the following auxiliary problem:

(3)

$$\begin{array}{rcl} Max\, f_{aux} & = & -x_6 - x_7 = \\ & = & -3 + 2x_1 + 8x_3 - x_4 - x_5 \\ f & = & -5x_1 - 21x_3 \\ \text{Subject to:} & & x_1 - x_2 + 6x_3 - x_4 + x_6 = 2 \\ & & x_1 + x_2 + 2x_3 - x_5 + x_7 = 1 \end{array}$$

From now on, we proceed as if we were working in the environment of the Mathematica with PLEMATH. The instructions executed will have a number for those cases in which we must return back to use them.

1) $var = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$

2) $restric = \{x_1 - x_2 + 6x_3 - x_4 + x_6 == 2, x_1 + x_2 + 2x_3 - x_5 + x_7 == 1\}$

3) $funobj = \{-3 + 2x_1 + 8x_3 - x_4 - x_5, -5x_1 - 21x_3\}$

The variables var, $restric$, $funobj$ are used in PLEMATH to introduce the variables, constrains and the objective function respectively. The variable funobj is commonly a list of objective functions. The order in which these appear in the list is the priority order with which they will be analyzed by the different implemented optimization functions that admit to work with more than an objective function.

4) $varbas = \{x_6, x_7\}$

In order to use any function built in PLEMATH should be loaded the program **plemath** (see 2.1) with the function *Get* of Mathematica (see in 1.2.2.iii). The program **plemath** contains the definition of all the functions in PLEMATH. After having loaded **plemath**, it is not necessary, neither recomendable, to

load plemath again while one is left withing Mathematica. The syntax for the utilization of the function *Get* is:

> *Get*["address"], where - address - is the location of **plemath**. For example:

5) *Get*["a: \plemath"]

6) *begin*[*funobj*, *restric*, *var*, *varbas*]

The function *begin* is the initializing function. This prepares the internal conditions that are required for the application of any other function of PLEMATH. There are functions that require of the previous application of *begin* and others that make internal use of it.

7) *showtable*[]

For seeing the current simplex table, the function *showtable* is used. In this example, upon applying the function *showtable*, the table 1 shown below will be returned.

Table 1.

		x_1	x_2	x_3	x_4	x_5
$fobj1$	-3	-2	0	-8	1	1
$fobj2$	0	5	0	21	0	0
x_6	2	1	-1	6	-1	0
x_7	1	1	1	2	0	-1

Table 2.

		x_1	x_2	x_6	x_4	x_5
$fobj1$	$\frac{-1}{3}$	$\frac{-2}{3}$	$\frac{-4}{3}$	$\frac{4}{3}$	$\frac{-1}{3}$	1
$fobj2$	-7	$\frac{3}{2}$	$\frac{7}{2}$	$\frac{-7}{2}$	$\frac{7}{2}$	0
x_3	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{-1}{6}$	$\frac{1}{6}$	$\frac{-1}{6}$	0
x_7	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{4}{3}$	$\frac{-1}{3}$	$\frac{1}{3}$	-1

The optimization will be carried out on the first objective function, the corresponding row of the second objective function will be affected by the usual operations in the simplex method.

We should emphasize that we will indicate the use and result of any function f through the notation:

8) f[arguments] $\rightarrow$ result.

$$findmin[row[1],\ 2] \rightarrow \{-8,\ \{4\}\}$$

The minimum of the row 1 (from position 2 on) is - 8 and is reached in the column 4. The reader can show this easily observing table 1. This step corresponds to the selection of the not basic variable that will enter to the base. Note that, in simple cases as this one, it is not necessary the use of the function *findmin*, since this step can be accomplished by cheking the table at 1.

9) *mincocient*[colum[1], colum[4]] $\rightarrow \{\frac{1}{3},\ \{3\}\}$

For selecting the variable that leaves the base we emphasize that the user must know how to apply this step. That is to say, the selection criterion of this variable so that he could decide correctly the arguments of *mincocient* and even, he must know that this is the function that sholud be used for the simplex, since also exists *maxcocient*. In this case the result of this function shows that the variable that leaves the base is the one which corresponds to the third row of table 1, x_6. If the corresponding column to the non basic variable that would enter to the base does not have positive elements, the function *mincocient* returns a list whose second argument is the empty list (corresponding to a not bounded problem).

10) $simplexpiv[3, 4] \rightarrow \{-\frac{1}{3}, \{x_1 -> 0, x_2 -> 0, x_3 -> \frac{1}{3}, x_4 -> 0, x_5 -> 0, x_6 -> 0, x_7 -> \frac{1}{3}\}\}$

The function *simplexpiv* is the pivoting of the Simplex, the output of this function is a list of two components, the first one is the value of the objective function and the second one is a list of the values of the variables. Therefore, the value of the objective function is $-\frac{1}{3}$ and the values of the variables are: $x_1 = x_2 = x_4 = x_5 = x_6 = 0$, $x_3 = \frac{1}{3}$ y $x_7 = \frac{1}{3}$.

11) $showtable[] \rightarrow$ (Table 2 pag. 8)

Next iteration step is similar: 12) $findmin[row[1], 2] \rightarrow \{-\frac{4}{3}, \{3\}\}$

13) $mincocient[colum[1], colum[3]] \rightarrow \{\frac{1}{4}, \{4\}\}$

14) $simplexpiv[4, 3] \rightarrow \{0, \{x_1 -> 0, x_2 -> \frac{1}{4}, x_3 -> \frac{3}{8}, x_4 -> 0, fx_5 -> 0, x_6 -> 0, x_7 -> 0\}\}$

15) $showtable[] \rightarrow$table 3.

Table 3.

		x_1	x_7	x_6	x_4	x_5
$fobj1$	0	0	1	1	0	0
$fobj2$	$-\frac{63}{8}$	$-\frac{1}{4}$	$-\frac{21}{8}$	$-\frac{21}{8}$	$\frac{21}{8}$	$\frac{21}{8}$
x_3	$\frac{3}{8}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{8}$	$-\frac{1}{8}$	$-\frac{1}{8}$
x_2	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{3}{4}$	$-\frac{1}{4}$	$\frac{1}{4}$	$-\frac{3}{4}$

We have already reached the optimal conditions for fobj1, therefore we are able to obtain a basic initial solution to optimize fobj2.

$delrow[i]$: Deletes the ith row of the Simplex table and eliminates any information of the variable corresponding to this row. If we apply *delrow* to eliminate the corresponding row in $fobj1$ we have:

16) $delrow[1]$

The definitions of the $funobj$ and the simplex table have changed.

17) $funobj \rightarrow \{-5x_1 - 21x_3\}$; 18) $showtable[] \rightarrow$ table 4.

Table 4.

		x_1	x_7	x_6	x_4	x_5
$fobj1$	$-\frac{63}{8}$	$-\frac{1}{4}$	$-\frac{21}{8}$	$-\frac{21}{8}$	$\frac{21}{8}$	$\frac{21}{8}$
x_3	$\frac{3}{8}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{8}$	$-\frac{1}{8}$	$-\frac{1}{8}$
x_2	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{3}{4}$	$-\frac{1}{4}$	$\frac{1}{4}$	$-\frac{3}{4}$

Table 5.

		x_1	x_4	x_5
$fobj1$	$-\frac{63}{8}$	$-\frac{1}{4}$	$\frac{21}{8}$	$\frac{21}{8}$
x_3	$\frac{3}{8}$	$\frac{1}{4}$	$-\frac{1}{8}$	$-\frac{1}{8}$
x_2	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$	$-\frac{3}{4}$

19) $varNbas \rightarrow \{1, 7, 6, 4, 5\}$

The artificial variables (x_6, x_7) take part of the non-basic variables and they correspond to the columns 3 and 4 of table 4. In next step of the Two Phases Method these artificial variables should be eliminated.

20) $delcol[\{3, 4\}]$; 21) $varNbas \rightarrow \{1, 4, 5\}$; 22) $showtable[] \rightarrow$table 5.

3.2. **Example 2:** We will show now the solution of problem (1) by straightforward computations. If we wish to work with the Two Phases Method without optimizing each objective function by means of simplex steps, we have the functions $simplexLPfobj$ and $simplexLP1$.

23) $clear[]$:

We apply the instructions 1, 2, 3, 4 to introduce the variables, constrains, the objective function and the basic variables. Then we apply 6 for the initiation of the problem.

24) $simplexLPfobj[] \rightarrow \{0, \{x_1 -> 0, x_2 -> \frac{1}{4}, x_3 -> \frac{3}{8}, 0,$
$x_5 -> 0, x_6 -> 0, x_7 -> 0\}\}$; 25) $showtable[] \rightarrow$table 3.

If we evaluate the variables *morecol* and *coladit* after applying $simplexLPfobj$ we have:

$morecol \rightarrow True$; $coladit \rightarrow 2$

Suppose we have applied $simplexLP1$, the difference with above is setted by the fact that after the evaluation of this function the values of *morecol* and *coladit* are $False$ and 0 respectively.

After we have obtained table 3 we proceed as in example 1 to reach table 5. Then we can apply any of the two functions to optimize $fobj1$, $simplexLP$ can not be used due to the fact that the problem has been already initialized, i.e. we have used the *begin* command.

26) $simplexLPfobj[] \rightarrow \{-\frac{31}{4}, \{x_1 -> \frac{1}{2}, x_2 -> 0, x_3 -> \frac{1}{4},$
$x_4 -> 0, x_5 -> 0\}\}$; 27) $showtable[] \rightarrow$ table 6.

Table 6.

		x_2	x_4	x_5
$fobj1$	$-\frac{31}{4}$	$\frac{1}{2}$	$\frac{11}{4}$	$\frac{9}{4}$
x_3	$\frac{1}{4}$	$-\frac{1}{2}$	$-\frac{1}{4}$	$\frac{1}{4}$
x_1	$\frac{1}{2}$	2	$\frac{1}{2}$	$-\frac{3}{2}$

Note that we have applied $simplexLP1$ to obtain table 3, then in order to apply $simplexLP1$ or $simplexLPfobj$ to obtain table 6 the auxiliary columns must be rebuild internally so as to avoid the cycles. In the way we have proceed for obtaining table 3, the auxiliary columns remain and there is no need of rebuilding them.

In the cases when the number of basic variables varies, the function $simplexLP$-$fobj$ should not be executed and in case that we use it the function should be applied after the evaluation of this function. The function $delcolaux$ deletes the auxiliary columns, but this process would be equivalent to apply $simplexLP1$ (see example 3).

We will show now how to solve the problem (1) with the function $simplexLP$. First, we apply the instruction 23 to start other situation, then we execute the instructions 1, 2, 3. To apply $simplexLP$ the initial basic variables are not introduced, neither the problem is initialized with the function $begin$; all these steps correspond to the internal work of $simplexLP$.

28) $simplexLP \rightarrow \{-\frac{31}{4}, \{x_1 -> \frac{1}{2},\ x_2 -> 0,\ x_3 -> \frac{1}{4},\ x_4 -> 0,\ x_5 -> 0\}\}$

We will see below we will see an example related to the integer linear problem. We will work with the same problem (1) specifying, that all the variables are integers.

3.3. **Example 3.**

We show how to solve problem (1) applying the algorithm Gomory1. We proceed first, step by step.

31) $funobj = -5x_1 - 21x_3$

32) $restric = \{x1 - x2 + 6x3 - x4 == 2,\ x1 + x2 + 2x3 - x5 == 1\}$

33) $simplexLP[] \rightarrow \{-\frac{31}{4}, \{x_1 -> \frac{1}{2},\ x_2 -> 0,\ x_3 -> \frac{1}{4},\ x_4 -> 0,\ x_5 -> 0\}\}$

34) $showtable[] \rightarrow$ table 7.

Tabla 7.

		x_2	x_4	x_5
$fobj1$	$-\frac{31}{4}$	$\frac{1}{2}$	$\frac{11}{4}$	$\frac{9}{4}$
x_3	$\frac{1}{4}$	$-\frac{1}{2}$	$-\frac{1}{4}$	$\frac{1}{4}$
x_1	$\frac{1}{2}$	2	$\frac{1}{2}$	$-\frac{3}{2}$

Tabla 8.

		x_2	x_4	x_5
$fobj1$	$-\frac{31}{4}$	$\frac{1}{2}$	$\frac{11}{4}$	$\frac{9}{4}$
x_3	$\frac{1}{4}$	$-\frac{1}{2}$	$-\frac{1}{4}$	$\frac{1}{4}$
x_1	$\frac{1}{2}$	2	$\frac{1}{2}$	$-\frac{3}{2}$
s_1	$-\frac{1}{4}$	$-\frac{1}{2}$	$-\frac{3}{4}$	$-\frac{1}{4}$

35) $newvar[\,s1\,]$: Define a new variable for introducing a new constrain.

36) $addrestric[s1 - \frac{1}{2}x_2 - \frac{3}{4}x4 - \frac{1}{4}x5 == -1/4]$

The above function introduces a constrain of the type "Gomory Cut", i.e., the basic variable of that constrain is introduced by means of the function *newvar* and the rest of the variables of the constrain are non-basic

Note that the number of basic variables increases after introducing the constrain of the cut, therefore, $simplexLPfobj$ could not be used.

37) $showtable[\,] \rightarrow$ table 8.

38) $dualsimplex[\,] \rightarrow \{-\frac{42}{5}, \{x_1 -> 0, x_2 -> \frac{1}{5}, x_3 -> \frac{2}{5}, x_4 -> \frac{1}{5}, x_5 -> 0, s1 -> 0\}\}$

39) $showtable[\,] \rightarrow$ table 9.

Table 9.

		$s1$	x_1	x_5
$fobj1$	$-\frac{42}{5}$	$\frac{21}{5}$	$\frac{4}{5}$	0
x_3	$\frac{2}{5}$	$-\frac{1}{5}$	$\frac{1}{5}$	0
x_4	$\frac{1}{5}$	$-\frac{8}{5}$	$-\frac{2}{5}$	1
x_2	$\frac{1}{5}$	$\frac{2}{5}$	$\frac{3}{5}$	-1

Table 10.

		$s1$	x_1	x_5
$fobj1$	$-\frac{42}{5}$	$\frac{21}{5}$	$\frac{4}{5}$	0
x_3	$\frac{2}{5}$	$-\frac{1}{5}$	$\frac{1}{5}$	0
x_4	$\frac{1}{5}$	$-\frac{8}{5}$	$-\frac{2}{5}$	1
x_2	$\frac{1}{5}$	$\frac{2}{5}$	$\frac{3}{5}$	-1
$s2$	$-\frac{2}{5}$	$-\frac{4}{5}$	$-\frac{1}{5}$	0

We realize in table 9 that the variables are not integers therefore we should execute the next iteration of the algorithm.

40) $newvar[s2]$;

41) $addrestric[s2 - \frac{4}{5}s1 - \frac{1}{5}x_1 == -2/5]$

42) $showtable[\,] \rightarrow$ table 10.

43) $dualsimplex[\,] \rightarrow \{-10, \{x_1 -> 2, x_2 -> 0, x_3 -> 0, x_4 -> 0, x_5 -> 1, s1 -> 0, s2 -> 0\}\}$

44) $showtable[\,] \rightarrow$ table 11.

Table 11.

		$s1$	$s2$	x_2
$fobj1$	-10	1	4	0
x_3	0	-1	1	0
x_4	0	-2	1	1
x_5	1	2	-3	-1
x_1	2	4	-5	0

As it can be observed in table 11, it is optimal, then the problem (1) with the variables restricted to be integers is solved by Gomory 1 in two iterations.

We see now, with the function *gomory*1, the direct way of solving the problem; we apply *ngomory*1 to obtain the number of iterations and to compare with the obtained result above in the solution by steps. We execute the operations 29, 30, 31, 32.

45) $ngomory1[\,] \rightarrow \{-10, \{x_1 -> 2, x_2 -> 0, x_3 -> 0, x_4 -> 0, x_5 -> 1, go1 -> 0, go2 -> 0\}, 2\}$

The function *ngomory*1 returns a list, where the last component indicates us that the problem was solved in 2 iterations of the algorithm Gomory 1.

4. Further directions in PLEMATH.

There are many directions in which PLEMATH can be extended. We will state those that we consider more important; moreover, those we are working and in those which we are working.

1. Incorporate in the algorithm Gomory 1 the requirements to avoid the degenerated solutions of the Dual after solving in the first step of gomory the continuous problem. If the degenerated dual solution were kept without making transformations on the algorithm, the finite character of the algorithm would be affected.
2. Implement specialized algorithms for binary variables.
3. Incorporate the algorithm of the Primal Cut for the integer linear problem and the algorithms of Branch and Bound and Gomory 2 for the problem of mixed linear programming.
4. Incorporate efficient algorithms to solve linear programming problems and integer linear programming problems directly.
5. Update the notebooks examples and the Help while the system is improved.

Conclusions.

The PLEMATH System has been made particularly to be used for teaching in the specialities of Mathematica and Computer Science. The principal objective that we pursued with this article is to contribute to the divulging of PLEMATH, as well as to show the simplicity and the wide possibilities that the system has for its utilization in these specialities.

PLEMATH can also be used for specialities and courses that use the topics of linear programming, and it is of particular interest in those where the study of the algorithms plays a principal role. We are working towards PLEMATH could be used in a future in direct calculations for real applications.

References

[1] Calvert E. James, Voxman L. Willian (1989). "Linear Programming". Harcourt Brace Jovanovich, Inc.

[2] Garfinkel S. Robert, Nemhauser L. George (1972). "Integer Programming". John Wiley & Sons, Inc.

[3] Handy A. Taha (1997). "Operations Research, An Introduccion, 6a. Ed". Prentice-Hall, Inc. A Simon & Schuster Company.

[4] Mokhtar S. Bazara, John J. Jarvis, Hanif D. Sherali (1998). "Linear Programming and Network flows". John Wiley & Sons, Inc.

[5] Simonnard M (1972). "Programación Lineal".

[6] Wayne L. Winston (1991). "Operations Research - Aplications and Algorithms, 2nd Ed. PWS - Kent Publishing Company.